CA PanAPT®

Administration Guide

r3.1

Second Edition

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 4: Modeling Facility 97

Chapter 1: Planning the Implementation

This section contains the following topics:

About Implementation Planning

Successful implementation of CA PanAPT requires that you review your current operational procedures. As with most systems, the decisions you make during initial implementation can be modified or enhanced later when you better understand the use of the system.

This chapter explains how to plan your CA PanAPT implementation. It tells you what decisions must be made. You can find the forms to record your implementation decisions and to assist you in the actual implementation of CA PanAPT in the Appendix A.

You need to make decisions concerning:

- Groups (for example, user ID Setup, Group Definition, Approval Category Setup)
- User Authority and Security Decisions
- Security Control
- Approval Categories (for example, user ID Setup, Group Definition,
- Approval Category Setup, Library Code Setup)
- Verification Categories
- Online Inventory Usage (for example, Library Code Setup)
- Library Codes (for example, Library Code Setup)
- Retrieve Processing

This chapter covers the decisions that must be made in the above categories.

Group and Operation Personnel Decisions

User-Defined Groups

You can assign authority for CA-PanAPT functions to a specific user ID, a set of specific user IDs, or you can use the flexible CA-PanAPT Group feature. Groups let you assign authority and responsibility without using specific user IDs. The major advantage in using groups is that you can more easily maintain system security.

You must first decide what classification of users will perform each CA-PanAPT function. You then give each classification a CA-PanAPT group name.

When you add user IDs to CA-PanAPT, you can assign them to one or more CA-PanAPT Groups. As specific user IDs change, your configuration of CA-PanAPT security remains the same. You enforce security by simply updating user IDs to add or remove group authority or membership.

You can define as many groups to CA-PanAPT as you want. A group name is any combination of one-to-eight alphanumeric characters. A single CA-PanAPT user can be included in up to five groups.

Group Admnistrator

In addition to assigning a user ID to a group, you can indicate that a user ID is a group administrator. You can use the CA-PanAPT security panels to limit action to group administrators only. For activities relating to Inventory Records, Move Requests and user ID records, a group administrator must share the same group as the owner to perform the activity. For all other activities, any group administrator can perform the function.

Operations

CA-PanAPT has an internally defined classification called Operations, which you can assign to a user ID. This assignment is in addition to the allowable (up to five) site-defined groups that can be specified for a user ID, that is, it is possible for a user ID to be assigned to five user-defined groups and also be assigned the Operations attribute. Users responsible for the daily operation of CA-PanAPT are generally given the Operations attribute.

Setting Up Groups and Operations Personnel

Important implementation decisions to make include what groups you want to use and what authority you want to assign to those groups. The CA PanAPT Group Set-up Form is provided in Appendix A to help you to make and to record these decisions. Copy the provided form, and fill it out as you proceed through implementation.

This form has two columns.

The first column lists the departments, areas, or general user types at your site that will use CA PanAPT. For example, you can list the following user types:

- Data Center
- Production Control
- Project Leaders
- Documentation Supervisor
- Programmers
- Quality Assurance

Column two of the form contains your CA PanAPT Group Name. Write the name you give to the CA PanAPT group next to each department, area, or group that you have listed.

Given the above example, you can choose to assign the following group names:

Name of Department or Area	CA PanAPT Group Name
Data Center	OPERATIONS attribute
Production Control	PRODCNTL
Project Leaders	PROJLDR
Documentation Supervisor	DOCSUP
Programmers	PROG
Quality Assurance	QUAL

To print a Cross-Reference Report listing all groups defined to CA PanAPT, users who belong to each group, and the authorized activities for each group, use JCLLIB member APJJ5105. This report can also be viewed online. Whenever you are asked to enter a CA PanAPT group name, check this report for the correct spelling. If you add another group later, run the report again to keep it current.

User Authority and Security Decisions

This topic describes the security authorization that CA PanAPT provides for each activity. You can further control authorization through the use of a user-written Security Exit Program (see Appendix B). The security exit can be activated individually for the following activities:

- MOVEREQ/APM-*level*
- MOVEREQ/APB-*level*
- MOVEREQ/MEMBER
- MOVEREQ/MEMCHG
- MOVEREQ/MEMDEL
- MOVEREQ/MEMPURGE

It is invoked after CA PanAPT security authorization has verified access to the requested activity.

How Authorization Works

The charts in the following tables summarize which users can perform a function, given a certain authorization on an Activity Record. Each column heading represents a field in an Activity Record. To see the results of setting the Group Administrators? flag to Y, look in the Group Admin. column. You can see that system administrators and group administrators in the owner's group can perform the activity, and no one else. If more than one flag in the Activity Record is set to Y, any category with a Y in either column can perform the function.

The owner of a Move Request is the user who added the Move Request. During user ID maintenance, the owner is the user ID being accessed.

The owner of an Inventory Record is determined as follows:

- Inventory Record maintenance actions (Add, Change, Approve, Delete, and Inquire).
  - The user ID in the Owner field (the permanent owner) is used.
  - If that field is blank, then the user ID in the Assigned-To field is used.
- Inventory Assignment actions (Assign, Retrieve, Transfer, Release). The user ID in the Owner field is always used. (The user ID in the Assigned-to field is always blank.)

The tables on the following pages summarize which users can perform an activity, given a certain authorization on an Activity Record.

Each vertical column represents a field on the Activity Record. For example, the Share group column on the first chart indicates which users can perform an activity on Move Requests, Inventory Records, or user IDs in which the users sharing a group? field is Y and the rest of the fields are N.

The table shows that system administrators, the owner, and any users who share a group with the owner can perform this activity, and users who do not share a group with the owner cannot. (In the left-hand column, Operations is a user ID with the Operations attribute who does not belong to any groups. It is possible for a user ID with the Operations attribute to also fall into another category, such as sharing a group with the owner.)

Note: If a specific user ID is listed on an Activity Record, that user can always perform that activity. If a group is listed, any user in that group can always perform that activity. No checking of groups or ownership is done in these instances.

The following table describes authorization for activities pertaining to Inventory Records, Move Requests, and user ID maintenance.

Designator	Operations	Group Admin	Owner	Share Group	Anyone
System Administrator	Y	Y	Y	Y	Y
Operations	Y	N	N	N	Y
Group Admin. (G.A.) – same group	N	Y	N	Y	Y
G.A. – other – shared group	N	N	N	Y	Y
G.A. – other – no shared group	N	N	N	N	Y
Share a group	N	N	N	Y	Y
Owner	N	N	Y	N	Y
Ordinary user – no shared group	N	N	N	N	Y

Key to table:

System Administrator

A specially designated CA-PanAPT user who can always use any CA-PanAPT activity. There can be more than one CA-PanAPT system administrator.

Operations

A user ID who does not belong to any group. A user ID with the Operations attribute might also belong to other categories.

Group Admin - same group

A user ID who shares a group with the owner and is an administrator in that group.

G.A.—other - shared group

A user ID who shares a group with the owner and is an administrator in a different group.

G.A.—other -no shared group

A user ID who is an administrator in a group but does not share any groups with the owner.

Share a group

A user who shares a group with the owner of the Move Request or Inventory Record.

Owner

The user who adds a Move Request is the owner of that Move Request. That user is then given authority to perform functions allowed to Move Request owners against his/her own Move Requests.

Ordinary user

No shared group

The following table describes authorization for all other activities. The concept of owner, and therefore, sharing a group with the owner has no meaning for these activities.

Designator	Operations	Group Admin	Owner	Share Group	Anyone
System Administrator	Y	Y	Y	Y	Y
Operations	Y	N	N	N	Y
Group Admin – any group	N	Y	N	N	Y
Share a group	N/A	N/A	N/A	N/A	N/A
Owner	N/A	N/A	N/A	N/A	N/A
Ordinary user – no shared group	N	N	N	N	Y

How to Authorize a User

Authorizing a CA PanAPT user to perform CA PanAPT functions is a two-step process.

1. Indicate what classification of CA PanAPT users can perform a function.
2. Place the CA PanAPT user into the appropriate classification.

If a user ID is not added to CA PanAPT, a user ID of *DEFAULT is used when performing authorization tests. *DEFAULT can be granted authorization as if it were a real user ID.

Many ways exist to authorize users for specific functions.

Note: As delivered, CA PanAPT gives the *DEFAULT user authority to any user who signs on. As delivered, *DEFAULT has CA PanAPT System Administrator authority. After you have added one or more user IDs with Administrator authority, remove Administrator authorization from *DEFAULT using Control File Maintenance panels as described in the "Control File" chapter in the *CA PanAPT Reference Guide*.

How to Authorize a Group Assignment

Group Assignment assigns the authority for a function to a group name. When you add the user ID, assign that group to that user ID.

Up to ten group names can be authorized for each function.

The designations of Group Administrator, owner, and users sharing a group, have meaning only for Inventory Members, Move Requests, and user ID records.

Note: Using the *DEFAULT user ID for authority might degrade performance.

How to Authorze a Group Administrator

Group Administrator assigns the authority for a function to a group administrator. When you add the user ID, assign a group to that user ID and indicate that the user ID is a group administrator for that group. He or she can then perform the function on Move Requests, Inventory Records or user ID records that are owned by users who are in that group. See below for the definition of owner.

For activities that do not relate to Move Requests, Inventory Records or user ID maintenance, this flag indicates that any group administrator in any group can perform this function.

How to Authorize an Owner

The owner for a Move Request, Inventory, or User ID maintenance functions assigns the authority for a function to the owner. In this way, users can be given functional authority against a limited subset of Move Requests, Inventory Records, or User IDs.

The user who adds a Move Request is the owner of that Move Request. That user is then given authority to perform functions allowed to Move Request owners against his or her own Move Requests.

Inventory Records can have two possible owners. For Inventory maintenance actions (Add, Change, Delete, Inquire and Approve) the owner is the permanent owner of the Inventory Record — that is, the user ID shown in the Owner field of the Inventory Record. If the Owner field is blank, the assigned-to user ID is used.

For Inventory Assignment actions (Assign, Retrieve, Transfer and Release) the owner is the user ID to whom the Inventory Record is assigned. If the assigned-to user is blank, the user ID of the permanent owner is used.

The owner of a user ID Record is the user ID itself.

How to Authorize Users Sharing a Group

Allow Users Sharing a Group grants the authority to any CA-PanAPT user who shares a group with the owner of the Move Request or Inventory Record, or with the user ID being acted upon.

If used, it indicates that Allow Users Sharing a Group authority has been given to this activity and it means that any other CA-PanAPT user who shares any group with the owner can perform this function. If the user ID being given authority is also the owner, the user cannot perform this function. So, if Sharing a Group is set to Y, but Owner is set to N, the owner cannot perform the activity.

Note: If the owner of an Inventory Record is a group, and users sharing a group? is set to Y for a certain activity, for any member of the group to perform the activity, the group must be defined to CA-PanAPT as a user ID, and the group must be listed on its own user ID record.

How to Authorize Operations

Operations assigns the authority for a function to CA-PanAPT users with the Operations attribute.

When you add a user ID to CA-PanAPT, you can assign that user to the CA-PanAPT internal classification of Operations. That user is now allowed all activities assigned to the Operations classification.

How to Authorize Anyone

Allow Anyone grants the authority to any CA PanAPT user. Every user who is able to use CA PanAPT is able to perform this function. Inquiry and browse functions can be granted to all users.

How to Authorize the CA-PanAPT System Administrator

The CA-PanAPT system administrator can always perform all CA-PanAPT functions.

When you add a user ID, you can indicate that this user ID is a CA-PanAPT system administrator. While there is no limit to the number of user IDs that can be designated as CA-PanAPT system administrators, we recommend that you keep the number small. Usually two or three system administrators are sufficient. Always have more than one.

You should *not* use CA-PanAPT system administrator user IDs for day-to-day CA-PanAPT work because they bypass authority checks. If system administrators do day-to-day CA-PanAPT work, they should have a second user ID that does not have System Administrator authority.

How to Allow a User ID, Set of User IDs, or a Group

Allow a user ID, set of user IDs, or a group indicates that a specific user ID or group can perform a specific CA-PanAPT function. That user ID or group is automatically allowed to perform that function.

You can authorize up to ten specific user IDs and groups for each function.

Planning CA-PanAPT Security

When establishing CA-PanAPT security, you determine who performs CA-PanAPT functions.

During Phases One and Two of implementation, you should change as little of the delivered security as possible. However, you should specify your own system administrators and remove System Administrator authority from *DEFAULT.

As you use the system in your own site, implementation of more sophisticated security grows naturally, instead of as a difficult obstacle to overcome before beginning. Report APCS5103-01 lists security definitions for Activities.

As you review the CA-PanAPT functions, decide which group or which user ID is responsible for performing each one. You can authorize *DEFAULT to perform the functions needed by the average user to minimize the volume of maintenance required.

Use the Control File Security Form in Appendix A to record your decisions. When you decide to authorize a group for a CA-PanAPT activity, refer to report APCS5105-01 to see the proper group name.

Planning Activity Authorization

The Control File Security Form has the following format:

Description of Activity

Brief description of a CA-PanAPT Activity.

Internal Activity Code

When updating the CA-PanAPT Activity to establish authority, you are asked to provide the Activity Code. This column shows the CA-PanAPT Activity Code that must be entered.

Allow to

Any User
Shared Group
Owner
Group Administrator
Operations
These Groups/User IDs

For each activity, indicate the type of user ID you want to give authorization.

You can enter up to ten specific TSO user IDs or ten CA-PanAPT group names in the last column.

You can activate a user-written Security Exit for activities for which the Security Exit is applicable. As delivered, the Security Exit Status is Inactive or Not Applicable.

1. Accessing/Updating the CA-PanApt Control File

Activity Code	User
Activity (CTL/ENTRY)	Who can enter Control File Maintenance?

Activity Code	User
Activity (CTLACT/CHG INQ)	Who can change CA-PanAPT system authority? This means who decides which user IDs or groups can perform CA-PanAPT activities? Who can look at current authorities? **Note:** You cannot add or delete any Activity records. To deactivate the function, remove all authorizations. The security delivered with the system lets any CA-PanAPT user look at the activity records and see which user IDs and groups can perform functions (INQ), but allows change (CHG) only by the CA-PanAPT system administrator.
User ID Authorization (CTLUSER/ADD CHG DEL INQ)	Who can add user IDs to CA-PanAPT, change their authority levels, delete user IDs, or look at (Inquire) the current user IDs? The security delivered with the system lets any CA-PanAPT user look at the user IDs (Inquire) but allows Add, Update, or Delete only by the CA-PanAPT system administrator.
System Information (CTLSYS/ADD CHG DEL INQ)	Who is responsible for adding, changing, deleting, and inquiring (looking at) your CA-PanAPT system information? The security delivered with the system lets any CA-PanAPT user look at the information (INQ) but allows Update only by the CA-PanAPT system administrator. **Note:** Add and delete only pertain to Access Methods and Configuration Manager information.

2. Accessing/Updating Inventory Records

Activity Code	User
Activity (INVENTRY/ENTRY)	Who can enter Inventory File Maintenance?
Adding Inventory Records (INVENTRY/ADD)	Who can add Inventory Records? The delivered system allows any user to add Inventory Records through the online system.

Activity Code	User
Approving Inventory Records (INVENTRY/APP)	Who can approve Inventory Records? The delivered system allows only the CA-PanAPT system administrator to approve Inventory Records.
Assigning Inventory Records (INVENTRY/ASN)	Who can assign Inventory Records for use during Move Requests? The delivered system allows all CA-PanAPT users to assign Inventory Records.
Updating Inventory Records (INVENTRY/CHG)	Who can update (change) Inventory Records? The delivered system allows group administrators in the owner's group, the Inventory Record owner, or users who share a group with the owner to update Inventory Records.
Deleting Inventory Records (INVENTRY/DEL)	Who can delete Inventory Records? The delivered system allows the group administrators in the owner's group, the Inventory record owner, or users who share a group with the owner to delete Inventory Records.
Viewing Inventory Records (INVENTRY/INQ)	Who can see Inventory Records? The delivered system allows all CA-PanAPT users to Inquire (view) Inventory Records.
Releasing Inventory Records (INVENTRY/REL)	Who can release Inventory Records? The delivered system allows operations personnel, group administrators in the owner's group, the Inventory Record owner, or users who share a group with the owner to Release Inventory Records.
Assigning Inventory Records with Retrieve (INVENTRY/RET)	Who can assign Inventory Records with Retrieve processing for use during Move Requests? The delivered system allows all CA-PanAPT users assign Inventory Records with Retrieve.

Activity Code	User
Transferring Inventory Records (INVENTRY/TRN)	Who can transfer Inventory Record Assignment from one user to another? The delivered system allows operations personnel, group administrators in the owner's group, the Inventory Record owner, or users who share a group with the owner to Transfer Inventory Records. **Note:** See How to Authorize a User earlier in this chapter for the definition of owner of an Inventory Record.

3. Accessing/Updating Library Codes

Activity Code	User
Activity (LIBCODE/ENTRY)	Who can enter Library Code Maintenance?
Activity (LIBCODE/ADD CHG COP DEL INQ)	Who is responsible for setting up and maintaining Library Code Definitions? Who can look at Library Code Definitions? The security delivered with the system lets any CA-PanAPT user look at the Library Codes but gives the ability to Add, Copy, Update (Change), and Delete Library Codes only to the CA-PanAPT system administrator.
Activity (LIBCODE/SECURITY)	Who can view the security fields for libraries in a Library Code definition? You can indicate that a user can view Library Code definitions but cannot view the security information for that Library Code. You can indicate which users who have viewing (Inquiry) authority can also view the security fields.

4. Accessing/Updating Move Requests

Activity Code	User
Activity (MOVEREQ/ENTRY)	Who can enter Move Request Maintenance?

Activity Code	User
Adding Move Requests (MOVEREQ/ADD)	Who can add Move Requests? The delivered system allows any CA-PanAPT user to add Move Requests.
Approving Move Requests (MOVEREQ/APM-*level*)	Who can approve Move Requests before the move to is done? Each move level is separately authorized. Those who should approve movement into Quality Assurance are not necessarily the same as those who should approve movement into Production. The delivered system allows the operations personnel to approve Move Requests into QA and Production. Users must also be authorized to grant specific approvals (1-21) using the User ID Maintenance panel. When another move level is defined to CA-PanAPT a MOVEREQ/APM-*level* activity record is created for it, with no one authorized to perform the approval. You must update this activity record before any approvals at the new level can be granted. The user-written Security Event Exit can further control users when approving Move Requests. See Appendix B.

Activity Code	User
Approving Move Request for Back Out (MOVEREQ/APB-*level*)	Who can approve Move Requests before backing them out? Each move level is separately authorized. Those who should approve Quality Assurance back outs are not necessarily the same as those who should approve Production back outs. The delivered system allows the operations personnel approve Move Requests for QA and Production back outs. Users must also be authorized to grant specific approvals (1-21) using the User ID Maintenance panel. When another move level is defined to CA-PanAPT a MOVEREQ/APB-*level* activity record is created for it, with no one authorized to perform the approval. You must update this activity record before any back out approvals at the new level can be granted. The user-written Security Event Exit can further control users when approving Move Requests for back out. See Appendix B.
Creating an Online List of Move Requests (MOVEREQ/BRO)	Who can use the Browse function to create an online list of Move Requests? The delivered system lets any user use the Browse function.
Updating Move Requests (MOVEREQ/CHG)	Who can update Move Requests before they are closed? The delivered system allows the following users to change Move Requests: the user who added the request (the owner), any user who shares a group with the owner, group administrators in the owner's group, and users with the Operations attribute.

Activity Code	User
Changing Closed Move Requests (MOVEREQ/CHGAWAPP)	Who can change Move Requests while awaiting approval? A Move Request cannot be changed once all approvals have been granted. The delivered system lets operations personnel change the Move Request while it is awaiting approvals. Once all approvals have been granted, the status of the Move Request must be changed to Being Created or Awaiting Approval before it can be changed. Changing the status removes all authorizations granted to date. See the Move Request Maintenance (CHG) function described in the "Move Requests" chapter in the *CA-PanAPT Reference Guide* for more information on this function.
Closing Move Requests (MOVEREQ/CLO)	Who can close Move Requests? The delivered system allows the following to close Move Requests: the user who added the request (the owner), any user who shares a group with the owner, group administrators in the owner's group, and operations.

Activity Code	User
Assignment Requirements At Close (MOVEREQ/CLOASSGN)	What criteria will be used to determine whether a member is properly assigned when a Move Request is closed? Proper assignment is determined by testing the relationship specified in this activity against two user IDs. The Close Assignment option under Control File Maintenance System Information specifies which user IDs to use for the test. The *CA-PanAPT Reference Guide* describes proper assignment in greater detail. The delivered system allows users who meet any of the following tests to fulfill assignment requirements: ■ The user who added the Move Request ■ Any user who shares a group with the Move Request owner ■ Group administrators in the Move Request owner's groups ■ Users with the operations attribute.
Back Out a Move Request (MOVEREQ/BAK)	Who can set up a Move Request for Back Out processing? The delivered system allows no users to back out Move Requests. Access to this activity is granted by the CA-PanAPT system administrator once the appropriate models, JCL, and Library Codes have been set up to perform this action.
Copying a Move Request (MOVEREQ/COP)	Who can Copy the contents of one Move Request to create a new one? The delivered system allows any CA-PanAPT user to copy Move Requests.
Activity (MOVEREQ/CR)	Who can copy the contents of one Move Request to create another one for rework purposes? The delivered system only allows the CA-PanAPT system administrator to do this.

Activity Code	User
Changing Move Dates for Move Requests (MOVEREQ/DAT)	Who can change the Move date for Move Requests? The delivered system lets the Operations personnel to change the Move Request Move Date.
Deleting Move Requests (MOVEREQ/DEL)	Who can delete Move Requests before they are closed? The delivered system allows the Operations personnel, the user who created the Move Request, any user who shares a group with the owner, and group administrators in the owner's group to delete the Move Request.
Deleting Closed Move Requests (MOVEREQ/DELCLOSD)	Who can delete closed Move Requests? The delivered system lets the Operations personnel delete closed Move Requests.
Viewing Move Requests (MOVEREQ/INQ)	Who can see Move Requests (Inquire)? The delivered system lets any user inquire Move Requests.
Adding Members to a Move Request (MOVEREQ/MEMBER)	Who can add members to a Move Request? The delivered system allows any CA-PanAPT user to add members to a Move Request. For the online preparation of Move Requests to be streamlined, keep the authorizations for MOVEREQ/MEMBER the same as those for MOVEREQ/ADD and MOVEREQ/CHG. The user-written Security Event Exit can further control users from adding members to a Move Request. See Appendix B.
Changing Members in a Move Request (MOVEREQ/MEMCHG)	Who can change members in a Move Request? The delivered system allows any CA-PanAPT user to change members in a Move Request. For the online preparation of Move Requests to be streamlined, keep the authorizations for MOVEREQ/MEMCHG the same as those for MOVEREQ/ADD and MOVEREQ/CHG. The user-written Security Event Exit can further control users from changing members in a Move Request. See Appendix B.

Activity Code	User
Deleting Members from a Move Request (MOVEREQ/MEMDEL)	Who can remove members from a Move Request? The delivered system allows any CA-PanAPT user to delete members from a Move Request. For the online preparation of Move Requests to be streamlined, keep the authorizations for MOVEREQ/MEMDEL the same as those for MOVEREQ/MEMBER. The user-written Security Event Exit can further control users from deleting members from a Move Request. See Appendix B.
Deleting Members from a Move Request and the Test Libraries (MOVEREQ/MEMPURGE)	Who can remove members from a Move Request and scratch them from the Test Libraries at the same time? The delivered system allows any CA-PanAPT user to delete members from a Move Request and the Test Library. The user-written Security Event Exit can further control users from purging members from a Move Request and Test Library. See Appendix B.
Printing Move Requests (MOVEREQ/PRT)	Who can print a hard-copy authorization form for a Move Request? The delivered system lets any user print a Move Request authorization form.

Activity Code	User
Run Verification Procedures (MOVEREQ/RVP)	Who can run Verification Procedures for a Move Request? The delivered system allows users currently authorized to CLOSE a Move Request, to Run Verification Procedures. This activity can be changed using online Control File maintenance.
Changing Status for Move Requests (MOVEREQ/STA)	Who can change the status of Move Requests? The delivered system lets the Operations personnel change the status of the Move Request.

5. Approving Online Report Processing

Activity Code	User
Activity (REPORT/ENTRY)	Who can enter the Online Report Facility?
Online Approval Category Report, APCS5104 (REPORT/APCS5104)	Who can run the APCS5104 report? The delivered system allows all CA-PanAPT users to run the report.

Activity Code	User
Online Group Cross-Reference Report, APCS5105 (REPORT/APCS5105)	Who can run the APCS5105 report? The delivered system allows all CA-PanAPT users to run the report.
Online Inventory Assignment Report, APCS6111 (REPORT/APCS6111)	Who can run the APCS6111 report online? The delivered system allows all CA-PanAPT users to run the report.

6. Accessing the Development Facility

Activity Code	User
Activity (DEV/ENTRY)	Who can enter the Development Facility?
Activity (DEV/MM)	Who can use the MM (Modify Members) command from the Development Facility MSL? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (DEV/PRINTTBL)	Who can print the development member selection list from the Development Facility MSL Options panel? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/BROWSE)	Who can browse members from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/CHECKIN)	Who can check in members from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/CHECKOUT)	Who can check out members from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.

Activity Code	User
Activity (MEMBER/COMPARE)	Who can compare members from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/COMPILE)	Who can compile, link edit, or compile and link edit members from the Development Facility. The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/EDIT)	Who can edit members from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/HISTORY)	Who can view member history from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/LISTING)	Who can browse compile and link edit listings from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/MERGE)	Who can merge members from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/OUTPUTCP)	Who can view compare output from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/OUTPUTMG)	Who can view merge output from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.

Activity Code	User
Activity (MEMBER/UTILITY)	Who can perform utility functions against members from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (MEMBER/XIR)	Who can view source to executable cross-reference information from the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (DEVADMIN/ENTRY)	Who can enter Project Administration?
Activity (PROJECT/ADD CHG DEL INQ)	Who can add, maintain, delete, and view Project definitions? The delivered system only allows the CA-PanAPT system administrator to do this.
Activity (USERLIB/CHG INQ)	Who can maintain and view user libraries for use with the Development Facility? The delivered system only allows the CA-PanAPT system administrator to do this.

Security Control

CA-PanAPT provides additional security control through its Security Event Exit. The Security Event Exit permits you to control the use of certain CA-PanAPT operations.

The Security Event Exit contains two main parts:

- User-written Security Exit Program
- Security Events (these are established in CA-PanAPT)

Security Exit Program

The user-written Exit Program controls security processing. It performs the testing that permits or restricts a user from performing a CA-PanAPT operation. If the user passes this authority check, processing continues normally. If a user fails it, the Exit Program displays an error message and returns the user to the main ISPF Panel or permits the user to make another selection. (Your exit determines how the system responds by setting the appropriate response code in the pass/fail indicator.)

Only one Active Security Exit Program is allowed in a CA-PanAPT environment.

The sample security exit APCS0104 provided on <SPFX>.APTEXSRC can be used as a model for developing a security exit program to meet your security needs.

Security Events

The Exit Program performs authority checking each time it encounters a Security Event. Security Events are logical points in the CA-PanAPT environment where the Exit Program gains control. These events occur when a user:

- Enters CA-PanAPT from the ISPF Panel
- Adds a member to a move request
- Changes a member on a Move Request
- Deletes a member from a Move Request
- Purges a member from a Move Request (same as delete but also deletes the member from the test library)
- Exits CA-PanAPT

CA-PanAPT has three types of security events:

- Initialization Event (INIT)
- Activity Event (ACTIVITY)
- Termination Event (TERM)

INIT and TERM events are always taken if the Security Exit Program has been activated from the System Security Information Maintenance panel.

Once the Security Exit Program has been activated, a CA-PanAPT system administrator (usually) or another designated person activates Activity Events for individual CA-PanAPT activities through Activity Maintenance panels. You can activate one, several, or all security events in your CA-PanAPT environment.

Initialization and Termination Events

The Initialization Event (INIT) initializes your CA-PanAPT security environment and the Termination Event (TERM) shuts it down. The CA-PanAPT installation tape contains a sample Security Exit Program in a member called APCS0401 (APT User Exit) on the PDS called <spfx>.APTEXSRC. As the systems programmer, you can modify this sample program to perform whatever initialization or termination routines your shop requires.

After you have created your exit program and activated the security exit, CA-PanAPT automatically invokes the INIT and TERM events each time a user enters and exits CA-PanAPT.

Activity Event

An Activity Event (ACTIVITY) can occur when a user attempts to:

- Add a member to a Move Request
- Change a member on a Move Request
- Delete a member from a Move Request
- Purge a member from a Move Request (same as delete but also deletes the member from the test library)

If CA-PanAPT verifies that the user is authorized to perform the activity, the ACTIVITY occurs.

Security Event Activation

All security events in CA-PanAPT are inactive in the delivered system. To activate Security for your CA-PanAPT environment, you must:

1. Place your user-written Security Exit Program in a load library accessible to your CA-PanAPT environment.
2. Activate the Security Exit Program globally for the INIT and TERM events through the Control File maintenance–System Security Information panel.
3. Activate the ACTIVITY event for each CA-PanAPT activity where the ACTIVITY event is applicable.

When an event is active, the Security Exit Program determines whether the user's request for an activity can be honored (see the Activity Event topic presented earlier).

Normally, CA-PanAPT user authority is considered before invoking the Security Exit Program. CA-PanAPT user authority can be overridden by the Security Exit Program. CA-PanAPT passes the Pass/Fail indicator to the Security Exit Program to let the exit know that this user passed or did not pass CA-PanAPT authority checking. If the exit does not change the value of the Pass/Fail indicator, CA-PanAPT acts as if the exit is not present or active.

The system administrator can decide the CA-PanAPT activities where these security events will be active, allowing for further control of security processing.

When an event is inactive for an activity, CA-PanAPT follows normal user authority to determine if the user's request can be performed.

The CA-PanAPT system administrator can individually activate and deactivate the ACTIVITY event. Such flexibility enables a phased implementation of security in your CA-PanAPT environment.

Approval Category Decisions

When updating Production Libraries, the members to be updated are usually reviewed or approved before the update takes place. Sometimes no formal review takes place, but someone must be notified so that they can take some action or plan for the update.

Different types of items can be approved or reviewed by different persons or departments. Some items can be strictly controlled and require the involvement of several persons.

CA-PanAPT provides up to 20 Approval Categories to accommodate this need for formal approval, review, and notification.

Assigning Approval Categories

Part of implementation assigns Approval Categories. Then, when Move Requests are set up to move the different types, CA-PanAPT users view and grant Approvals using CA-PanAPT panels.

A two-part form, the Approval Category Set-up Form (see Appendix A) records your Approval Categories decisions. This information is then used in setting up Library Codes as well. You can enter your Approval Category Descriptions as part of CA-PanAPT Control File maintenance of system information.

Review this form as you read this chapter.

Step 1: Listing Production Types

The first thing you have to do is to make a list of the types of Production entities you have. You can easily determine this by reviewing your Production Library types.

For instance, you can have the following types:

- JCL
- Source programs
- Executable Programs
- Documentation

Enter the list you have developed in the ITEM TYPE column of the Approval Category Set-up Form.

Step 2: Assessing Approvals for Each Production Type

Review each Item type on your list, and indicate who gives the approvals, reviews, or notifications before these libraries and members can be updated. You might find yourself dividing some of your Item Types into multiple categories.

For instance, given the above example, you might find:

- JCL: 1 approval
  (Production control supervisor)
- Source programs: 2 approval
  (each project leader)
  (Production control supervisor)
- Executable (payroll): 3 approvals
  (project leader)
  (payroll supervisor)
  (Production control supervisor)
- Executable (other):2 approvals
  (project leader)
  (Production control supervisor)
- Documentation: 1 approval
  (Documentation coordinator)

Fill this information in the Description of Approval column on page 1 of the Approval Category Set-up Form.

Step 3: Assigning Approval Category Numbers

Assign each different type of approval to a CA-PanAPT Approval Category Number. Copy each different Description of Approval from column two of page one on the Approval Category Set-up Form to column two of page two.

Given the above example, you will now have:

Approval Number	Approval Category Description
1	Production Control Supervisor

Approval Number	Approval Category Description
2	Project Leaders
3	Payroll Supervisor
4	Documentation Coordinator

Note: Allow more than one line for Approval Categories that have more than four TSO user IDs in the list of Approvers. See Step 4 for an example.

Step 4: Specifying TSO User IDs of Approvers

Now, complete page two of the Approval Category Set-up Form by indicating the TSO user IDs of the persons who will use CA-PanAPT to grant approvals.

There is no limit to the number of users who can be given authority to grant approvals, use as many lines as necessary on the form, to list the user IDs of approvers for each Approval Category. See the example of page two of the Approval Category Set-up Form below.

CA-PanAPT Approval Number	Approval Category Description	TSO User IDs of Approvers			
1	Production Control Supervisor	ABC			
2	Project Leaders	DEF	GHI	JKL	MNO
		PRS	TUV		
3	Payroll Supervisor	AAD			
4	Documentation Coordinator	MLI			

Step 5: Recording the Approval Category Numbers

Finally, copy your approval category numbers from Column one of page two on the Approval Category Set-up Form to the Approvals Required column of page one.

Summary

When completed, page one of the Approval Category Set-up Form contains the Approval Categories required for each Production entity type. When you set up Library Codes, you will use this information to assign Approval Categories to the Library Codes.

Page two of this form shows the Approval Categories and which users grant those approvals. You will use this information when you set up user IDs.

You can enter your Approval Category descriptions into CA-PanAPT as part of Control File maintenance of the system information.

You can run JCLLIB member APJJ5104 to print APCS5104-01, a report that lists Approval Categories and the user IDs that are authorized to grant each approval. You can also run this report from the Online Reports selection for viewing under ISPF.

Verification Category Decisions

A Verification Category is a program or set of programs designed to analyze the attributes of a Move Request and post a pass or fail indication. Verification Categories are very similar to the 20 Approval Categories except Approval Categories are approved or disapproved by authorized people where as Verification Procedures are approved or disapproved by a specified process (for example, program). You implement your decisions by performing the following steps:

1. Determine the types of items that need verification.
2. Determine the verification process (which programs and models to use to perform the verification).
3. Assign a verification category to each type of item.

Online Inventory Usage Decisions

The CA-PanAPT Online Inventory file has several basic purposes:

- Retains information about Production Library Members for reporting
- Allows CA-PanAPT to monitor assignment of Inventory members
- Allows CA-PanAPT to Retrieve members to a user
- Stores member-related data for use in Models.

You implement your decisions by specifying options in Library Code Definitions. You can choose different options for each Library Code.

Inventory File Member Data

The Inventory file keeps a large quantity of data for each member. Essentially, the Inventory file can be used as a large directory for all Production Library members.

Information retained on the file includes:

- Member Description and Comments
- User ID and Move Request number where the member is assigned
- Date and time member was assigned
- Language
- User Application where item is used
- User Environment where item is used
- Compiler options for the member
- Link edit options for the member
- 20 freeform user data fields
- Originating Library Code

When CA-PanAPT processes a member that has an Inventory Record, CA-PanAPT has access to all of the member information, and can use the information to customize the processing for that member.

For more information on this topic, see the "Inventory Records" chapter in the *CA-PanAPT Reference Guide.*

Inventory File Assignment Release

The Inventory file Assignment/Release/Transfer functions give you control over the number and type of Move Requests that will move a specific member.

When an Inventory Member is assigned to a CA-PanAPT user, no other user can have it assigned. Users can add the member to any number of Move Requests. CA-PanAPT relies on the user to determine when it is appropriate to add a member to more than one Move Request.

When an Inventory Member is released, it is then available for any user to assign and include in a Move Request.

The Inventory Member Transfer does a release from the currently assigned-to user and an assignment to the new user.

You can set your Inventory file utilization up so that Inventory Records must be Assigned to a user before they can be processed in Move Requests. Or you can set up the Inventory file Assignment/Release scenario so that it is automatic. In that case CA-PanAPT does the assignment when a member is added to a Move Request and releases the member when it is moved to Production. If a member has been added to several Move Requests, it is reassigned to another Move Request when the first Move Request is moved to Production. The member can also be reassigned to another user, based on the value of the Reassign/Transfer flag in the Control file.

You specify Inventory assignment and release options as part of the Library Code definition. Different Library Codes can use different Inventory options.

Retrieve

Retrieve copies members from the destination libraries or their corresponding backup and back out libraries to the test level library where you will modify them. You can only Retrieve members that have been assigned to you. You can initiate Retrieve as part of the assignment request or, later, as part of a specific Retrieve request.

To retrieve members, Inventory, Assignment, and Retrieve must be enabled in their Library Code definitions. If a member is not assigned and its Library Code definition specifies Auto Assignment and Auto Retrieve, CA-PanAPT assigns the member when adding it to a Move Request and gives the user the opportunity to Retrieve the member.

Inventory Record Initialization

You can build Inventory Records in three ways:

- Use Inventory maintenance panels to build the records.
- Allow CA-PanAPT to build the records automatically when entities without Inventory Records are added to a Move Request.
- Run the batch jobs that build the Inventory Records, fill most of the required fields, and add the records to the Inventory file.

You can use any combination of the previous three methods.

CA-PanAPT provides two Jobs: APJJ6910, which reads the directory of a library, and APJJ6920, which produces Inventory Records for each member. APJJ6910 reads PDS, CA-Panvalet, and CA-Librarian library formats. APJJ6920 uses the library information from Job APJJ6910 to create Inventory Records.

Because the directory does not contain all the information fields supported by CA-PanAPT, most of these fields are blank. You must specify the Library Code where you are adding members. You must run APJJ6910 and APJJ6920 for each Library Code.

APJJ6920 creates Inventory Records for CA-PanAPT using the directory information extracted by APJJ6910. As delivered, APJJ6920 will mark all Inventory Records it creates as NOT-APPROVED. This is an indication that they contain only the basic information extracted from the library directory. They will not contain any of the optional fields that CA-PanAPT does not validate.

If your Models require specific values in any of these fields, use the CA-PanAPT panels to change the Inventory Record values (see the "Inventory Records" chapter in the *CA-PanAPT Reference Guide)*. As you review each member, mark it APPROVED as a record of your progress.

The records created during APJJ6910 execution are in the format mapped by copybook APCCDIB2. For detailed information, refer to Appendix C. As an alternative to running APJJ6910, you can write your own program to create Inventory Records in APCCDIB2 format; then run APJJ6920, using these records as input.

You can define a Library Code that requires all members to have an Approved Inventory Record before a Move Request that contains the members can be closed. This helps to ensure that inventory data is in a correct form for models. You can also add an Inventory Edit Exit to help ensure correct data.

Library Code Decisions

A Library Code is the set of characteristics that defines a type of Move. Setting up a Library Code includes specifying which libraries are involved in the Move.

Each Library Code differs from other Library Codes used in the libraries. Another difference can be which approvals are required. One Library Code can be set up to use the same libraries as another, but requires more or different approvals.

The Library Code Set-up Form is included in Appendix A to assist you in defining your Library Codes. Fill out a copy of this form for each of your Library Codes and use it as reference during the online definition of the Library Codes.

Library Code Setup

For information on setting up Library Codes, see the *CA-PanAPT Reference Guide*, Which describes the panels used for creating and maintaining Library Codes through the Online system. The explanations for the fields in the Library Code processing apply to the fields on the Library Code Set-Up Form.

You can determine which CA-PanAPT Approvals should apply to the Library Code by determining what sort of members are processed by this Library Code. See your Approval Category Set-up Form for your list of Production member types, and the CA-PanAPT approvals that you have already determined applies to each type.

Retrieve Processing Setup Decisions

Before you can use the Retrieve Facility you must modify your Library Codes to contain the proper Retrieve Model Specifications. The following describes how to make those modifications for the CA-PanAPT supplied models:

- CA-Librarian
- CA Panexec
- CA Panvalet
- CA Panvalet to CA Telon
- CA-PFF
- CA Telon to CA Telon
- Partitioned Data Sets (PDS)
- PDS to CA-Telon.

CA Librarian

The following explains the CA-PanAPT model for CA-Librarian Retrieve setup. Change it to meet your needs.

Name

APJCLIBR

Description

Model APJCLIBR generates AFOLIBR JCL and control statements to copy a single member from one CA-Librarian master to another, using -OPT UTILITY. You control the JCL and control statement generation through the use of Modeling keywords.

Keyword Usage

$MEMBERCOUNT

Set to the current Retrieve request model execution count for the purpose of generating unique JCL field names such as STEP names. The character value ranges from 0000000 through 9999999 and is incremented by one for each member.

$CHKREP

Set to the same value (Y or N) as the Replace Member field of the Retrieve Processing Options panel. A value of Y indicates that the transferred CA-Librarian member can replace an existing member of the same name ($TONAME) in the destination CA-Librarian master ($DEST1DSN).

$ORIGDSN

Set to the data set name that the member is being Retrieved from, as defined to the Library Code.

$DEST1DSN

Set to the destination data set name the member is being Retrieved to, as defined in the Library Code.

$FROMNAME

Set to the same value as the From Member field of the Retrieve Processing Options panel. This is the member name on the $ORIGDSN master that is copied to the $DEST1DSN master.

$TONAME

Set to the same value as the To Member field of the Retrieve Processing Options panel. This is the name the member is copied to on the $DEST1DSN master.

CA Panexec

The following explains the CA-PanAPT model for CA-Panexec Retrieve setup. Change it to meet your needs.

Name

APJCPEX

Description

Model APJCPEX generates CA-Panexec JCL and control statements to copy a single member from a source CA-Panexec library to a target CA-Panexec library. You control the JCL and control-statement generation through the use of Modeling keywords.

Keyword Usage

SELGRP

Set to the default CA-Panexec group name associated with this Library Code. The SELGRP value is set by a Keyword Assignment Statement in the Retrieve Model specification of the Library Code. The $FROMDATA value can override the SELGRP value. If the From Data field is empty, this variable must be set to a group.

Note: The next three variables (SELMODE, SELSTAT, and SELTYPE) must be set to valid values or modeling errors occur.

SELMODE

Set to the CA-Panexec Mode name associated with the from data set Library Code. The SELMODE value is set by a Keyword Assignment Statement in the Retrieve Model specification of the Library Code.

SELSTAT

Set to the CA-Panexec Status name associated with this Library Code. The SELSTAT value is set by a Keyword Assignment Statement in the Retrieve Model specification of the Library Code.

SELTYPE

Set the SELTYPE value to the CA-Panexec Type name associated with this Library Code. The SELTYPE value is set by a Keyword Assignment Statement in the Retrieve Model specification of the Library Code.

$MEMBERCOUNT

Set to the current Retrieve request model execution count for the purpose of generating unique JCL field names such as STEP names. The character value ranges from 0000000 through 9999999 and is incremented by one for each member.

$CHKREP

Set to the same value (Y or N) as the Replace Member field of the Retrieve Processing Options panel. A value of Y indicates that the copied CA-Panexec member can replace an existing member of the same name ($TONAME) in the TEST CA-Panexec Library ($DEST1DSN). To accomplish this with CA-Panexec, a User Keyword REP is set to the string ',R' when $CHKREP = Y. The REP keyword is specified in the CA Panexec Copy control statement to request member replacement. If member replacement is not allowed, the REP keyword is set to a null string.

$ORIGDSN

Set to the data set name that the member is being Retrieved from, as defined to the Library Code.

$DEST1DSN

Set to the target data set name.

$DEST1SEC

A composite of the CA-Panexec access and override values. If $DEST1SEC is specified, the first four characters are used for the access value and the next four for the override values.

$FROMNAME

Set to the same value as the From Member field of the Retrieve Processing Options panel. The $FROMNAME value is used in the CA-Panexec COPY control statements to request the source library member.

$FROMDATA

Set to the same value as the From Data field of the Retrieve Processing Options panel. If the $FROMDATA value is specified, the value is used for the CA-Panexec GROUP name. The $FROMDATA value overrides the SELGRP value.

$TONAME

Set the $TONAME value to the same value as the To Member field of the Retrieve Processing Options panel. You can modify the To Member field to rename the member. The $TONAME value is used in the CA-Panexec COPY control statement to request the named target library member after the copy.

CA Panvalet

The following explains the CA-PanAPT model for CA-Panvalet Retrieve setup. Change it to meet your needs.

Name

APJCPANV

Description

Model APJCPANV generates PAN#2 JCL and control statements to copy a single member from a source CA-Panvalet library to a target CA-Panvalet library using the ++TRANSFER direct method.

In all cases, the STATUS of the member in the TEST Library is taken from the RSTSTAT operand in the PVOPT macro.

You control the JCL and control-statement generation through the use of Modeling keywords. The distributed model does not support rename.

Keyword Usage

LOCK

Set to the same value (Y or N) as the LOCK keyword through the keyword assignment statement in the Retrieve model specifications in the associated Library Code. A Y value generates a PAN#1 STEP to ++LOCK the From Member in the From Library when the From Library is QA or Production. The LOCK keyword must always be set to N for releases of CA-Panvalet prior to 14.1.

$MEMBERCOUNT

Set to the current Retrieve request model execution count for the purpose of generating unique JCL field names such as STEP names. The character value ranges from 0000000 through 9999999 and is incremented by 1 for each member.

$CHKREP

Set to the same value (Y or N) as the Replace Member field of the Retrieve Processing Options panel. A value of Y indicates that the transferred CA-Panvalet member can replace an existing member of the same name ($TONAME) in the TEST CA-Panvalet Library ($DEST1DSN).

Note: Because the column 72 R option of the ++TRANSFER direct method causes the STATUS of the member in the destination library to be the same as the STATUS in the origin library, this method *cannot* be used. If the origin library member had a PROD STATUS, that would be carried to the destination library. Thus, the To Member must be deleted from the To Library so that the column 72 R option is *not* required.

To accomplish this with CA-Panvalet, complete the following steps:

1. CONDITIONALLY GENERATED: If the REPLACE (Retrieve) option is specified, you need From Member existence test. This step is used as a safeguard against deleting the requested member from the To Library when a From Member does not exist in the From Library. A STEP is generated to test whether the From Member exists in the From Library. If this step is not successful, the succeeding steps are not executed.

2. CONDITIONALLY GENERATED: If the REPLACE option is specified, a PAN#2 STEP is generated to ++DELETE the existing To Member from the TEST Library.
3. ALWAYS GENERATED: A PAN#2 STEP is always generated to ++TRANSFER the From Member in the From Library directly to the To Library using the From Member name.

$ORIGDSN

Set to the data set name that the member is being Retrieved from, as defined to the Library Code.

$DEST1DSN

Set to the target data set name.

$DEST1SEC

Set to the CA-Panvalet Library Control Code for the Test Library defined in the associated Library Code.

$FROMNAME

Set to the same value as the From Member field of the Retrieve Processing Options panel. The $FROMNAME value is used in the CA-Panvalet control statements to request the source library member.

$TONAME

Set to the same value as the To Member field of the Retrieve Processing Options panel. The $TONAME value is used in CA-Panvalet control statements to request the target library member after the copy. It must be the same as $FROMNAME.

CA Panvalet to CA Telon

The following explains the CA-PanAPT model for CA-Panvalet to CA-Telon Retrieve setup. Change it to meet your needs.

Name

APJCPV2T

Description

Model APJCPV2T invokes the CA-Telon Import Facility to move production source from CA-Panvalet to CA-Telon TDF.

You control the JCL and control-statement generation through the use of modeling keywords.

Keyword Usage

$MEMBERCOUNT

Set to the current count for the member name being executed in the model. This generates unique JCL field names such as step names. The character value ranges from 0000000 through 9999999 and is incremented by 1 for each change in member.

$ORIGDSN

Set to the data set name that the member is being Retrieved from, as defined to the Library Code.

$DEST1DSN

Set to the data set name of the TEST Library defined in the associated Library Code. The name associated in the Library Code is the CA-Telon procedure VSQUAL name, not the complete library name.

$FROMNAME

Set to the same value as the From Member field for the Retrieve Processing Options panel. The From Member field must be the same as the To Member field because this model does not support Rename.

Limitations and Restrictions

You must modify model APJCPD2T to point to various qualifiers and data sets. +++ MODIFY +++ indicates the lines that you must change.

You must modify the model to specify the correct JCL jobcard information in the INIT section.

You must modify the model to set values for the following user keywords. +++ MODIFY +++ indicates the lines that you must change.

TELONQUAL CA-Telon load library to be included in the STEPLIB allocation.

PANLOAD CA-Panvalet software library name.

This model does not let you change the member name, and the replace option is not used in this model. If the member names do not match, a message is issued.

For CA-Telon Release 2.0C or higher, this Retrieve model has the RUNTYPE set to I, which means to ignore any data administration data in the TDF. In this case, the import always works.

If necessary, set the RUNTYPE to C for compare processing or M for compare or merge processing. The C and M processes are determined by the variable MAXSEVR, which indicates the highest allowable severity code. Details of this process can be found in the *CA-Telon Programming Concepts Guide.*

Supplies statements that check that the final two characters on the member name match the DEFTYPE specified in the Library Code.

If the member name characters are different, uses the last two characters on the specified member name. This is done to ensure that the member existence exit checks for the correct name.

Model APJCPV2T invokes the CA-Telon procedure TLNUMPAN.

CA PFF

The following explains the CA-PanAPT model for Protection File Facility (CA-PFF) Retrieve setup. Change it to meet your needs.

Name

APJCPFF

Description

Model APJCPFF generates PFF, JCL, and control statements to copy a single member from a source PDS library to a target PDS library. You control the JCL and command generation through the use of Modeling keywords.

Keyword Usage

ALIAS

Set to the same value (Y, N, or NULL) as the ALIAS keyword. CA-PanAPT does this through the keyword assignment statement in the Retrieve Model specifications in the associated Library Code. The $FROMDATA value can be used to override the ALIAS value.

$MEMBERCOUNT

Set to the current Retrieve request model execution count for the purpose of generating unique JCL field names such as STEP names. The character value ranges from 0000000 through 9999999 and is incremented by 1 for each member.

$CHKREP

Set to the same value (Y or N) as the Replace Member field on the Retrieve Processing Options panel. A value of Y indicates that the copied member can replace an existing member of the same name ($TONAME) in the target library ($DEST1DSN).

$ORIGDSN

Set to the data set name that the member is being Retrieved from, as defined to the Library Code.

$DEST1DSN

Set to the target library data set name.

$FROMNAME

Set to the same value as the From Member field of the Retrieve Processing Options panel. The $FROMNAME value is used on the PFF SELECT command statement to request the source library member.

$FROMDATA

Set to the same value as the From Data field on the Retrieve Processing Options panel. If the $FROMDATA value is specified, the value is used to override the ALIAS value specified in the Retrieve Model specifications. See ALIAS for valid values to use in the From Data field.

$TONAME

Set to the same value as the To Member field of the Retrieve Processing Options panel. You can modify the To Member field to rename the source member. The $TONAME value is used on the PFF SELECT APPLY NEWNAME command statement to request the name of the member on the target library.

CA Telon to CA Telon

The following explains the CA-PanAPT model for CA-Telon Retrieve setup. Change it to meet your needs.

Name

APJCT2TD

Description

Model APJCT2TD invokes the CA-Telon Export and Import Facilities to move production source from a CA-Telon TDF to a CA-Telon TDF.

You control the JCL and control-statement generation through the use of modeling keywords.

Keyword Usage

$MEMBERCOUNT

Set to the current count for the member name being executed in the model. This generates unique JCL field names such as step names. The character value ranges from 0000000 through 9999999 and is incremented by one for each change in member.

$ORIGDSN

Set to the data set name that the member is being Retrieved from, as defined to the Library Code.

$DEST1DSN

Set to the data set name of the TEST Library defined in the associated Library Code. The name associated in the Library Code is the CA-Telon procedure VSQUAL name, not the complete library name.

$FROMNAME

Set to the same value as the From Member field for the Retrieve Processing Options panel. The From Member field must be the same as the To Member field because this model does not support Rename.

$DIBSAPPL

The $DIBSAPPL field includes positions 1, 2, and 3, and it is used to specify the ENV Environment, FORMAT, and PSB parameters to the CA-Telon TLNUXDEF procedure. ENV is in position 1, FORMAT is in position 2, and PSB is in position 3. Valid values are:

ENV

CA-Telon program environment. Use T for TSO, I for IMS, C for CICS, or B for batch.

FORMAT

Type of screen formatting required. Use **M** for IMSMFS, **B** for CICSBMS, or **N** for CA-Telon generated (for CICS only).

PSB

Type of PSB to be generated (I=IMSPSB, D=DLIPSB, or N=NONE).

Limitations and Restrictions

You must modify model APJCT2TD to point to your source library PDS (for example, your CA-Telon Export Library). +++ MODIFY +++ indicates the lines that you must modify.

You must modify the model to specify the correct JCL jobcard information in the INIT section.

You must modify the model to set values for the following user keywords. +++ MODIFY +++ indicates the lines that you must change.

SRCLIB Temporary export and import PDS.

TELONQUAL CA-Telon load library to be included in the STEPLIB allocation.

This model does not let you change the member name, and the replace option is not used in this model.

If the member names do not match, a message is issued.

For CA-Telon Release 2.0C or higher, the RUNTYPE set to I in this Retrieve model, which means to ignore any data administration data in the TDF. In this case, the import always works.

If necessary, set the RUNTYPE to C for compare processing or M for compare or merge processing. The C and M processes are determined by the variable MAXSEVR, which indicates the highest allowable severity code. Details of this process can be found in the *CA-Telon Programming Concepts Guide*.

Supplies statements that check that the final two characters on the member name match the DEFTYPE specified in the Library Code.

If the member name characters are different, uses the last two characters on the specified member name. This is done to make sure that the member existence exit checks for the correct name.

This model invokes the CA-Telon procedures TLNUMDEF and TLNUXDEF.

Partitioned Data Sets

The following explains the CA-PanAPT model for Partitioned Data Sets (PDS) Retrieve setup. Change it to meet your needs.

Name

APJCPDS

Description

Model APJCPDS generates IEBCOPY JCL and control statements to copy a single member from a source PDS library to a target PDS library. You control the JCL and control-statement generation through the use of Modeling keywords.

Keyword Usage

$MEMBERCOUNT

Set to the current Retrieve request model execution count for the purpose of generating unique JCL field names such as STEP names. The character value ranges from 0000000 through 9999999 and is incremented by 1 for each member.

$CHKREP

Set to the same value (Y or N) as the Replace Member field of the Retrieve Processing Options panel. A value of Y indicates that the copied member can replace an existing member of the same name ($TONAME) in the Test Library ($DEST1DSN). Accomplish this with IEBCOPY as follows:

```
if $CHKREP=N the IEBCOPY COPY statement generated is:
"...INDD=FROMLIB"
if $CHKREP=Y the IEBCOPY COPY statement generated is:
"...INDD=((FROMLIB,R))".
```

$ORIGDSN

Set to the data set name that the member is being Retrieved from, as defined to the Library Code.

$DEST1DSN

Set to the data set name of the Test Library defined in the associated Library Code.

$FROMNAME

Set to the same value as the From Member field of the Retrieve Processing Options panel. The $FROMNAME value is used in the IEBCOPY SELECT control statement to request the source library member.

$TONAME

Set to the same value as the To Member field of the Retrieve Processing Options panel. You can modify the To Member field to rename the member. The $TONAME value is used in the IEBCOPY SELECT control statement to request the name target library member after the copy.

$DEST1SEC

The destination data set's Security field for this Library Code for this move.

PDS to CA Telon

The following explains the CA-PanAPT model for PDS to CA-Telon Retrieve setup. Change it to meet your needs.

Name

APJCPD2T

Description

Model APJCPD2T invokes the CA-Telon Import Facility to move production source from a PDS library to CA-Telon TDF. This model invokes the CA-Telon procedure TLNUMDEF.

You control the JCL and control-statement generation through the use of modeling keywords.

Keyword Usage

$MEMBERCOUNT

Set to the current count for the member name being executed in the model. This generates unique JCL field names such as step names. The character value ranges from 0000000 through 9999999 and is incremented by one for each change in the member.

$ORIGDSN

Set to the data set name that the member is being Retrieved from, as defined to the Library Code.

$DEST1DSN

Set to the data set name of the TEST Library defined in the associated Library Code. The name associated in the Library Code is the CA-Telon procedure VSQUAL name, not the complete library name.

$FROMNAME

Set to the same value as the From Member field for the Retrieve Processing Options panel. The From Member field must be the same as the To Member field because this model does not support Rename.

Limitations and Restrictions

This model does not let you change the member name, and the replace option is not used in this model.

If the member names do not match, a message is issued.

For CA-Telon Release 2.0C or higher, this Retrieve model has the RUNTYPE set to I, which means to ignore any data administration data in the TDF. In this case the import always works.

If necessary, the RUNTYPE can be set to C for compare processing or M for compare or merge processing. The C and M processes are determined by the variable MAXSEVR, which indicates the highest allowable severity code. Details of this process can be found in the *CA-Telon Programming Concepts Guide*.

Statements are supplied to check that the final two characters on the member name, match the DEFTYPE specified in the Library Code.

If the member name characters are different, the last two characters on the specified member name are used. This is done to make sure that the member existence exit checks for the correct name.

Library Codes and Inventory File Interrelation

Relating Library Codes and the Inventory file as follows:

1. When you define a Library Code (see Library Code Setup Form in Appendix A), you can specify Inventory is enabled for this Library Code. You would expect that Inventory Records exist for all Move Request Members that use this Library Code.

   You can specify that if CA-PanAPT does not find an Inventory Record for a member being added to a Move Request, it will automatically prompt the user to create one and provide default values on the creation panel (Automatic Create).

   If you specify that Inventory is enabled, the Inventory Record does not exist, and Automatic Create is not enabled, then each member involved is identified by a flag in the status column of the Member Moves panel. Members that require an Inventory Record or an approved Inventory Record have an I in the status column. Members that require assignment have an A in the status column.

2. You can indicate that a member must be properly assigned.

   Move Requests that contain the member cannot be closed until the member is properly assigned. Your site defines proper assignment through values selected for the Close Assignment option (Control File Maintenance; System Information) and the MOVEREQ/CLOASSGN activity (Control File Maintenance Activity).

   Alternately, you can indicate that if the member is not already assigned to that user (and not assigned to anyone else), CA-PanAPT will do an Automatic Assignment to the user when the member is added to the Move Request.

   When a member is assigned to a user, no other user can assign or Retrieve it. This ensures that duplicate moves are not done without knowledge of the assigned user.

   Note: Assignment does not stop other users from adding a member to more than one Move Request.

3. You can indicate whether CA-PanAPT automatically Releases the Inventory Assignment when an assigned member is finally moved to Production or deleted from a Move Request or when a Move Request is deleted.

If a member to be released exists on any other Move Request in an active status, the assigned-to Move Request is changed to the Move Request containing the member with the earliest Move Date. The member can be reassigned to the owner of this other Move Request depending on the value of the Reassign/Transfer flag on the Control file.

4. You can specify that all members of the Library Code must have Approved Inventory Records before any Move Request containing them can be closed. CA-PanAPT flags any members of the Library Code that do not have Approved Inventory Records.

   You must approve each flagged Inventory Record before the Move Request can be closed and processed. This helps ensure that the Inventory Records contain data acceptable to the models that will use the records during daily move processing. If a model detects bad data, it might signal an error that terminates processing for all Move Requests being processed in that Job.

   An inventory edit exit can assist in ensuring correct inventory data. The exit is invoked for Inventory maintenance activities. It ensures that appropriate data is entered when a Move Request is added or changed; this decreases the need for manual inspection.

5. You can specify whether newly-added Inventory Records for the Library Code are marked Approved or Unapproved. You might want to mark them Unapproved to require manual inspection later. If the Library Code models do not have stringent data requirements, you can choose to have CA-PanAPT mark each new record Approved.

6. You can specify whether members can be retrieved. You can only retrieve members after they have been assigned to you. Retrieve can be initiated as part of the assignment request or, later, as part of a specific Retrieve request.

   To Retrieve members, their Library Code definition must have Inventory, Assignment, and Retrieve enabled.

7. You can specify an Inventory Qualifier that is different from the default. The default is the Library Code.

   CA-PanAPT treats members with the same name but different Inventory Qualifiers as different members. For example, a source module and a load module could have the same member name, be resident in different libraries and be treated differently.

When Library Codes share the same Production Library, however, their members are not different. These Library Codes must specify the same Inventory Qualifier so that CA-PanAPT recognizes that their members are the same for purposes of assignment and release.

Chapter 2: Implementation

This section contains the following topics:

About the Implementation

This chapter describes CA-PanAPT implementation. System implementation is done in three phases:

- Phase One: System Familiarization
- Phase Two: Pilot Project
- Phase Three: Ongoing Implementation

For CA-PanAPT, we suggest the following scenarios for the phases:

Phase One: System Familiarization

Goal: To exercise all of the CA-PanAPT facilities so users can become familiar with the system.

Action: Exercise the entire CA-PanAPT Production Turnover scenario using the supplied and unchanged CA-PanAPT PDS model. Test the system with test libraries, members, and a limited number of users.

Phase Two: Pilot Project

Goal: To implement the first live user Library Code and live Move Request so users can understand and use models.

Action: Implement one Library Code using one set of actual user libraries. Use supplied CA-PanAPT models, copy and modify slightly as necessary.

Phase Three: Ongoing Implementation

Goal: To continue the implementation process by repeating Phase Two for each additional set of libraries. To become independent and knowledgeable in system usage.

Action: Continue live implementation, one Library Code at a time, using actual user libraries. Use supplied CA-PanAPT models, copy and modify as necessary.

Each of these three phases is a separate subject in this chapter.

Phase One: System Familiarization

Phase One of system Implementation is normally done immediately after the physical installation and is part of ensuring that the installation is complete and correct.

Description

Use the following steps to familiarize yourself with the CA-PanAPT system:

1. Use the supplied PDS model.
2. Limit the number of users to from one to four.
3. Set up one CA-PanAPT Library Code.
4. Set up two CA-PanAPT Move Requests.
5. Complete a full CA-PanAPT cycle, including all reporting.

The expected completion time is from 1 to 16 hours.

User Preparation and Planning

Use the following steps to prepare and plan:

1. Perform *one* of the following:
   - Select five current PDS libraries for test use, one from each category (TEST, QA, PROD, PROD Backup, and PROD Backout), and specify members for testing
   - Create five new PDS libraries for test use. Add several members for testing.
2. Determine which members in each Move Request to use for testing.
3. Review the chapter Planning the Implementation complete the forms in Appendix A," and review the following chapters in the *CA-PanAPT Reference Guide*:
   - Library Codes
   - Control Files
   - Setups for Different Types of Moves
   - Move Requests
4. Review the Library Code Set-up Form in Appendix A of this guide.

Preparation Steps

Use the following steps to prepare for Phase One Implementation:

1. User IDs:

   Determine user roles in CA-PanAPT implementation. Use one user ID for each different role, so you can see the effects of security authorization.

2. Library Code:

   Fill out the three-page Library Code Set-up Form for your Phase One Library Code.

3. Determine any changes required to CA-PanAPT batch:

   Review the "Setups for Different Types of Moves" chapter in the *CA-PanAPT Reference Guide*. Make appropriate changes to PROCLIB or PARMLIB as described.

4. Plan Move Requests:

   Set up two Move Requests in this phase. Note the member names for each Move Request. Who do you want to add the Move Requests? Who do you want to approve them? One good plan is to have two users, each adding one Move Request and then each approving the other's Move Request.

Performing Phase One Using CA PanAPT

CA PanAPT

Continue Phase One by following these steps:

1. The designated administrator uses CA-PanAPT to add users. For this phase, keep the default activity authorities.

   Note: After adding at least one user as system administrator, change the *DEFAULT user ID so that it no longer makes everyone an administrator.

2. The designated administrator modifies the System Information records.
3. Run Job APJJ5103 in JCLLIB to print your updated Control file records. Review these to be sure you understand the report and what you did.
4. The designated administrator adds the Library Code, using the Library Code Set-up Form as input.
5. Run Job APJJ5102 in JCLLIB to print the Library Code file. Review this report to be sure you understand the report contents.
6. Create your two planned Move Requests using CA-PanAPT.
7. Retrieve the members where Retrieve is enabled.

   Note: Steps 10 and 11 can be performed in either order.

8. Review your Move Requests, change as needed. Print the Move Requests.
9. Close the Move Requests. Print the Move Requests.
10. Approve the Move Requests.
11. Run Job APJJ5310 in JCLLIB to select the Move Requests.
12. Run Job APJJ5320 in JCLLIB to move the members in the selected requests.
13. Review all reports, submit and review any others you feel are appropriate and informative.
14. Back up the CA-PanAPT files using the Job APJJBKUP in JCLLIB.

Phase Two: Pilot Project

The pilot project has two purposes:

- Ensures that the system operates at your site and meets your stated needs.
- Lets the primary user, through *live* usage, learn and understand the system, and more importantly, to develop internal procedures for other users.

This is probably the most important phase of the three phases of implementation.

Description

The pilot project consists of the following steps:

1. Select one set of libraries you want to place under the control of CA-PanAPT.
2. Select the move processing related to your library type. (For example, PDS, CA-Librarian, CA-Panvalet, CA-Panexec.)
3. Review the CA-PanAPT Modeling Facility. Review in detail your selected model. Copy and modify it as necessary.
4. Limit the number of users to one, two, three, or four.
5. Set up one CA-PanAPT Library Code for your *live* libraries.
6. Set up CA-PanAPT Move Requests to move members into Production.
7. Complete a full CA-PanAPT cycle, including all reporting.

The expected completion time for set-up is: one to eight hours; cycle completion depends on your libraries, possibly one to seven days.

User Preparation and Planning

Use the following steps to prepare and plan:

1. Select the set of libraries to be controlled. Designate which is to be a TEST, QA, PROD, PROD Backup, or PROD Backout library.
2. Determine the names of members to be used for the initial Move Request.
3. Review the Modeling Facility later in this guide.

Preparation Steps

Use the following steps to prepare for Phase Two implementation:

1. User IDs:

   Determine user roles in CA-PanAPT implementation.

2. Library Code:

   Fill out the pages of the Library Code Set-up Form for your Phase Two Library Code.

3. Set up model:

   Copy and change the sample provided if necessary, review the Modeling Facility chapter later in this guide.

   Review your selected/changed model to be sure you understand its functionality.

4. Set up a Retrieve model if your Library Code has Retrieve enabled.
5. Determine any changes required to CA-PanAPT Batch:

   If necessary, see the "Setups for Different Types of Moves" chapter in the *CA-PanAPT Reference Guide.* Make appropriate changes to PROCLIB or PARMLIB as described.

Performing Phase Two Using CA PanAPT

Continue Phase Two by following these steps:

1. Add your new Library Code.
2. Run Job APJJ5102 in JCLLIB to print the Library Code file. Review this report to be sure you understand the report contents.
3. Create your Move Request for your new Library Code.
4. Retrieve the members for which Retrieve is enabled.
5. Repeat Steps 12-17 from Phase One. Because this is, in essence, your first "live" CA-PanAPT Move, you might want to complete the steps over an extended period of days instead of all in one test session.

Phase Three: Ongoing Implementation

Now that you have learned to set up CA-PanAPT, it is time to complete your implementation. In this phase you use what you learned to set up the pilot project to set up your remaining libraries. This is an ongoing task, as more and more libraries and applications are ready to be managed by CA-PanAPT.

Ongoing Implementation Overview

Ongoing implementation consists of the following steps:

1. Repeat Phase Two steps for each set of libraries you place under the control of CA-PanAPT.
2. Establish any necessary protocol or external procedures.
3. Open utilization of CA-PanAPT to other users, phasing them in along with libraries.

The expected setup completion time of each set of libraries is one to eight hours; cycle completion depends on your environment.

User Preparation and Planning

Use the following steps to prepare and plan:

1. Review the results of the Phase Two implementation.
2. Determine the groups to be used in your CA-PanAPT system.
3. Plan CA-PanAPT security and update Activity records to reflect your needs.
4. Review your Production files and the establish order of implementation for each set of libraries.
5. Establish protocol and schedule of the utilization of CA-PanAPT in accordance with your current needs or procedures.
6. Document your CA-PanAPT procedures as you would any other data center procedure.
7. Schedule your internal training for other users of CA-PanAPT to demonstrate the system, and to alert them to new procedures.

Phase Three Steps

Phase Three is an ongoing phase. It consists of three steps:

1. Plan and implement a new Library Code.

   Note: Repeat Phase Two steps for each set of libraries.

2. Train new users in CA-PanAPT system usage.

   This must include a review of current procedures outside CA-PanAPT to determine if usage of CA-PanAPT affects them in any way Repeat from Step 1.

Chapter 3: Problem Reporting Procedures

This chapter describes procedures to assist you in working with CA Technical Support to identify, document, report and resolve errors that you find in CA PanAPT.

This section contains the following topics:

Identifying an Error

The indication that a software product is not performing as expected might be as obvious as an ABEND, as minor as a misspelling in the documentation, or as subtle as a warning message that is not always displayed when it should be.

The first step of problem reporting is to determine whether a perceived abnormal condition is a product error, an operating system error, or a matter of user perception. If you think that the product is working contrary to the documentation, find the area in the guide that is inconsistent with the actual performance of the product. If the guide agrees with the operation of the product, this is generally not considered an error. If the guide is vague or inconsistent, this could be an error in the documentation.

Client Service Process

When you believe that you have encountered an error that you cannot resolve yourself, go to http://ca.com/support. Before contacting us, review the topics in this section so you can gather information to help us work through your issue.

Possible Questions Asked

The following questions will probably be asked by a Technical Support Representative. Have the answers handy to speed up your error resolution.

- What operating system, version, and PUT level are you using?
- What version and PUT level of TSO/ISPF are you using?
- Are you using the IBM LE/390 COBOL runtime libraries? What version of COBOL are you using? IBM VS COBOL and COBOL II runtime libraries cannot be used.
- What is your company's experience level with CA-PanAPT?
- What is your experience level with the function that has the apparent error? What specific occurrences indicate the problem (ABEND, error message, unexpected results)?
- Is the problem consistent or did it happen just once?
- Does the problem occur with more than one member, Library Code and Move Request? If so, what is common among them?
- Has anything recently changed in your environment? Examples of changes are installation or upgrade of other vendor software, upgrade of IBM software (especially TSO/ISPF and LE/390/COBOL) new hardware, etc.
- Which exit points are you using in CA-PanAPT?
- What PTFs (Program Temporary Fixes) have been applied?
- Exactly what steps were taken to arrive at the problem. Be very detailed.
- What is the exact text of any error messages?
- What changes have you made to panels, CLISTs, skeletons, JCL members, cataloged procedures, Models, and PARM members?
- Has the problem occurred only since the application of CA-PanAPT maintenance?
- What CA-PanAPT authority does the user who is experiencing the problem have?

- What is the severity code? This is a number (from one to four) that you assign to the problem. Use the following to determine the severity of the problem:
  - 1—A system down or inoperative condition
  - 2—A suspected high-impact condition associated with the product
  - 3—A question about product performance or an intermittent low-impact condition associated with the product
  - 4—A question concerning general product utilization or implementation

Possible Documentation Needed

The following is a list of documentation that might be requested by a Technical Support Representative:

- Any dumps that are taken.
  - SYSUDUMP—Usually this is sufficient. Send a printed copy. Be sure to include *all* subtasks.
- Any tapes that are sent must be accompanied by the JCL that created the tape as well as a list of the data sets on the tape.
  - SYSABEND—Only if requested. A tape can be used. Be sure to include all subtasks.
  - SYSMDUMP—Only if requested. This *must* be on tape and unformatted.
- Screen prints of all screens from initial entry of CA-PanAPT until the error occurs.
- The TSO LOGON procedure CLIST used to allocate the CA-PanAPT software libraries (hardcopy).
- All installation output (hardcopy).
- Copies of any modules with site-specific modifications, with the modifications highlighted. Include any altered ISPF panels.
- Copies of any batch output showing the error.
- Library Code, Control, Inventory and partial Pending file reports produced by jobs APJJ5111, APJJ5102, APJJ5103, APJJ5104, APJJ5105, and AJJJ6111.
- Severe problems might require an IDCAMS PRINT of the VSAM files in hex and character.

Saving Screen Prints and ISPF Logfile

The following is the procedure to save the screen prints of all screens from initial CA-PanAPT entry to where the problem occurred.

1. Fill in all appropriate information on the CA-PanAPT panel.
2. Type **PRINT** on the ISPF Command line and press ENTER. This writes the filled-in CA-PanAPT screen to your ISPF logfile.
3. Print your ISPF log and add it to your other documentation.

Calling Technical Support

For further technical assistance with this product, go to http://ca.com/support for a complete list of CA locations and phone numbers. Technical support is available 24 hours a day, seven days a week.

Problem Determination Reports

The following is a list of the reports that might be required from you in order to completely resolve your software problem or to assist in identifying the cause of the problem. For sample listings of these reports, see Appendix A in the *CA-PanAPT Reference Guide*.

- Release Exception Report (APCS5310-02)
- Member Exception Report (APCS5310-03)
- Duplicate Exception Report (APCS5310-04)
- Model Processing Errors (APCS5320-02)
- Move Request Status Changes (APCS5391-01)
- Move Processing Exceptions (APCS5395-02)
- Move Request Status Changes (APCS5395-03)

Chapter 4: Modeling Facility

This section contains the following topics:

Modeling Facility Description

The CA-PanAPT Modeling Facility lets you create customized JCL and utility control statements for each type of batch move that needs to be done. The actual model might contain CA-PanAPT commands, operating system JCL, and utility control statements. The CA-PanAPT model file is a standard PDS.

The Modeling Facility uses CA-PanAPT models to generate the JCL and/or control statements that are used to move members from one library to another.

A model is a series of 80-character statements. The statements are either CA-PanAPT Control Statements or Data Statements. CA-PanAPT Control Statements use simple logic and keywords to control the generation of the output statements. The Data Statements are either the exact output lines to be generated or a skeleton that is filled in by CA-PanAPT and then generated as output.

CA-PanAPT Model Control statements generate output statements. As part of the batch processing of Move Requests, CA-PanAPT reads a model, accesses fields from CA-PanAPT files, and substitutes values from these fields and other user-defined variable values into the model.

The base CA-PanAPT product provides sample models for moving CA-Librarian, CA-Panvalet, CA-Panexec, or PDS members. If you need a customized model, you can either copy one of the provided samples and make the necessary changes, or write your own model. Models are stored on the PDS APTMODEL. They can be set up, copied, and modified by any editor that can edit standard PDS members. The CA-PanAPT Model Exchange file contains a wide variety of special-purpose models.

Specific requirements for move models supplied with CA-PanAPT are discussed in the "Setups for Different Types of Moves" chapter in the *CA-PanAPT Reference Guide*. Retrieve models supplied with CA-PanAPT are described in the "Retrieve Processing" chapter of the *CA-PanAPT Reference Guide*.

Models can be shared between Library Codes that use the same type of library. A Library Code can invoke several models.

Model Processing

The components of CA-PanAPT that perform model processing are Move Modeling, On-Demand Modeling, Verification model processing and Retrieve processing. Members to be processed are presented in order of groups.

There are three phases of processing for Retrieve model processing:

- Before processing a group, the models are processed in an INIT phase.
- For each member of the group, the models are processed in a MOVE phase.
- Upon completion of the group, the models are processed again in a TERM phase.

Move, Verification, and On-Demand model processing have two additional phases:

- A START phase precedes the INIT phase before all groups have been processed.
- An END phase follows the TERM phase after all groups have been processed.

The order in which members are grouped is:

Move Modeling (APJJ5320)

Order

Move directions (Backouts before Moves), move level (by relative position, not name), Model Base (unique for each Library Code), and Member name.

Group

Members with the same move direction, move level, and Library Code are grouped together.

On Demand

Order

The order can be specified by user, but should be sorted in Library Code or Model Base order to prevent erroneous INIT and TERM phases.

Group

Members with the same Library Code are grouped together.

Verification

Order

Library Code and Member name.

Group

Members with the same Library Code are grouped together.

Retrieve

Order

Model Base and Member name.

Group

There is no grouping. Each member is processed by all three phases.

In addition, Retrieve and Verification model processing copy leading job statements (specified on a panel during the Retrieve/Verification processing) to the submit file, which is the destination of data produced by the models. After the models have been invoked for the last TERM phase (Retrieve) or for the END phase (Verification), the JCL is automatically submitted to the system for execution. Verification model processing can be invoked multiple times in one online action if multiple Verification categories are being processed, producing a job for each category.

Modeling and the Parts of CA-PanAPT

Modeling relates processing for a particular type of move to:

- Models and Model Statements
- Members of APTMDLO
- Move Request, Library Code, and Inventory Data
- JCL for APJJ5320, the Standard Move JOB
- Move and External Processing jobs invoked from APJJ5320
- Retrieve Processing
- Verification Processing
- On Demand Processing

Models and Model Statements

Models are stored as members of a PDS. CA-PanAPT models consist of source statements that direct CA-PanAPT to write 80-character images to one or more outputs. These outputs can be sequential files or members of the CA-PanAPT APTMDLO PDS. Outputs can be written for any of the following reasons:

- To supply information for posting Move Request status.
- To create JCL and control statements to move or back out members.

  Typically, the model creates one set of members per grouping. One of these members contains JCL. Other members contain control statements. The JCL generated has DD statements referencing the control statement members.

  The Model facility provides a unique prefix per grouping that can be used to prevent different groupings from accidentally using the same members.
- To create JCL for external processing, such as compiles and link edits. The sample model provided for compiles creates one member with a single job. Each member to be compiled has its own steps in this job.

  You can use a different setup if you like, creating separate jobs for each compile, one submitting the next when it finishes.
- To create JCL for submission to the operating system as part of Retrieve processing. CA-PanAPT creates separate jobs for each Retrieve request so that they can be handled as they occur, rather than waiting for a scheduled batch job. One job is created that includes all members requested at the same time.

  CA-PanAPT submits Retrieve JCL to the operating system. You need not take any other action to start the Retrieve process. Retrieve models supplied with CA-PanAPT do not write to APTMDLO members or sequential files. All information required for the Retrieve is included with the JCL.
- To create JCL for submission to the operating system to perform automated verification of a Move Request.

Move (Turnover) vs. Retrieve Models

Move (turnover) processing controls moving members into the Production environment. Retrieve processing controls copying members to the test library so they can be changed. Move (turnover) processing is concerned with Move Requests, but Retrieve processing is not.

The models supplied with CA-PanAPT support either move (turnover) processing or Retrieve processing, but not both.

Note: The modeling process is performed online for Retrieve and Verification modeling. The modeling process is performed during batch for Move and On Demand modeling.

Move (Turnover) vs. On Demand Modeling

On Demand Modeling is very similar to Move Modeling, but it differs in the use of system keywords and the source of the model specifications. The system keywords have the same meanings as those described in Keywords later in this chapter. The $MODELTIME keyword is O for On Demand Modeling.

Model specifications are provided through an APTSYSIN file rather than a Library Code, and only one model specification can be provided per execution. Up to twelve model specification lines can be supplied, and data can be entered in positions 1 to 75 of these records.

On Demand Modeling is typically used to extract information from a Move Request for some reason other than movement. On demand modeling can be used to generate compile and link edit JCL for members residing on test libraries, utilizing inventory for compiler and link edit parameters.

Note: The modeling process is performed during batch execution for both Move and On Demand. CA PanAPT uses On Demand modeling in the DB2 option for Validating a Move Request.

Verification vs. Retrieve Procedure Modeling

Verification modeling is very similar to Retrieve modeling, but it differs in the use of system keywords and the source of the model specifications. The system keywords are described in Keywords later in this chapter. The $MODELTIME keyword is V for Verification procedures. Verification Model specifications are kept in the Control file rather than a Library Code. Up to three model specification lines are supplied in which data can be entered.

Note: The modeling process is performed online for both Retrieve and Verification procedure models.

Members of APTMDLO

APTMDLO, a standard PDS containing 80-character records, contains the output of CA-PanAPT modeling. Some members contain status posting information, some contain control statements, and some contain JCL.

Move Request, Library Code, and Inventory Data

CA-PanAPT supplies many items of information to the models through system keywords. (Refer to the complete list of all system keywords later in this chapter.) Some keyword information, like member name and user data, come from the Move Request. Some of it, like data set names, come from the Library Code definition.

If Inventory is enabled for the Library Code, more keyword information comes from the Inventory Record. If your model uses these keywords, Inventory must be enabled for each Library Code that uses it.

APJJ5320, the Standard Move Job

APJJ5320 is the job that initiates the moves selected by APJJ5310. It performs the move modeling by running program APCS5320. Then, before any of the generated move JCL is submitted, it runs the posting program (APCS5391) to initialize each Move Request to be processed. It then runs the posting program again to post the completion of any null moves.

In prior releases of CA PanAPT, the moves were performed next in the APJJ5320 job, by move JCL procedures that were included in the APJJ5320 job. Your site might have your own move procs that have been inserted into the APJJ5320 job.

Many of the models on the CA PanAPT Model Exchange have move procs that work along with them that must be placed into the APJJ5320 job. Finally, the generated move JCL is submitted for execution using the APCS5905 submit program. The submit program is used, instead of other job submission facilities to the internal reader, such as the TSO submit command and IEBGENER, because it has an include expansion facility that might be required by some models.

Generated Move Jobs

When the APJJ5320 job completes, it submits a Move job. The JCL for this job was generated by your models. Depending upon your setup and the types of moves being performed, you might have all of your moves in a single move job, or you might have a series of jobs, each submitting the next as their final step.

The move job or jobs migrate your members, but they typically do not perform any compiles, link edits, or other similar processing for the members being moved. Upon completion of each group of moves, the move job runs the posting program (APCS5391) to mark the members as being moved. If compiles, link edits, or other further processing is required for the members, the posting program notes this.

External Processing Jobs

After the last group of moves has finished, it is time to run compiles, link edits, and any other additional processing your members require. This is known as External Processing. If there is external processing to be performed, the job(s) to do this are submitted after the last group of moves is complete. External processing can be done in a single job, or multiple ones; your models control this. Your models also control the order of external processing, allowing for instance, compiles to be run first, then link edits, then DB2 binds.

As the external processing completes for a member, the external processing job runs the posting program (APCS5391) to clear the external processing required indicator.

APTMDLO Member Initialization

In prior releases of CA-PANAPT, it was common for models to only create control statements which were referenced by corresponding Move PROCs in the APJJ5320 job. Many models on the model exchange are set up this way. ABENDs could occur in these PROCs if they attempted to access these control statement members when they didn't exist, because there were no members to move in that move cycle for the Move PROC.

One way to prevent these ABENDs was to assure that there were always control statement members for the PROCs to access even if they were empty. This is accomplished using the APJRMDLO PARMLIB member. In this member you specify the names of APTMDLO members that are to be unconditionally created.

Two statement types can be specified in the APJRMDLO member, comments and member initialization statements. Comments are specified by placing an * in position one of the record. Member initialization statements are specified by placing the characters APTMDLO starting in position one followed by a member name starting in position 10, optionally followed by a comment in positions 18 through 80. Every member name specified in the APJRMDLO member exists in the APTMDLO data set when Move Modeling is complete, although they are empty if no models wrote anything to them. On-Demand Modeling can also use a member initialization member or file simply by allocating ddname M5320F01 to one.

The APJRMDLO member distributed with CA-PanAPT has sample and prototype statements for many of the models and Move Procs on the Model Exchange.

None of the models distributed with CA-PanAPT require the use of the APJRMDLO parmlib member, so the APJJ5320 job does not use it. If you have models that need it, you must alter the APJP5320 PROC, uncommenting the M5320F01 DD statement in step APCS5320.

Retrieve Processing

Retrieve models generate JCL to copy a member to the TEST library so it can be changed. Retrieve processing is generally simpler than move (turnover) processing because it does not have to deal with several library levels, Backup, Back Out, or Move Requests.

Verification Processing

CA PanAPT maintains up to 20 Verification Procedures categories. A Verification Procedure is a program or set of programs designed to analyze the attributes of a Move Request and post a pass or fail indication. Verification Procedures are very similar to the 20 Approval Categories except approval categories are approved or disapproved by authorized people where as Verification Procedures are approved/disapproved by a specific process.

Verification models generate JCL to run a verification procedure. The model can extract information from the Move Request, Library Codes and Inventory Records used by the Move Request to build this JCL.

The CA-PanAPT DB2 option uses verification modeling to extract information from a Move Request that is then used to verify that related DB2 components are being moved together.

On Demand Processing

On-demand models can be used to extract information from a Move Request or a group of Move Requests. The CA PanAPT DB2 option uses on-demand modeling to extract information from a Move Request which is then used to verify that related DB2 components are being moved together. An example of on-demand modeling can be found in job APJJ5530 and procedure APJP5530 of the CA PanAPT DB2 option. On-demand modeling can be used by client sites to provide information regarding specific Move Requests, for reporting, or a variety of other uses. No statuses are changed by On-demand modeling, and no movement is implied.

Model Control Statements

Control statements let you control the logic flow of your model. Model Control statements can be:

- Keyword Assignment Statements
- Logic Control Statements
- Comments

General Syntax

An @ (at-sign) in Column 1 indicates to the Model processor that the line contains one or more Model Control Statements. The @ is the default Control Statement delimiter. You can change this delimiter for any particular model by using the $DELIM keyword.

You can place multiple Model Control statements on a single line if they are separated by a ; (semi-colon). To make the statements easier for you to read, you can put spaces on either side of the semi-colon.

The value of a Model Control statement must be surrounded by *single quotes.* If the value itself must contain a quote, use two single quotes to distinguish the embedded quote from the value delimiters.

This subject covers each of the Model Control Statements.

Keyword Assignment Statements

A keyword represents the value of a data element. There are two types of keywords: system keywords and user keywords.

User keywords *must* be defined by a Keyword Assignment Statement before they can be used in Logic Control Statements or substituted into a Data Statement.

CA-PanAPT automatically initializes most system keywords. Most of them cannot be changed by a Keyword Assignment Statement. See System Keywords topic later in this chapter for those keywords that can be changed by assignment statements.

Keyword values can be tested by the Model Control Statements, and thus can be used to control logic in a model. Keyword values can be substituted into a data statement.

System Keyword Rules

System keywords start with a $ and are initialized by CA PanAPT. Many system keywords represent a field on a CA PanAPT file. Most system keywords cannot be changed in a Model. (See “System Keywords” later in this chapter for a complete list of keywords.)

User Keywords Rules

User keywords are defined in a Model or on the Model specifications.

For Move and Retrieve Modeling, the model specifications can be found in the Library code.

For On Demand Modeling, the model specifications are provided in the file APTSYSIN to the on-demand modeling program, APCS5920.

For Verification modeling, the model specifications are found in the Control file.

A User keyword is defined on the first Model statement where it is encountered. A User keyword must be initialized by a Keyword Assignment Statement before it can be used in any other Model statement.

No more than 1000 user keywords can be defined at a time.

User keywords have the following syntax rules:

- Keyword names can be from 1 to 20 characters long.
- Keyword names must not begin with a $, except for the special prefixes G and L.
- Keyword names must not contain embedded delimiter characters ($ < > ! , : =) or blanks.
- Keyword value *cannot* exceed 80 characters.
- Keyword names cannot be one of the following reserved words:
  - AND
  - ELSE
  - ENDIF
  - ENDWHILE
  - IF
  - INC
  - INCLUDE

- OR
- WHILE

Format

This statement has the following format:

```
@   keyword = 'value'
```

@

Model Statement delimiter in column 1, indicating that this is a Model Control statement, not a Data Statement.

keyword

The user keyword or a system keyword for which the value can be changed.

'value'

The value assigned to the given keyword. The value must be surrounded by single quotes. If the value itself must contain a quote, use two single quotes to distinguish the embedded quote from the value delimiters.

Examples

The following examples show you how to use the Keyword Assignment Statement. Examples 1 and 2 show user keywords as part of the Keyword Assignment Statement. Example 3 shows a system keyword as part of the Keyword Assignment Statement.

Example 1

```
@       OUTMSG = 'THIS LIBCODE CANNOT BE PROCESSED'
```

The string THIS LIBCODE CANNOT BE PROCESSED is assigned to the User Keyword OUTMSG.

Example 2

```
@       MYMSG = 'THIS LIBCODE CAN''T BE PROCESSED'
```

The string THIS LIBCODE CAN'T BE PROCESSED is assigned to the User Keyword MYMSG.

Example 3

```
@       $MSG = 'THIS WON'T BE PROCESSED'
```

This modeling statement is flagged as a syntax error when modeling is done.

Scope of User Keywords

User keywords have three different scopes, determined by their member names.

Standard User keywords are those with names that start with any character other than a "$" (dollar sign). These keywords have a scope local to the current phase. When the current phase ends, the keyword is automatically deleted. When multiple MOVE phases occur (for multiple members) between an INIT and a TERM phase it should be noted that these user keywords are deleted at the end of each MOVE phase, (that is, at the end of each member's processing).

When a user keyword is prefixed with the characters "L" its scope is local to the current INIT-MOVE-TERM group. See the topic Model Processing earlier in this chapter for information about groups. You can set these during the INIT or MOVE phase and continue to use them through the TERM phase. If these keywords are set during a phase not related to a group, such as the START phase, they are deleted at the end of that phase.

User keywords prefixed with the characters "G" are global throughout modeling. They are not deleted until modeling ends. If you create global user keywords recall that there can only be 1000 user keywords defined at a time and you have no means to delete global user keywords when you are done with them.

Logic Control Statements

Logic Control Statements provide IF/ELSE logic to the model, and thus control the generation of output records.

Syntax Rules

Every IF statement must have a corresponding ENDIF statement. IF statements can be nested up to 20 levels deep.

When present, ELSE statements must match logically with the corresponding IF statement and must precede the ENDIF statement for that IF.

Format of IF Statements

```
@  IF   keyword1   operator   'value'
```

or

```
@  IF   keyword1   operator   keyword2
```

Format of AND Statements

```
@  AND   keyword1   operator   'value'
```

or

```
@  AND   keyword1   operator   keyword2
```

Format of OR Statements

```
@   OR   keyword1   operator   'value'
```

or

```
@   OR   keyword1   operator   keyword2
```

Explanation of Format

@

Model Statement delimiter in column 1, indicating that this is a Model Control statement, not a Data Statement.

keyword1

Can be any user keyword or one of the system keywords.

operator

Can be any of the following:

Operator Symbol	Meaning
EQ or =	equal
NE or ¬	not equal
LT or <	less than

Operator Symbol	Meaning
LE or <=	less than or equal to
GT or >	greater than
GE or >=	greater than or equal to

keyword2

Can be any user keyword or system keyword.

'value'

The value to be tested against keyword1. The value must be surrounded by single quotes. If the value itself must contain a quote, use two single quotes to distinguish the embedded quote from the value delimiters.

Rules for AND OR Statements

AND OR statements must be immediately preceded by an IF, AND, or OR statement. No other statements can lie between an IF and an AND or OR statement.

When both AND and OR statements are specified, the AND conditions are evaluated before the OR conditions. For example:

```
@  IF A = B;  AND C = D;  OR E = F
```

is evaluated as

```
IF (A = B AND C = D) OR E = F
```

Format of ELSE Statement

```
@ ELSE
```

Format of ENDIF Statement

```
@ ENDIF
```

Logic Examples

Example 1

Example 1 shows a Logic Control Statement (1) directing the steps CA-PanAPT needs to execute based on a condition of ABCDE.

```
@  IF $MSG    = 'ABCDE'                (1)
 //STEP1 DD DISP=SHR,DSN=SYS1.LOADLIB   (2)
 @  ELSE                                (3)
 //STEP2 DD DISP=SHR,DSN=SYS2.LOADLIB   (4)
 @  ENDIF                               (5)
```

(1) This is a Model Control Statement because it has an @ in Column 1. It is also a Logic Control Statement that is testing the value of the system keyword $MSG because of the IF statement.

(2) This is a Data Statement (no @ in Column 1). This Data Statement is generated (written to the output file) as an output statement only if the IF statement in (1) is true. If the IF statement is false, this Data Statement is not included in the output.

(3) This is a Model Control Statement because it has an @ in Column 1. It is also a Logic Control Statement because of the ELSE statement. The ELSE corresponds to the preceding IF. The ELSE statement is always optional.

(4) This is a Data Statement (no @ in Column 1). This Data Statement is generated as an output statement only if the IF statement in (1) is false. If the IF statement in (1) is true, this Data Statement is not included in the output.

(5) This is a Model Control Statement because it has an @ in Column 1. It is also a Logic Control Statement because of the ENDIF statement. It is the ENDIF that corresponds to the preceding IF.

The ENDIF statement is required. If this model omitted the ENDIF statement, a Model Processing Error would occur.

Example 2

Example 2 shows how to set the value of a system keyword.

The following statement is invalid:

```
@ IF <$MSG,1,1> = 'A'
```

$MSG is a system keyword, but <$MSG,1,1> is **not.** Therefore, CA-PanAPT rejects the statement. You can express the test correctly with the following two statements:

```
@ MSG1 = '<$MSG,1,1>'
@ IF MSG1 = 'A'
```

Comments

The Comment Statement lets you specify a comment in the CA-PanAPT Model Statement.

Syntax Rules

The characters, @*, concatenated at any point in a Model Control Statement causes all remaining text on that line to be treated as a comment. Comments cannot be specified on a Model Data Statement because all data is written as part of the model output.

Examples

Example 1

Example 1 shows a comment statement in the middle of a Model Control Statement.

```
@   IF KEYWORD = 'ABC'          @*   THIS IS A COMMENT
```

Example 2

Example 2 shows a comment at the beginning of a statement.

```
@*  THIS WHOLE LINE IS A COMMENT
```

Example 3

Example 3 shows an attempt to specify a comment in the middle of a Model Data Statement.

```
This line is a data line       @*   including this comment
```

This entire line, including: "@* including this comment", is written as part of the model output.

Invoking Other Models

Your models can invoke other models. This is typically done when many models need to do the same thing in the same way. For instance, if you have models that need to generate JOB statements you can write a separate model to do this, and then have the other models invoke this model. Then if you later want to change what is in your job statements you need only change the one model. This is similar to having programs call subroutines to separate out common processing.

Models invoke other modules through the INCLUDE statement

Format of INCLUDE Statements

```
@   INCLUDE   model
```

or

```
@   INC       model
```

Model Invocation Rules

The model name is the 1 to 8-character PDS member name of the model. You can use keyword substitution when specifying the name, but this is usually not done.

Models cannot be nested more than 50 models deep.

Models can be recursively invoked. That is, a model can include itself. In some situations this might be useful. It is the responsibility of the model writer to prevent endless recursion. If you accidentally write a model that invokes itself forever, CA-PanAPT halts the model with an error once the model nesting level exceeds 20 models deep.

Example

The following example illustrates how a compile model uses common models for job statement generation:

```
@*
@* Compile a COBOL program
@*
@JOBNAME  = 'COBCOMP'   @* Name of JOB
@JOBCLASS = 'A'         @* Execution class of JOB
@INCLUDE    JOBCARDS    @* Build the JOB statements
//COMP  EXEC COBUCL,...
```

Model Data Statements

Model Data Statements are any statements in a model that do not begin with the model delimiter. The Model Processor uses Model Data Statements to generate output records. CA-PanAPT writes these output records to a sequential file or to members of PDSs. Use of substitution keywords on Data Statements can alter the positions where the model characters are moved in the output record.

Normally, one Data Statement in a model produces one record written to the output file.

More than one Data Statement can be used to generate a single output file record. This is accomplished by placing an exclamation point (!) as the last non-blank character in the Data Statement.

Examples

Example 1

Assume the value of the system keyword $LIBC is SRCE, and the value of the system keyword $LIBSUBC is C:

```
MODEL:   LIBRARY CODE IS <$LIBC>    !
MODEL:                       /<$LIBSUBC>

OUTPUT:  LIBRARY CODE IS SRCE/C
```

Note that even though the second model statement seems to overlap the first model statement, on the output there is no actual overlaying of data because the value of $LIBC is only four characters long.

Example 2

Characters from subsequent Data Statements must not overlay characters from previous Data Statements or a model error occurs:

```
Model: THIS IS AN ERROR     !
Model:         NO

OUTPUT: No output, model error
```

Positional Reset In a Data Statement

Use of substitution keywords on Data Statements can alter the positions where the model characters are moved in the output record.

Examples

Example 3

To reset the output alignment position to correspond to the model, an exclamation point (!) can be used.

```
MODEL:  <$TONAME>,<$MR> !LIBCODE = <$LIBCODE>

OUTPUT:     MPS100,000312        LIBCODE = SRCE
```

Example 4

To illustrate how the length of keyword values effect positioning on the modeling output. Without the !, positioning in the output record might change.

```
MODEL:   <$TONAME>,<$MR>   LIBCODE = <$LIBCODE>

OUTPUT1: APT100,000312   LIBCODE = SRCE
OUTPUT2: XX,000312     LIBCODE = SRCE
```

Note that the blanks in the model are preserved but the LIBCODE=portion of the output slides to the left because the names of the keywords (for example, <$TONAME>) are longer than the values they replace.

Delimiters

The Modeling Facility uses two types of model statement delimiters: Resettable and Non-resettable.

Resettable Model Statement Delimiters

The following are the delimiters that you can reset:

- Control Statement Delimiter @ (at sign)
- Keyword Substitution Start < (less than)
- Keyword Substitution Stop > (greater than)
- Keyword Substitution Positional ! (exclamation point)

The exclamation point delimiter (!) is used for concatenating multiple statements and for positional reset.

These delimiters can be reset at any point in a model.

Each time a model is processed, the delimiters are reset to their default values. If you reset the values of the delimiters, the new values are not available to other models that might be called later, nor are the new values available the next time the current model is processed.

You can set new values for the delimiters at the beginning of the model, outside all @IF tests, to make the new values available to all statements in the model. All subsequent control statements in the model use the new delimiter values. You need not reset them to their default values before exiting the model.

You can set new values for the delimiters temporarily in a model to allow the model to generate Data Statements containing the delimiters. In this case, you reset the keywords to new values, generate the Data Statements as required, and then reset the delimiters to their default values for processing the rest of the model.

Resetting model statement delimiters is done with an assignment statement for the system keyword $DELIM. The required format for resetting these values is shown:

```
@   $DELIM = 'wxyz'
```

w

New Control Statement Delimiter

x

New Keyword Substitution Start

y

New Keyword Substitution Stop

z

New Keyword Substitution Positional Delimiter.

You include values for all four resettable delimiters and these delimiters must be unique.

The following shows you how to reset the system Delimiters to default values:

```
w   $DELIM = RESET
```

w

Current Control Statement Delimiter

RESET

Required keyword value.

Note that while all Model Control Statements that follow the keyword Assignment for $DELIM must use the NEW Control Statement Delimiter, the $DELIM statement itself must use the old delimiter.

Example

```
@ $DELIM = '#<>!'       @* sets Control Statement delimiter to "#"
#* New delimiter
# $DELIM = RESET        #* resets Control Statement delimiter
@* Old delimiters
```

Non-Resettable Model Statement Delimiters

Do not reset the following four delimiters in a model:

- Keyword Substitution Operator Separator (comma) ,
- Control Statement Separator ; (semi-colon)
- Keyword Assignment Indicator = (equal to)
- Keyword Assignment Value Delimiter ' (single quote)

Examples

This subject shows examples of model statements along with explanations of these statements.

Example 1

```
MODEL:  @         MYWORD = 'ABCD'                (1)
MODEL:  @       YOURWORD = 'FGHIJ'               (2)
MODEL:  OUTPUT DATA STATEMENT IS <MYWORD> !      (3)
MODEL:                                <YOURWORD>  (4)

OUTPUT: OUTPUT DATA STATEMENT IS ABCDFGHIJ
```

- (1)—This is a Model Control Statement (@ in Column 1) assigning the value of ABCD to the user keyword MYWORD.
- (2)—This is another Model Control Statement (@ in Column 1) assigning the value of FGHIJ to the user keyword YOURWORD.
- (3 & 4)—This is a Data Statement. The value of MYWORD is substituted. The "!" indicates that the following Data Statement is combined with this one, and the combined statements are written as a single statement.

  The first Data Statement is processed and the value of the keyword is substituted. The second Data Statement is processed, and the value of keyword <YOURWORD> is substituted, replacing blank characters of the first Data Statement.

Note: CA-PanAPT reports an error when the second Data Statement attempts to overlay a non-blank character on the first Data Statement.

Example 2

```
MODEL:  @   MYWORD = 'ABCDE'              (1)
 MODEL:  @   YOURWORD = 'EFGHI'            (2)
 MODEL:  DATA STATEMENT <MYWORD>           (3)
 MODEL:  <MYWORD,1,3>,<YOURWORD>           (4)

OUTPUT: DATA STATEMENT ABCDE              (5)
 OUTPUT: ABC,EFGHI                         (6)
```

- (1)—This is a Model Control Statement (@ in Column 1). This is a Keyword Assignment Statement, and the value ABCDE is placed in the user Keyword MYWORD.
- (2)—This is another Model Control Statement (@ in Column 1). This is also a Keyword Assignment Statement, and the value EFGHI is placed in the user Keyword YOURWORD.
- (3)—This is a Data Statement (no @ in Column 1). This statement contains a substitution keyword. The value of the user keyword MYWORD is placed in the generated output. (See output in line 5.)
- (4)—This is another Data Statement (no @ in Column 1). This statement contains a substitution keyword. The value of the user keywords MYWORD and YOURWORD is placed in the generated output. Note that only part of the value of MYWORD is to be substituted in the output statement. (See output in line 6.)

Example 3

Desired Output

At the beginning of the members for this Library Code you want:

```
//COMPILES JOB (),'COMPILE INTO PROD',
 //            CLASS=A,
 //            MSGCLASS=X
//*
```

And for each member to be compiled you want:

```
//*
//PGM1     EXEC COBOL,
 //             SRCEMEM=PGM1,
 //             LOADDSN='PROD.LOADLIB',
 //             LOADMEM=PGM1,
 //             LNKPARM='RENT'
```

Model Used to Produce this Output:

```
@ IF $PHASE = 'INIT'
//COMPILES JOB (),'COMPILE INTO PROD',
 //             CLASS=A,
 //             MSGCLASS=X
//*
@ ENDIF
@***
@ IF $PHASE = 'MOVE'
//*
//<$TONAME> EXEC COBOL,
 //             SRCEMEM=<$FROMNAME>,
//             LOADDSN='<$TODSN>',
 //             LOADMEM=<$TONAME>,
//             LNKPARM='$DIBSLINKOPT'
@ ENDIF
```

Example 4

Desired Output

At the beginning of the members of this Library Code you want:

```
COPY INDD=((JCLT,R)),OUTDD=JCLP
```

And for each member to be compiled you want:

```
SELECT MEMBER=PGM1
```

Model Used to Produce this Output:

```
@ IF $PHASE = 'INIT'
  COPY INDD=((<$LIBCODE>T,R)),OUTDD=<$LIBCODE>P
@ ENDIF
@***
@ IF $PHASE = 'MOVE'
      SELECT MEMBER=<$TONAME>
@ ENDIF
```

Keyword Substitution

Keyword substitution can be used on statements in the Model to cause the value of the keyword to replace the keyword itself. The keyword substitution symbol < begins the substitution keyword and the keyword substitution symbol > ends the substitution keyword. (See also Delimiters in this chapter.)

Note: Users can change these symbols (< and >) using the $DELIM system keyword.

Format

The keyword substitution format is:

```
<keyword,position,length>
```

Keyword

Any system keyword or user keyword.

Position

The beginning position of the keyword to be moved (default is 1). It is required when "length" is specified.

Length

The number of characters to be moved. If length is specified, position must also be specified. If omitted, the entire value of the keyword, beginning with the given starting position, is moved.

No blanks are allowed between the brackets for keyword substitution.

Placement in the Output Record

Use of keyword substitution causes the replacement text to be placed in the output record at the next available position.

The substitution of keyword values into Data Statements can result in some shifting of the output. If a keyword value is to be located in a specific column of the output record, an exclamation point (!) at one end of the keyword, in place of the bracket, causes the value to be **anchored** to that position.

Example

Given Model Control Statements:

```
MODEL:  @   MYWORD = 'ABCDE'              (1)
MODEL:  @   YOURWORD = 'EFGHI'            (2)
```

Review the outputs of the keyword substitution:

```
MODEL:  <MYWORD,2,3>,<YOURWORD>

OUTPUT: BCD,EFGHI

MODEL:  <MYWORD,2,3>,<YOURWORD!

OUTPUT: BCD,              EFGHI

MODEL:  <MYWORD,2,3>,!YOURWORD>

OUTPUT: BCD,         EFGHI
```

Substituting Keywords in Control Statements

Keyword Assignment Statements can also contain embedded keyword substitution.

The Model Processor substitutes the value of the keywords according to substitution syntax rules before processing the model line.

Examples

Example 1

Given Model Control Statements:

```
@   MYWORD  = 'ABCDE'   ;  NEWWORD = '<MYWORD>1'
@   MYWORD  = 'EFGHI'   ;  NEWWORD = '<MYWORD>1'
```

The model is processed as if the following statements had been entered:

```
@   MYWORD  = 'ABCDE'   ;  NEWWORD = 'ABCDE1'
@   MYWORD  = 'EFGHI'   ;  NEWWORD = 'EFGHI1'
```

Example 2

Given Model Control Statements and $LIBCODE=SRCE:

```
@   $OUTMEM = '<$LIBCODE>B'
```

Is processed as if it read:

```
@   $OUTMEM = 'SRCEB'
```

And for $LIBCODE=LOAD as if it read:

```
@   $OUTMEM = 'LOADB'
```

Example 3

You want to generate a name based on characters three through six of the member name supplied in $FROMNAME. The new name must be no more than eight characters, but $FROMNAME can be any length from five to eight characters.

For example, if the value of $FROMNAME is 'ABC123', you want to generate a value of 'XC123YZ'.

The following modeling statement assigns the needed value to USERVAR.

```
@USERVAR = 'X<$FROMNAME,3,4>YZ'
```

Example 4

You want to generate a name based on characters three to six of the member name supplied in $FROMNAME. The new name must be no more than eight characters, but $FROMNAME can be any length from one to eight characters.

For example, if the value of $FROMNAME is 'ABC123' you want to generate a value of 'XCYZ'.

This example is very similar to the previous one, except that $FROMNAME can be shorter than the value returned by keyword substitution. In other words, if the value of $FROMNAME is 'ABC', the statement used for Example 3 would generate a value of 'XC YZ' (note the three embedded blanks).

To eliminate the embedded blanks, you introduce a temporary user keyword, TEMP so that the new name generated is 'XCYZ' (without any blanks).

The following modeling statements assign the needed value to USERVAR.

```
@ TEMP = '<$FROMNAME,3,4>'
@ USERVAR = 'X<TEMP>YZ'
```

Nested Keyword Substitution

As shown before, the keyword substitution format is:

```
<keyword,position,length>
```

where position and length are optional. You can substitute other keywords into a keyword name being substituted. This is best described by the examples below.

Examples

Given

```
@A = 'Y'
@B = 'YES'
@C = 'XYZ'
```

The result of

```
<ABC,5,7>
```

is the same as the result of

```
<XAction:Z,5,7>
```

and is the same as the result of

```
<X<B,1,1>Z,5,7>
```

and is the same as the result of

```
<©,5,7>
```

In the first example (<ABC,5,7>), variable ABC is looked up and position 5 for a length of 7 is substituted.

In the second example (<XAction:Z,5,7>), the value of A is looked up and substituted in XAction:Z, giving XYZ. Variable XYZ is then substituted just as in the first example.

In the third example (<X<B,1,1>Z,5,7>), the value of B is looked up. Position 1 for a length of 1 (the character Y) is substituted into <X<B,1,1>Z, once again giving XYZ. This is then substituted just as in the first example.

In the fourth example (<©,5,7>), the value of C is looked up. The result (the characters XYZ) are substituted into © once again giving XYZ. This is then substituted just as in the first example.

This can be used to access variables associated with different move levels for a Library Code. Perhaps there is a user who should be notified when moves into a Library Code's QA library take place, and a different user that should be notified when moves into that Library Code's production library take place. And perhaps the same holds true for other Library Codes as well, with different users to be notified. Within the Library Code's model specifications you could place the following:

```
QA_NOTIFY_USER = 'JOHNSON'
PROD_NOTIFY_USER = 'SMITH'
```

When the model constructs the JCL to notify the user, the following can be used to access the appropriate variable:

```
WHO_TO_NOTIFY = '<<$DEST1SHORTNAME>_NOTIFY_USER>'
```

In another example, you have a model that needs to test and set a flag to determine whether this is the first time is has been entered for a Move level, such a QA or PROD. Each level uses its own first time user keyword flag, with the level name as part of the keyword name.

```
@ IF $G$<$DEST1SHORTNAME>_FIRST_TIME = ' '
@   $G$<$DEST1SHORTNAME>_FIRST_TIME = 'N'
... first time processing ...
 @ ENDIF
```

When the Move level is QA, these statements would be the same as:

```
@ IF $G$QA_FIRST_TIME = ' '
@   $G$QA_FIRST_TIME = 'N'
... first time processing ...
 @ ENDIF
```

When the Move level is PROD, these statements would be the same as:

```
@ IF $G$PROD_FIRST_TIME = ' '
@   $G$PROD_FIRST_TIME = 'N'
... first time processing ...
 @ ENDIF
```

Rules

You cannot nest keywords more than 10 levels deep. Nesting of keywords is only allowed for keyword names. You cannot specify keyword substitution in the position or length operands of a keyword substitution specification.

System Keywords

All system keywords are prefixed with a $ (dollar sign). Note that keywords beginning with L or G are user keywords with extended scopes. The value of each system keyword is reset at the beginning of each modeling phase using the appropriate data from the CA-PanAPT system files (with the exception of MEMBERCOUNT AND LIBCODEMEMBERCOUNT). The system keywords let you control the modeling process, and build modeling output based on the values of the keywords.

Some system keywords can be set by you in Keyword Assignment Statements while others cannot be reset. In addition, not all system keywords are available at all times for every type of Modeling facility. This chapter describes the system keywords in detail as follows:

- Resettable system keywords
- Non-Resettable system keywords
- System keywords Availability.

Resettable System Keywords

The system keywords described below have no value until they are assigned a value by Keyword Assignment Statements in your model. These resettable system keywords let you control the modeling facility directly. Note that not all of the keywords are supported in every modeling facility, check the table under System Keywords Availability later in this chapter.

$ALTLIBCFORM

Specifies an alternate Library Code to access. When you set this keyword, the other Library Codes values are accessible using the $ALT and $LVLALT prefixed keywords. Until this is specified, all the $ALT and $LVLALTLIBC keywords are null. Also if the Library Codes specified do not exist, all of the $ALT and $LVLALTLIBC prefixed keywords including this one are nullified.

This keyword must be specified in the same format as $LIBCFORM, a seven-character Library Code and Subcode formatted with a slash (xxxx/yyy). However, strict adherence is not necessary. Embedded spaces are stripped. If there is no Subcode, the slash is optional. Be aware that the value of this keyword is immediately reformatted. For instance, if you set it to xx / y it is reformatted to xx/y.

$DELIM

Specifies or resets model statement delimiters. Specify model statement delimiters with an assignment statement, such as $DELIM = 'wxyz'. Resetting model statement delimiters by coding $DELIM = RESET is equivalent to coding an assignment statement, such as $DELIM = '@<>!'. See Delimiters earlier in this chapter.

$EXTERNAL

Specifies that External Processing is required for the member being modeled. Set this keyword to a value of YES during the MOVE phase with an assignment statement, such as $EXTERNAL = 'YES'. Any other value besides YES is invalid. If you do not set the value to YES, then no External Processing is expected for the member being modeled.

External processing requires you to execute APCS5391 to post the completion of processing done outside APJJ5320. For details on External Processing, see the "Setups for Different Types of Moves" chapter in the *CA-PanAPT Reference Guide*.

$LEVEL

Specifies the level associated with the $LVL prefixed scrollable keywords. You must set this keyword to the short name of the level to change the level positioning of the scrollable values. Until you set this, all of the scrollable keywords are null. If you set this to a non-existent level name, CA-PanAPT changes the value of the $LEVEL keyword and all of the scrollable keywords to null.

For example, to access the data set name of the production library for the current Library Code, set $LEVEL to 'PROD' (or whatever the short name for production is), and access the $LVLLIBCDESTDSN scrollable variable.

$LIBCODEFLUSH

Flushes the remaining members of the current Library Code from the modeling process. Set this keyword to a value of Y at any time with an assignment statement such as, $LIBCODEFLUSH = 'Y'.

The model statements following the assignment of this keyword are processed, however, no further processing is done for the active Library Code. When this keyword is assigned during an INIT or MOVE phase, no subsequent MOVE phases or TERM phase is provided for the Library Code.

$MSG

Provides a 1 to 55-character message to the modeling facility. Online modeling facilities, such as Retrieve and Verification modeling, display the message on the user's terminal. Batch modeling facilities print the message on reports.

The message is not displayed unless a non-zero value is assigned to the system keyword $RC.

$OUTDD

Specifies the ddname of the file where modeling output records are written. Set this keyword to a value identifying any currently allocated sequential file or APTMDLO. When available, this keyword must be assigned before any modeling output records can be generated.

The ddname value assigned to this keyword can identify a sequential data set. You cannot subsequently assign a value to the $OUTMEM system keyword after assigning $OUTDD to a sequential data set. Allocate the data set with DISP=MOD because it is opened and closed multiple times. The ddname must identify a real sequential data set, it cannot be DUMMY or NULLFILE.

When the value APTMDLO is assigned to this keyword, modeling output is directed to the standard output library. The system keyword $OUTMEM must then be assigned before any modeling output records can be generated. A new member is created in the APTMDLO library, or an existing member is extended. APTMDLO is the ddname of the standard output PDS data set.

Whenever the value of $OUTDD is changed, the system keywords $OUTMEM and $OUTPOST are reset to blanks. Be careful when assigning $OUTDD to APTMDLO after having previously assigned it to a sequential file. The system keyword $OUTMEM will not have its previous value. In addition, $OUTPOST must be assigned a value for posting records to be properly generated.

$OUTMEM

Specifies the 1 to 8-character name of the member where modeling output statements are directed when $OUTDD is set to APTMDLO.

This keyword must be assigned after assigning $OUTDD with a value of APTMDLO and before any modeling output statements are generated. This keyword cannot be assigned unless $OUTDD has a value of APTMDLO.

The value of this keyword is reset to blanks whenever the value of $OUTDD is changed and at the start of each modeling phase (such as INIT, MOVE, and TERM).

$OUTPFX

A unique prefix is needed for each group's use to prevent different groups of members that are being moved (see the topic Model Processing earlier in this chapter for information about groups) from inadvertently constructing JCL and control statements into the same members on the APTMDLO data set. A common naming scheme is needed for this, and adherence to this by your models is required for this to work. This is the purpose of the $OUTPFX variable. It is initialized to a generated prefix that is unique for each group. Typically models use this keyword when the $OUTMEM keyword is specified.

For example, a model that generates a main JCL member and two control statement members referenced by the JCL sets $OUTMEM as following:

<$OUTPFX>	for the main JCL member,
<$OUTPFX>A	for the first control statement member, and
<$OUTPFX>B	for the second control statement member.

$OUTPFX is initialized by modeling as follows:

- Position 1 contains the letter M for a group being moved, or the letter B for a group being backed out.
- Positions 2 and 3 contain the abbreviated name of the destination level of the group. Each trailing blank is replaced with a $ (dollar sign).
- Positions 4 through 7 contain the Model Base for the group's Library Code. Each trailing blank is replaced with a $ (dollar sign).

This gives you a unique seven-character prefix, giving you a single character to use as a suffix. It also lets you easily identify to which group the members of APTMDLO pertain.

If this does not suit your needs, you can alter the $OUTPFX value. It must be 1 to 7 characters long and adhere to the rules for a valid PDS member name (position 1 contains uppercase alphabetic or national characters, the remaining characters contain uppercase alphabetic, national, or numeric characters). If you modify this for Move Modeling, the APJMLEAD module is an ideal place to do this, for you can be assured this is set before any of your Library Code Model specifications.

Note: The $OUTPFX variable is only available during the INIT, MOVE, and TERM phases of modeling.

$OUTPOST

Specifies the 1 to 7-character name of the PDS member where additional posting records are written. A # (pound) is automatically suffixed to the value assigned. The value you assign cannot contain a # character.

The Move Modeling Facility always generates two posting records for each member modeled. The first is written to member INITMEM# of PDS APTMDLO. You direct the second posting record to another member by assigning a value to this keyword. If a value has not been assigned to this keyword by the end of the modeling process for a member, then the second posting record is written to a member whose name is derived from the value of $OUTPFX with the # character suffix.

The value of this keyword is reset to blanks whenever the value of $OUTDD is changed, and at the start of each modeling phase (for example, INIT, MOVE, and TERM).

$RC

Specifies a four-digit Return Code. Set this keyword to any value in the range 0000 through 4096 with an assignment statement, such as $RC = '0008'. The highest value assigned during the modeling process is used to set the return code for the modeling facility.

Online modeling facilities, such as Retrieve and Verification modeling, cancel the modeling process when the return code is set to a value greater than zero. Batch modeling facilities, such as Move and On-Demand modeling, might be sensitive to specific return code values. JCL specifications in the standard Move Job (APJJ5320) cause the Move Job to terminate when $RC is assigned a value greater than 7, for instance.

Non-Resettable System Keywords

The system keywords described below cannot be assigned a value by Keyword Assignment Statements. These non-resettable system keywords provide values for your use in generating model output records. Note that not all of the keywords are available for every phase nor in every modeling facility. Any non-resettable system keyword that is unavailable has a value of blanks. See the table under System Keywords Availability later in this chapter.

Values for the non-resettable system keywords are taken from CA-PanAPT Inventory, Library Code, and Move Request data, or are provided by the modeling facility.

Keywords beginning with $ALT provide data from an alternate Library Code. The Library Code that these keywords reflect is specified using the $ALTLIBCFORM keyword. Until you set this keyword, all of the $ALT keywords have null values.

Keywords beginning with $ASSIGNED or with $DIBS provide data from the Inventory Record for the member being modeled. If inventory is not active for a Library Code, then these keywords have a value of blanks when members of the Library Code are being modeled. When inventory is active, some inventory keywords might still have a value of blanks. You can use the value of the $ASSIGNED keyword to determine whether inventory keywords contain useful values for a member.

Keywords beginning with $DEST are given by the modeling facility and provide single keywords that apply to the primary and secondary targets of a move. These keywords can be used without the conditional logic that might otherwise be necessary. The primary target is the final destination for members of a move. The keywords with names that begin with $DEST1 identify the primary target. Keywords that start with $DEST2 identify the secondary target. A secondary target is the backup for the primary target when applicable (for example, Backup or Back Out libraries). During Verification Modeling, the $DEST keywords contain the tentative values that they will have during the next move.

Keywords with names that begin with $LVL contain information that varies by level. The level that the current contents of these keywords reflects is controlled by the $LEVEL keyword. Until you set this keyword, all of the $LVL keywords have null values.

Furthermore, keywords beginning with $LVLALTLIBC contain information that varies by level for the alternate Library Code. These keywords have null values until you set both the $ALTLIBCFORM and the $LEVEL keywords.

Keywords with names that begin with $CFG contain CA-Pan/LCM Configuration Manager data from the Control file. If you did not add Configuration Manager information to your Control file, these keywords have null values.

Keywords with names that begin with $ORIG refer to the source or sending library for a move, or to the library where members are currently stored when no movement is implied (for example, On-Demand modeling). During Verification Modeling, the $ORIG keywords contain the tentative values they will have during the next move.

$ALT

Keywords beginning with $ALT provide data from an alternate Library Code. The Library Code that these keywords reflect is specified using the $ALTLIBCFORM keyword. Until you set this keyword, all of the $ALT keywords have null values.

$ALTLIBC

The four-character Library Code.

$ALTLIBCDESC

The 1 to 55-character Description field value from the Library Code.

$ALTLIBCODE

The 1 to 7-character Library Code and Subcode combined with all blanks and slashes removed (xxxxyyy).

$ALTLIBSUBC

The three-character Library Subcode.

$ALTMODELBASE

The 1 to 4-character Model Base.

$ALTRELCOMPINC

Zero to 4 sets of formatted Library Codes and Subcodes used for the CA-Panvalet ++INCLUDE or CA-Librarian -INC members for compiles. CA-PanAPT formats each set with a slash (xxxx/yyy). The first set occupies positions 1 through 8 of this keyword, the second occupies positions 10 through 17, the third occupies positions 19 through 26, and the fourth occupies positions 28 through 35. One or more spaces is padded to the end of each set to ensure the next set starts at the correct position. For example:

```
1               10      19      28
|               |       |       |
xxxx/yyy        xxxx    xx/yy   xxxx/yyy
```

$ALTRELCOMPSYSLIB

Zero to 4 sets of formatted Library Codes and Subcodes used for the compile syslibs (for expanding COPY/INCLUDE statements and macros). See $ALTRELCOMPINC for an explanation of how this is formatted.

$ALTRELLKEDINC

Zero to 4 sets of formatted Library Codes and Subcodes used for the CA-Panvalet ++INCLUDE or CA-Librarian -INC members for Link Edits. See $ALTRELCOMPINC for an explanation of how this is formatted.

$ALTRELLKEDSYSLIB

Zero to 4 sets of formatted Library Codes and Subcodes used for the Link Edit syslibs (for including object and load modules). See $ALTRELCOMPINC for an explanation of how this is formatted.

$ALTRELLKEDSYSLIN

An optional Library Code and Subcode formatted with a slash (xxxx/yyy) used for Link Edit control statement members.

$ALTRELOUTEXEC

An optional Library Code and Subcode formatted with a slash (xxxx/yyy) used for holding executable output, typically load modules.

$ALTRELOUTLISTING

An optional Library Code and Subcode formatted with a slash (xxxx/yyy) used for holding output listings.

$ALTRELOUTOBJECT

An optional Library Code and Subcode formatted with a slash (xxxx/yyy) used for holding object members produced from compiles.

$ALTRELOUTOTHER

An optional Library Code and Subcode formatted with a slash (xxxx/yyy) used for holding other types of output, such as DB2 DBRMs.

$ALTRELOUTSOURCE

An optional Library Code and Subcode formatted with a slash (xxxx/yyy) used for holding generated source output, such as what is produced when a CICS BMS map is compiled or what is produced from a DB2 DCLGEN.

$ALTRELPRECOMPSYSLIB

Zero to 4 sets of formatted Library Codes and Subcodes used for the pre-compile syslibs. For instance, this would be used for expanding DB2 SQL FETCH statements. See $ALTRELCOMPINC for an explanation of how this is formatted.

$ASSIGNED

A one-character value that indicates if the current member's Inventory Record is assigned Y, unassigned N, or not available (blank). When the value is blank, then there is no Inventory Record for the member and the other inventory keywords are also blank.

$ASSIGNEDMR

The six-digit number of the Move Request to which the Inventory Record is assigned. The value can be different from the value of $MR.

$ASSIGNEDTO

The 1 to 8-character user ID of the CA-PanAPT to whom this Inventory Record is assigned. The value is blank when the Inventory Record is not assigned or is unavailable.

$CFG

Keywords with names that begin with $CFG contain CA-Pan/LCM Configuration Manager data from the Control file. If you did not add Configuration Manager information to your Control file, these keywords have null values.

$CFGAMKLISTARGS

The 0 to 50-character AMKLIST argument string.

$CFGDEPMBR

The 1 to 8-character member name for the Dependency file member.

$CFGOPTMBR

The 1 to 8-character member name for the Global Options member, or null if there is no Global Options member.

$CFGPROFDSN

The 1 to 44-character name of the CA-Pan/LCM PROFILE data set.

$CFGPROJDSN

The 1 to 44-character name of the CA-Pan/LCM Project data set.

$CFGSUPPERRS

A one-character value indicating whether AMKLIST errors are to be suppressed while running Verification Procedures. The values can be Y, indicating errors are to be suppressed, or N.

$CFGWARNLVL

The one-character warning level value to be provided to AMKLIST while running Verification Procedures. The value can be 0, 1, 2, or 3.

$CHKREP

The one-character REPLACE value for the member as specified on the Retrieve Options panel (APIP710). The value indicates that a member with the same name on the destination library is to be replaced Y or is not to be replaced N.

$DBDSN

The 1 to 44-character data set name of the CA-PanAPT database. This is the name associated with the APTDB DD in effect while Modeling is running.

$DB2TYPE

The 0 to 8-character DB2 Entity type for the Library Code. If you do not have the CA-PanAPT DB2 option, this keyword is null. The value of this keyword is derived from the Library Code type as follows:

Library Code Type	DB2 Type
DB2SRC	SOURCE
DB2SDB	SRCEDBRM
DB2DBRM	DBRM
DB2PACK	PACKAGE
DB2PLAN	PLAN
all others	null

$DEST*n*

Keyword names beginning with $DEST1 provide a single keyword that applies to the target of the move and can be used without conditional logic. Keyword names beginning with $DEST2 apply to a secondary target (for example, the Backup or Back Out Library).

$DEST1AM

The 1 to 2-character Access Method code for the primary destination data set.

$DEST1DDN

The 1 to 8-character ddname for the primary destination data set.

$DEST1DSN

The 1 to 44-character dsname for the primary destination data set.

$DEST1SEC

The 1 to 10-character security data for the primary destination data set.

$DEST1SHORTNAME

The 1 to 4-character short name associated with the destination level.

For Moves and Back outs, the $DEST1 fields pertain to the move level. For a Production Move/Back out, the Production level is the one these fields pertain to.

For Retrieve modeling, the $DEST1 fields pertain to the starting Test level.

When no members are being moved, as with On-demand and Verification modeling, these keywords all have values of blanks.

$DEST2AM

The 1 to 2-character Access Method code for the secondary destination data set.

$DEST2DDN

The 1 to 8-character ddname for the secondary destination data set.

$DEST2DSN

The 1 to 44-character dsname for the secondary destination data set.

$DEST2SEC

The 1 to 10-character security data for the secondary destination data set.

For Moves, the $DEST2 fields pertain to the Back up data set for the move level.

For Back Outs, the $DEST2 fields pertain to the Back Out data set for the move level.

When no members are being moved, as with On-demand and Verification modeling, or when no secondary destination is applicable, as with Retrieve modeling or moves without Back Up data sets, these keywords are all set to null values.

$DIBSAPPL

The 1 to 8-character Application field value from the Inventory Record.

$DIBSAPPROVED

A one-character value indicating whether the Inventory Record is approved Y, not approved N, or unavailable (blank).

$DIBSCICSOPT

The 1 to 60-character CICS precompiler option field from the Inventory Record.

$DIBSCOM

The 1 to 55-character Comment field value from the Inventory Record.

$DIBSCOMPOPT

The 1 to 60-character Compiler Options field value from the Inventory Record.

$DIBSDBOPT

The 1 to 60-character Data Base precompiler option field from the Inventory Record.

$DIBSDESC

The 1 to 55-character Description field value from the Inventory Record.

$DIBSENV

The 1 to 8-character Environment field value from the Inventory Record.

$DIBSLANG

The 1 to 8-character Language field value from the Inventory Record.

$DIBSLINKOPT

The 1 to 60-character Link Edit field value from the Inventory Record.

$DIBSLINKSTR

The 1 to 10-character Link Edit Stream member name from the Inventory Record.

$DIBSOWNER

The 1 to 8-character user ID of the Permanent owner of the Inventory Record.

$DIBSQUAL

The 1 to 8-character Inventory Qualifier of the Inventory Record.

$DIBSSAVEOBJECT

The one-character value from the Inventory Record that indicates whether to save an object module. The value can be one of the following:

blank	Default to the Library Code setting.
Y	Save object.
N	Do not save object.

Also see keyword $SAVEOBJECT, which has a definitive Y/N value.

$DIBSSAVELIST

The one-character value from the Inventory Record that indicates whether to save a listing. The value can be one of the following:

blank	Default to the Library Code setting.
Y	Save listing.
N	Do not save listing.

Also see keyword $SAVELIST, which has a definitive Y/N value.

$DIBSSAVELOAD

The one-character value from the Inventory Record that indicates whether to create a load module. The value can be one of the following:

blank	Default to the Library Code setting.
Y	Create load module.
N	Do not create load module.

Also see keyword $SAVELOAD, which has a definitive Y/N value.

$DIBSUSER01-through-$DIBSUSER20

The user data field values from the Inventory Record. $DIBSUSER01 through $DIBSUSER05 are 1 to 8-character values. $DIBSUSER06 through $DIBSUSER10 are 1 to 16-character values, and $DIBSUSER11 through $DIBSUSER20 are 1 to 50-character values.

$ESSHORTNAME

If there is an Early Stop level specified for a Move Request, this keyword contains the 1 to 4-character short name of that level. Otherwise, this field is null.

$FIRSTLVL

The 1 to 4-character short name for the first level defined to your CA-PanAPT system, commonly referred to as the Test level.

$FIRSTRUNDATE

The First Run date for the Move Request in the format YYYYMMDD.

$FROMDATA

The 1 to 8-character From User-data field for the member on the $ORIGDSN library.

For moves from the starting Test level, the value is the From Starting Level User-data field specified on the Member Moves panel (APIP140). For moves from subsequent levels, and for Back Outs, the value is the To Destination Lvls User-data field.

For Retrieve, the value is the From User-data field on the Retrieve Processing Options panel (APIP710).

$FROMNAME

The 1 to 10-character name for the member on the $ORIGDSN library.

For moves from the starting Test level, the value is the From Starting Level Member field specified on the Member Moves panel (APIP140). For moves from subsequent levels, and for Back Outs, the value is the To Destination Lvls Member field.

For Retrieve, the value is the From Member name on the Retrieve Processing Options panel (APIP710).

$LASTLVL

The 1 to 4-character short name of the last move level defined to your CA-PanAPT system. Typically this is your PROD level, but it doesn't have to be.

$LEVELS

A three-digit number that describes the number of levels defined to your CA-PanAPT system. For example, in a system with Test, Quality Assurance, and Production defined, the value is 003.

$LIBC

The four-character Library Code.

$LIBCDESC

The 1 to 55-character Description field value from the Library Code.

$LIBCFORM

The seven-character Library Code and Subcode formatted with a slash (xxxx/yyy).

$LIBCODE

The 1 to 7-character Library Code and Subcode concatenated with all blanks and slashes removed (xxxxyyy).

$LIBSUBC

The three-character Library Subcode.

$LIBCODETYPE

The 0 to 8-character Type field from the Library Code.

$LIBCODEMEMBERCOUNT

A seven-digit count of the number of members processed because the INIT phase in the range 0000000 through 9999999. The value is set to 0000000 in the INIT phase and incremented by one for each member processed.

$LVL

Keywords with names that begin with $LVL contain information that varies by level. The level that the current contents of these keywords reflects is controlled by the $LEVEL keyword. Until you set this keyword, all of the $LVL keywords have null values. If you specify a non-existent level name for the $LEVEL keyword, all $LVL keywords have a null value, and the $LEVEL keyword is reset to a null value.

Keywords with names that begin with $LVLLIBC contain information defined to the current Library Code for the specified level. If the level controlled by the $LEVEL keyword is not defined to the Library Code, all of the $LVLLIBC keywords contain null values.

Keywords with names that begin with $LVLALTLIBC contain information defined to the alternate Library Code for the specified level. Like the $LVLLIBC keywords, if the level controlled by the $LEVEL keyword is not defined to the alternate Library Code, all of the $LVLALTLIBC keywords contain null values.

$LVLABBREVNAME

The 1 to 2-character abbreviated name of the level.

$LVLALTLIBCACTIVE

A one-character value that contains the letter A if the level is active for the alternate Library Code or that contains the letter I if the level is inactive.

$LVLALTLIBCBKOTAM

The 0 to 2-character Access Method code for the alternate Library Code's Back Out data set for the level.

$LVLALTLIBCBKOTDDN

The 1 to 8-character ddname of the alternate Library Code's Back Out data set for the level. If the level does not use Back Out, this keyword is null.

$LVLALTLIBCBKOTDSN

The 1 to 44-character data set name of the alternate Library Code's Back Out data set for the level. If the level does not use Back Out, this keyword is null.

$LVLALTLIBCBKOTSEC

The 0 to 10-character security data for the alternate Library Code's Back Out data set for the level.

$LVLALTLIBCBKUPAM

The 0 to 2-character Access Method code for the alternate Library Code's Back Up data set for the level.

$LVLALTLIBCBKUPDDN

The 1 to 8-character ddname of the alternate Library Code's Back Up data set for the level. If the level does not use Back Up, this keyword is null.

$LVLALTLIBCBKUPDSN

The 1 to 44-character data set name of the alternate Library Code's Back Up data set for the level. If the level does not use Back Up, this keyword is null.

$LVLALTLIBCBKUPSEC

The 0 to 10-character security data for the alternate Library Code's Back Up data set for the level.

$LVLALTLIBCDESTAM

The 0 to 2-character Access Method code for the alternate Library Code for the level.

$LVLALTLIBCDESTDDN

The 1 to 8-character ddname of the alternate Library Code for the level.

$LVLALTLIBCDESTDSN

The 1 to 44-character data set name of the alternate Library Code for the level.

$LVLALTLIBCDESTSEC

The 0 to 10-character security data for the alternate Library Code for the level.

$LVLALTLIBCMOVECTL

The one-character Move Control value for the alternate Library Code for the level. See $LVLLIBCMOVECONTROL for a description of the values.

$LVLALTLIBCNEXT

The 1 to 4-character short name of the next highest level defined to the alternate Library Code or null if the current level is the last level for the Library Code.

$LVLALTLIBCNUMA

A three-digit number that describes the number of levels in the alternate Library Code that follow the current level.

$LVLALTLIBCNUMB

A three-digit number that describes the number of levels in the alternate Library Code that precede the current level.

$LVLALTLIBCOPTBKOT

A one-character value that contains the Backout Enabled value for the alternate Library Code for the level. When the level is the starting Test level, this keyword is null.

$LVLALTLIBCOPTBKUP

A one-character value that contains the Backup Enabled value for the alternate Library Code for the level. When the level is the starting Test level, this keyword is null.

$LVLALTLIBCPREV

The 1 to 4-character short name of the next lowest level defined to the alternate Library Code or null if the current level is the starting test level.

$LVLLIBCACTIVE

A one-character value that contains A if the level is active for the Library Code, or contains I if the level is inactive for the Library Code, and doesn't pertain to any moves.

$LVLLIBCBKOTAM

The 0 to 2-character Access Method code for the Library Code Back Out data set for the level.

$LVLLIBCBKOTDDN

The 1 to 8-character ddname of the Library Code Back Out data set for the level. If the level does not use Back Out, this keyword is null.

$LVLLIBCBKOTDSN

The 1 to 44-character data set name of the Library Code Back Out data set for the level. If the level does not use Back Out, this keyword is null.

$LVLLIBCBKOTSEC

The 0 to 10-character security data for the Library Code Back Out data set for the level.

$LVLLIBCBKUPAM

The 0 to 2-character Access Method code for the Library Code Back Up data set for the level.

$LVLLIBCBKUPDDN

The 1 to 8-character ddname of the Library Code Back Up data set for the level. If the level does not use Back Up, this keyword is null.

$LVLLIBCBKUPDSN

The 1 to 44-character data set name of the Library Code Back Up data set for the level. If the level does not use Back Up, this keyword is null.

$LVLLIBCBKUPSEC

The 0 to 10-character security data for the Library Code Back Up data set for the level.

$LVLLIBCDESTAM

The 0 to 2-character Access Method code for the primary Library Code data set for the level.

$LVLLIBCDESTDDN

The 1 to 8-character ddname of the primary Library Code data set for the level.

$LVLLIBCDESTDSN

The 1 to 44-character data set name of the primary Library Code data set for the level.

$LVLLIBCDESTSEC

The 0 to 10-character security data for the primary Library Code data set for the level.

$LVLLIBCMOVECONTROL

The one-character Move Control value for the Library Code for the level. The value can be one of the following:

blank

The level is the starting test level, where movement does not apply.

C

Members are copied into this level. They are not deleted from the origin level.

M

Members are moved into this level. They are deleted from the origin level after they are copied.

D

Members are deleted from the origin level without being copied to this level.

I

The level is inactive, and doesn't participate in moves.

$LVLLIBCNEXT

The 1 to 4-character short name of the next highest level defined to the Library Code. If the current level is the last level for the Library Code, this keyword is null.

$LVLLIBCNUMA

A three-digit number that describes the number of levels in this Library Code that follow the current level. For example, in a system with Test, Quality Assurance, and Production defined, with the current level being Test, the value is 002. When the current level is the last level, the value is 000.

$LVLLIBCNUMB

A three-digit number that describes the number of levels in this Library Code that precede the current level. For example, in a system with Test, Quality Assurance, and Production defined, with the current level being Quality Assurance, the value is 001. When the current level is the starting Test level, the value is 000.

$LVLLIBCOPTBKOT

A one-character value that contains the Backout Control value for this Library Code for the level. When the level is the starting Test level, this keyword is null.

$LVLLIBCOPTBKUP

A one-character value that contains the Backup Enabled value for this Library Code for the level. When the level is the starting Test level, this keyword is null.

$LVLLIBCPREV

The 1 to 4-character short name of the next lowest level defined to the Library Code. If the current level is the starting Test level, this keyword is null.

$LVLLONGNAME

The 1 to 20-character long name of the level.

$LVLNEXT

The 1 to 4-character short name of the next highest level defined to the CA-PanAPT system. If the current level is the last level, this keyword is null.

$LVLNUMA

A three-digit number that describes the number of levels in this CA-PanAPT system that follow the current level. For example, in a system with Test, Quality Assurance, and Production defined, with the current level being Test, the value is 002. When the current level is the last level, the value is 000.

$LVLNUMB

A three-digit number that describes the number of levels in this CA-PanAPT system that precede the current level. For example, in a system with Test, Quality Assurance, and Production defined, with the current level being Quality Assurance, the value is 001. When the current level is the starting Test level, the value is 000.

$LVLPREV

The 1 to 4-character short name of the next lowest level defined to the CA-PanAPT system. If the current level is the starting Test level, this keyword is null.

$LVLSHORTNAME

The 1 to 4-character short name of the level. This keyword's value is always the same as the $LEVEL keyword value.

$MDLODSN

The 1 to 44-character data set name of the APTMDLO modeling output PDS.

$MEMBERCOUNT

A seven-digit count of the number of members processed in the range 0000000 through 9999999. The value is incremented by one for each member processed.

$MEMBEREXIST

A one-character value that indicates whether the member has passed Y or failed N the Member Existence Exit check. A value of blank indicates that a Member Existence Exit is not specified in the Library Code for the current move, so no exit was invoked to perform the check.

$MMPCPFX

The MMPCPFX node value of data sets created during the Move Processing Cycle. The value is taken from PARM specifications for the Move and On-demand Modeling Facilities. A value of blanks indicates that it was not specified. When properly specified, the value is a 1 to 8-character standard IBM data set qualifier node.

$MODELBASE

The 1 to 4-character Model Base for the Library Code.

$MODELTIME

A one-character code that indicates what modeling facility is being used, as follows:

M

Move Modeling for normal movement

B

Move Modeling for Back Out movement

C

Retrieve Modeling

V

Verification Procedure Modeling

O

On-Demand Modeling

$MODEL01-through-$MODEL12

The 1 to 75-character Model Specification Statements (up to 12 lines) currently being processed.

$MOVEDATE

The next scheduled move date for the Move Request in the format YYYYMMDD. For moves to a level before the final level, this comes from the next scheduled move date, which can be blank. For the final level, this contains the final move date.

$MOVETYPE

The one-character Move Type field value from the Move Request. The Move Type can be used to control the Move Processing Cycle. Use of Move Processing Cycles is dependent upon your site's requirements.

$MR

The six-digit Move Request number of the Move Request being processed, or of the Move Request the member being modeled is assigned.

When this keyword has a value of blanks, then the other Move Request related keywords, such as $MOVEDATE, $MOVETYPE, $MRDESC, and $SR also hold blanks.

$MRCFGINOPTMBR

The 1 to 8-character member name of the CA-Pan/LCM Configuration Manager Option member for specifying additional impacting members for the Impact Analysis Member Selection List (MSL) and Verification Procedure. If this name was not specified when obtaining an Impact Analysis MSL or if you did not obtain an Impact Analysis MSL for this Move Request then this keyword is null.

$MRCFGOUTOPTMBR

The 1 to 8-character member name of the CA-Pan/LCM Configuration Manager Option member for restricting the impacted members for the Impact Analysis Member Selection List (MSL) and Verification Procedure. If this name was not specified when obtaining an Impact Analysis MSL or if you did not obtain an Impact Analysis MSL for this Move Request, this keyword is null.

$MRDESC

The 1 to 55-character Description field value from the Move Request identified by $MR.

$MREXDESC1-through-$MREXDESC12

The 1 to 70-character Expanded Description field values from the Move Request identified by $MR (up to 12 lines).

$OPT

Keyword names that begin with $OPT provide data from the Library Code definition. Some $OPT keywords provide data derived from the Library Code definition and the move time. These keywords let you simplify models because CA-PanAPT modeling has already done the tests associated with move time. The other $OPT keywords provide data directly from the Library Code definition.

$OPTD1TOD2

A one-character value indicating whether primary to secondary destination moves are supported, Y, or are not supported, N, for this move.

For normal moves in which the Library Code has Backup enabled, this keyword has a value of Y, because the Backup Library is the secondary destination (the $DEST2 data set).

For Back Out moves in which the Library Code has Backout Control specified as B (Backout and restore), this keyword also has a value of Y, because the Backout Library is the secondary destination.

In all other cases, this keyword has a value of N.

$OPTORIGDEL

A one-character value indicating whether to delete, Y, or not to delete, N, the member in the $ORIGDSN data set after this move is complete.

$OPTORIGTOD1

A one-character value that indicates whether a primary move is to be performed, Y, or is not to be performed, N.

For normal moves, this keyword has a value of Y when the Library Code's Move Control has a value of C (Copy) or M (Move), and a value of N when the Move Control value is D (Delete).

For Back Out moves in which the Library Code has Backout Control specified as B (Backout and restore) or R (Restore without backout), this keyword has a value of Y. When Backout Control is P (Prohibited) or blank, this keyword has a value of N.

Note that it is possible for $OPTORIGDEL to have a value of Y while this keyword is set to a value of N.

$ORIG

Keyword names that begin with $ORIG refer to the source or sending library for a move, or to the Library where members are currently stored when no movement is implied (for example, On-Demand modeling).

For normal moves, the $ORIG keywords pertain to the primary data set for the level the members are being moved from.

For Backout moves, the values pertain to the Back up data set for the current move level.

For Retrieve, the values depend upon which level you are retrieving from and whether it is from the primary data set or the Back up or Back out data set.

When no members are being moved, as with On-demand and Verification modeling, the values correspond to what they would be if the members were being moved at that time, based upon the current status of the Move Request. If the Move Request is in a Move Complete, Back Out Complete, or Deleted status, the keyword values are null.

$ORIGAM

The 1 to 2-character Access Method code for the origin data set.

$ORIGDDN

The 1 to 8-character ddname for the origin data set.

$ORIGDSN

The 1 to 44-character dsname for the origin data set.

$ORIGSEC

The 1 to 10-character security data for the origin data set.

$ORIGSHORTNAME

The 1 to 4-character short name of the origin move level.

$OTHRPFX

The 1 to 8-character OTHRPFX data set name prefix specified during the execution of the APJP5320 proc. This allows your models to construct the same OTHER data set names that APJP5320 used.

$OWNER

The 1 to 8-character user ID of the Owner field value from the Move Request identified by $MR.

$PHASE

A four-character code indicating which Model Processing Phase is currently active. This keyword is always available with a value of START, INIT, MOVE, TERM, or END. See Model Processing earlier in this chapter.

$REL

Keyword names that begin with $REL contain the names of Library Codes that are related to the current Library Code. They fall into two formats. One is where a single Library Code name is specified, and the other is where a list of Library Code names is specified. Each library code, whether its the value for a single Library Code or one of the Library Codes in a list, is formatted such that the $ALTLIBCFORM keyword can be set to it. In the listed values, each Library Code except the last is padded with spaces on the end so its total length is nine characters, making it easy to pick the next Library Code out of the list.

$RELCOMPINC

A list of up to four sets of formatted Library Codes used for the CA-Panvalet ++INCLUDE or CA-Librarian -INC members used for compiles.

$RELCOMPSYSLIB

A list of up to four sets of formatted Library Codes used for the compile syslibs (for expanding COPY/INCLUDE statements and macros.

$RELLKEDINC

A list of up to four sets of formatted Library Codes used for the CA-Panvalet ++INCLUDE or CA-Librarian -INC members used for Link Edits.

$RELLKEDSYSLIB

A list of up to four sets of formatted Library Codes used for the Link Edit syslibs (for including object and load modules).

$RELLKEDSYSLIN

An optional formatted Library Code used for Link Edit control statement members.

$RELOUTEXEC

An optional formatted Library Code used for holding executable output, typically load modules.

$RELOUTLISTING

An optional formatted Library Code used for holding output listings.

$RELOUTOBJECT

An optional formatted Library Code used for holding object members produced from compiles.

$RELOUTOTHER

An optional formatted Library Code used for holding other types of output, such as DB2 DBRMs.

$RELOUTSOURCE

An optional formatted Library Code used for holding generated source output, such as what is produced when a CICS BMS map is compiled or what is produced from a DB2 DCLGEN.

$RELPRECOMPSYSLIB

A list of up to four sets of formatted Library Codes used for the pre-compile syslibs. For instance, this would be used for expanding DB2 SQL FETCH statements.

$SAVEOBJECT

A one-character value that indicates whether to save the object from a compile. This has a value of Y or N. If the member has inventory, and its inventory states to save the object or not, then the value of this field is determined from the inventory. Otherwise, the value is Y if the Library Code has a related object module Library Code, or it is N if there is no related object module Library Code.

$SAVELIST

A one-character value that indicates whether to save listings. This has a value of Y or N. If the member has inventory, and its inventory states to save listings or not, then the value of this field is determined from the inventory. Otherwise, the value is Y if the Library Code has a related listing Library Code, or it is N if there is no related listing Library Code.

$SAVELOAD

A one-character value that indicates whether to create load modules. This has a value of Y or N. If the member has inventory, and its inventory states to create load modules or not, then the value of this field is determined from the inventory. Otherwise, the value is Y if the Library Code has a related load module Library Code, or it is N if there is no related load module Library Code.

$SR

The 1 to 16-character Service Request field value from the Move Request identified by $MR.

$TESTDATA

The 1 to 8-character From Starting Level User-data field for the member as specified on the Member Moves panel (APIP140). The starting level is commonly referred to as the TEST level.

$TESTNAME

The 1 to 10-character From Starting Level Member name field for the member as specified on the Member Moves panel (APIP140). The starting level is commonly referred to as the TEST level.

$TODATA

The 1 to 8-character To Data-user field value for the member.

For Retrieve, the value is the To Data-user field on the Retrieve Processing Options panel (APIP710). Otherwise, the value is the To Destination Lvls User-data field specified on the Member Moves panel (APIP140).

$TODAY

Today's system date in the format YYYYMMDD.

$TONAME

The 1 to 10-character To Destination Lvls Member or To Member name field value for the member.

For Retrieve, the value is the To Member name on the Retrieve Processing Options panel (APIP710). Otherwise, the value is the To Destination Lvls Member name specified on the Member Moves panel (APIP140).

$USERID

The 1 to 8-character user ID user performing the modeling. Online modeling facilities, such as Retrieve and Verification, set this keyword to the value of the TSO userid performing the Retrieve or Verification.

$UT

Keyword names that begin with $UT are used during Compare and Merge utility model processing.

During Compare processing, the $UTLIST keywords describe the output location for the report, the $UT1 keywords describe the NEW member, and the $UT2 keywords describe the OLD member.

During Merge processing, the $UTLIST keywords describe the output location for the report. The $UT1 and $UT2 keywords describe the members being merged. The $UT3 keywords describe the parent member for the merge. The $UTWORK1, $UTWORK3, and $UTWORK3 keywords describe work data sets to stage the $UT1, $UT2, and $UT3 members to when CA-Pan/Merge cannot read them directly. The $UTOUT keywords describe where to create the merged result.

$UTLISTDSN
$UTOUTDSN
$UTWORK1DSN
$UTWORK2DSN
$UTWORK3DSN
$UT1DSN
$UT2DSN
$UT3DSN

The 1 to 44-character dsname for the utility data set.

$UTLISTNAME

$UTOUTNAME

$UT1NAME

$UT2NAME

$UT3NAME

The 1 to 10-character member name.

$UTLISTAM

$UTOUTAM

$UTWORK1AM

$UTWORK2AM

$UTWORK3AM

$UT1AM

$UT2AM

$UT3AM

The 0 to 2-character Access Method code.

$UTLISTSEC

$UTOUTSEC

$UT1SEC

$UT2SEC

$UT3SEC

The 0 to 10-character security data.

$VERCAT

The two-character Verification Category for the Verification being modeled.

$VERDESC

The 1 to 25-character description of the Verification Category for the Verification being modeled.

$VERREQUIRED

A one-character value indicating whether the Verification category being modeled is required by the current Library Code. This can be used to bypass members that do not pertain to the Verification Procedure if appropriate.

$VERSHORTNAME

The 1 to 4-character short name of the level at which the Verification is being run.

$VSAMPFX

The prefix for your CA-PanAPT VSAM files. Normally this should be the same as the VSAMPFX specification in the APJP5320 or APJP5920 procs; however, it is not derived from there. The current data set name of the APTDB file (the CA-PanAPT database) is stripped of the .APTDB suffix, giving this keyword value. If the data set name of your database does not end with an APTDB qualifier, the value of this keyword is null.

This keyword is useful for constructing JCL that uses the procs provided with CA-PanAPT, most of which require a VSAMPFX parameter.

$X00

A string of 80 binary zeros. When using products such as CA-Panexec to invoke compilers, it is often necessary to provide ddname overrides to compilers. Usually the parm string passed to the compiler required binary zeros as a place holder for any ddnames that are not altered. This keyword can be used to generate the binary zero place holders.

System Keywords Availability

The list shown on the next few pages provides information showing the times or processing phases when the system keywords have been initialized and are available. The column keyword entries are:

S

Keyword is available during the Start phase

I

Keyword is available during the Init phase

M

Keyword is available during the Move phase

T

Keyword is available during the Term phase

E

Keyword is available during the End phase

N/A

Keyword does not apply to the Modeling facility

****** **(Retrieve)**

The value comes from the Move Request where the member is assigned or is blank.

++

The value is null if the corresponding level is a development level.

A modeling error results if a model attempts to use a keyword when it is not available.

Checkin/ Cancel

Keywords	Checkin Move	Checkout/ Retrieve	On Demand	Verifi- cation	Compile	for Checkout	Cancel Migration	Development	Compare	Merge
$ALTLIBC	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTLIBCDESC	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTLIBCFORM	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTLIBCODE	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTLIBSUBC	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTMODELBASE	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELCOMPINC	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELCOMPSYSLIB	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELLKEDINC	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELLKEDSYSLIB	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELLKEDSYSLIN	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELOUTEXEC	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELOUTLISTING	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELOUTOBJECT	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELOUTOTHER	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELOUTSOURCE	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ALTRELPRECOMPSYSLIB	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$ASSIGNED	M	M	M	M	M	M	M	M	M	M

$ASSIGNEDMR	M	M	M	M	M	M	M	M
M	M							
$ASSIGNEDTO	M	M	M	M	M	M	M	M
M	M							
$CFGAMKLISTARGS	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$CFGDEPMBR	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$CFGOPTMBR	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$CFGPROFDSN	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$CFGPROJDSN	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							

Keywords	**Checkin/ Move**	**Retrieve**	**Checkin On Demand**	**Cancel Verifi-cation**	**Compile**	**Checkout for Development**	**Migration**	**Cancel Checkout Compare**	**Merge**
$CFGSUPPERRS	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT
IMT	IMT								
CFGWARNLVL	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT
IMT	IMT								
$CHKREP	N/A	IMT	N/A	N/A	N/A	IMT	N/A	N/A	N/A
N/A	N/A								
$DBDSN	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT
IMT	IMT								
$DB2TYPE	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT								
$DELIM	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT
IMT	IMT								
$DEST1AM	IMT	IMT	N/A	IMT	IMT	IMT	IMT	N/A	N/A
N/A	N/A								
$DEST1DDN	IMT	IMT	N/A	IMT	IMT ++	N/A	IMT	N/A	N/A
N/A	N/A								
$DEST1DSN	IMT	IMT	N/A	IMT	IMT	IMT	IMT	N/A	N/A
N/A	N/A								
$DEST1SEC	IMT	IMT	N/A	IMT	IMT	IMT	IMT	N/A	N/A
N/A	N/A								

$DEST1SHORTNAME	IMT	IMT	N/A	IMT	IMT	IMT	IMT	N/A
N/A	N/A							
$DEST2AM	IMT	N/A	N/A	IMT	N/A	N/A	N/A	N/A
N/A	N/A							
$DEST2DDN	IMT	N/A	N/A	IMT	N/A	N/A	N/A	N/A
N/A	N/A							
$DEST2DSN	IMT	N/A	N/A	IMT	N/A	N/A	N/A	N/A
N/A	N/A							
$DEST2SEC	IMT	N/A	N/A	IMT	N/A	N/A	N/A	N/A
N/A	N/A							
$DIBSAPPL	M	M	M	M	M	M	M	M
M	M							
$DIBSAPPROVED	M	M	M	M	M	M	M	M
M	M							
$DIBSCICSOPT	M	M	M	M	M	M	M	M
M	M							
$DIBSCOM	M	M	M	M	M	M	M	M
M	M							
$DIBSCOMPOPT	M	M	M	M	M	M	M	M
M	M							
$DIBSDBOPT	M	M	M	M	M	M	M	M
M	M							
$DIBSDESC	M	M	M	M	M	M	M	M
M	M							
$DIBSENV	M	M	M	M	M	M	M	M
M	M							
$DIBSLANG	M	M	M	M	M	M	M	M
M	M							
$DIBSLINKOPT	M	M	M	M	M	M	M	M
M	M							
$DIBSLINKSTR	M	M	M	M	M	M	M	M
M	M							
$DIBSOWNER	M	M	M	M	M	M	M	M
M	M							
$DIBSQUAL	M	M	M	M	M	M	M	M
M	M							
$DIBSSAVEOBJECT	M	M	M	M	M	M	M	M
M	M							
$DIBSSAVELIST	M	M	M	M	M	M	M	M
M	M							
$DIBSSAVELOAD	M	M	M	M	M	M	M	M
M	M							

$DIBSUSER01-20	M	M	M	M	M	M	M	M
M	M							
$ESSHORTNAME	M	M **	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$EXTERNAL	M	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	N/A							
$FIRSTLVL	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$FIRSTRUNDATE	M	M **	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$FROMDATA	M	M	M	M	M	M	M	M
M	M							
$FROMNAME	M	M	M	M	M	M	M	M
M	M							
$LASTLVL	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$LEVEL	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$LEVELS	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$LIBC	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$LIBCDESC	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$LIBCFORM	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$LIBCODE	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$LIBCODEFLUSH	IMT	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	N/A							
$LIBCODEMEMBERCOUNT	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$LIBCODETYPE	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$LIBSUBC	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$LVLABBREVNAME	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$LVLALTLIBCACTIVE	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$LVLALTLIBCBKOTAM	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							

$LVLALTLIBCBKOTDDN	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCBKOTDSN	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCBKOTSEC	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCBKUPAM	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCBKUPDDN	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCBKUPDSN	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCBKUPSEC	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCDESTAM	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCDESTDDN	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	

Checkin/ Cancel

Checkin Checkout/

Keywords	**Move**	**Retrieve**	**On Demand**	**Verifi-cation**	**Compile**	**for Checkout**	**Cancel Migration**	**Development**	**Compare**	**Merge**
$LVLALTLIBCDESTDSN	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCDESTSEC	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCMOVECTL	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCNEXT	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCNUMA	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCNUMB	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCOPTBKOT	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$LVLALTLIBCOPTBKUP	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	

```
$LVLALTLIBCPREV       SIMTE  IMT      SIMTE   SIMTE    IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCACTIVE        IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCBKOTAM        IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCBKOTDDN       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCBKOTDSN       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCBKOTSEC       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCBKUPAM        IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIVCVKUPDDN       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCBKUPDSN       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCBKUPSEC       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCDESTAM        IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCDESTDDN       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCDESTDSN       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCDESTSEC       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCMOVECONTROL    IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCNEXT          IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCNUMA          IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCNUMB          IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCOPTBKOT       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCOPTBKUP       IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
$LVLLIBCPREV          IMT    IMT      IMT     IMT      IMT      IMT      IMT       IMT
IMT      IMT
```

```
$LVLLONGNAME          SIMTE  IMT      SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$LVLNEXT              SIMTE  IMT      SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$LVLNUMA              SIMTE  IMT      SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$LVLNUMB              SIMTE  IMT      SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$LVLPREV              SIMTE  IMT      SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$LVLSHORTNAME         SIMTE  IMT      SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$MDLODSN              SIMTE  N/A      SIMTE   N/A      N/A      N/A       N/A       N/A
N/A       N/A
$MEMBERCOUNT          SIMTE  IMT      SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$MEMBEREXIST          M      N/A      N/A     N/A      N/A      N/A       N/A       N/A
N/A       N/A
$MMPCPFX              SIMTE  N/A      SIMTE   N/A      N/A      N/A       N/A       N/A
N/A       N/A
$MODELBASE            IMT    IMT      IMT     IMT      IMT      IMT       IMT       IMT
IMT       IMT
$MODELTIME            SIMTE  IMT      SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$MODEL01-03           IMT    IMT      SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$MODEL04-12           IMT    IMT      SIMTE   N/A      IMT      IMT       IMT       IMT
IMT       IMT
$MOVEDATE             M      M **     SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$MOVETYPE             M      M **     SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$MR                   M      M **     SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$MRCFGINOPTMBR        M      M **     SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$MRCFGOUTOPTMBR       M      M **     SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$MRDESC               M      M **     SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
$MREXDESC1-12         M      M **     SIMTE   SIMTE    IMT      IMT       IMT       IMT
IMT       IMT
```

$MSG	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT	
$OPTD1TOD2	IMT	N/A	N/A	IMT	N/A	N/A	N/A	N/A	N/A	N/A	
$OPTORIGDEL	IMT	N/A	N/A	IMT	N/A	N/A	N/A	IMT	N/A	N/A	
$OPTORIGTOD1	IMT	N/A	N/A	IMT	N/A	N/A	N/A	N/A	N/A	N/A	
$ORIGAM	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT	N/A	N/A	
$ORIGDDN	IMT	IMT	IMT	IMT	IMT ++	IMT ++	N/A	N/A	N/A	N/A	
$ORIGDSN	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT	N/A	N/A	
$ORIGSEC	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT	N/A	N/A	
$ORIGSHORTNAME	IMT	IMT	N/A	IMT	IMT	IMT	IMT	IMT	N/A	N/A	
$OTHRPFX	SIMTE	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
$OUTDD	SIMTE	N/A	SIMTE	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Checkin/ Cancel

Checkin Checkout/

Keywords	**Move**	**Retrieve**	**On Demand**	**Verifi-cation**	**Compile**	**for Checkout**	**Cancel Migration**	**Development**	**Compare**	**Merge**
$OUTMEM	SIMTE	N/A	SIMTE	N/A	N/A	N/A	N/A	N/A	N/A	N/A
$OUTPFX	IMT	N/A	IMT	N/A	N/A	N/A	N/A	N/A	N/A	N/A
$OUTPOST	M	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
$OWNER	M	M **	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$PHASE	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT
$RC	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT	IMT	IMT

$RELCOMPINC	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$RELCOMPSYSLIB	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$RELLKEDINC	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$RELLKEDSYSLIB	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$RELLKEDSYSLIN	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$RELOUTEXEC	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$RELOUTLISTING	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$RELOUTOBJECT	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$RELOUTOTHER	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$RELOUTSOURCE	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$RELPRECOMPSYSLIB	IMT	IMT	IMT	IMT	IMT	IMT	IMT	IMT
IMT	IMT							
$SAVEOBJECT	M	M	M	M	M	M	M	M
M	M							
$SAVELIST	M	M	M	M	M	M	M	M
M	M							
$SAVELOAD	M	M	M	M	M	M	M	M
M	M							
$SR	M	M **	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$TESTDATA	M	N/A	M	M	N/A	N/A	N/A	N/A
N/A	N/A							
$TESTNAME	M	N/A	M	M	N/A	N/A	N/A	N/A
N/A	N/A							
$TODATA	M	M	M	M	M	M	M	M
M	M							
$TODAY	SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							
$TONAME	M	M	M	M	M	M	M	M
M	M							
$USERID	N/A	IMT	N/A	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT							

$UTLISTAM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							
$UTLISTDSN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							
$UTLISTNAME	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							
$UTLISTSEC	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							
$UTOUTAM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT							
$UTOUTDSN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT							
$UTOUTNAME	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT							
$UTOUTSEC	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT							
$UTWORK1AM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT							
$UTWORK1DSN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT							
$UTWORK2AM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT							
$UTWORK2DSN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT							
$UTWORK3AM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT							
$UTWORK3DSN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT							
$UT1AM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							
$UT1DSN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							
$UT1NAME	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							
$UT1SEC	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							
$UT2AM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							
$UT2DSN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							
$UT2NAME	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT							

$UT2SEC		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IMT	IMT								
$UT3AM		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT								
$UT3DSN		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT								
$UT3NAME		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT								
$UT3SEC		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
N/A	IMT								
$VERCAT		N/A	N/A	N/A	SIMTE	N/A	N/A	N/A	N/A
N/A	N/A								
$VERDESC		N/A	N/A	N/A	SIMTE	N/A	N/A	N/A	N/A
N/A	N/A								
$VERREQUIRED		N/A	N/A	N/A	IMT	N/A	N/A	N/A	N/A
N/A	N/A								
$VERSHORTNAME		N/A	N/A	N/A	SIMTE	N/A	N/A	N/A	N/A
N/A	N/A								
$VSAMPFX		SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT								
$X00		SIMTE	IMT	SIMTE	SIMTE	IMT	IMT	IMT	IMT
IMT	IMT								

Sample Models

This topic gives you many important sample models. You must adapt these models to conform to your own site's standards.

All of these models are included on the distribution tape as members of APTMODEL.

This topic covers the following models:

- Model for Generating Assemble and Link Edit JCL
- Model for Generating COBOL Compile and Link Edit JCL
- Model for Generating Link Edit JCL
- Model to blank out $MSG
- Model to Do Nothing (support for member posting)
- Model for CA-Panexec Moves Using Protection Files
- Model for CA-Panexec Moves Not Using Protection Files
- Model for CA-Librarian Moves
- Model for CA-Panvalet Moves
- Model for PDS Moves
- Model for CA PFF Moves
- Models for Retrieve Moves
- Models for the Development Facility

Checkout and Retrieve Models

Checkout/Retrieve CA-Librarian Members

Model Name: APJCLIBR

Usage: Checkout/Retrieve members of CA-Librarian masters.

User keywords passed to the model: None.

Processing Overview: The member is copied from the origin master to the starting test level master (Retrieve processing) or to the development level master (Checkout processing), using a multi-step process involving a utility copy. Changing the member name is supported. MCD security is also supported. This is facilitated by prefacing the job with a step that calls a Librarian preprocessor, APCS5921. The module takes the MCD code supplied in the Libcode security field and add today's date to it. A member name that already exists on the starting test level master either is or is not replaced, as specified on the Retrieve panel.

During Checkout processing, a step is run to update the database to update the status of the member.

Model specification notes: You should consider adding an INCLUDE to add a JOBLIB DD statement for model APJCJOBL prior to the INCLUDE of APJCLIBR.

If you want to send a message to the user who invoked the Checkout/Retrieve, you can add an INCLUDE for either model APJMSEND, APJCSEND, or APJCMSGS after the INCLUDE of APJCLIBR. APJMSEND is the best choice because the method it uses to send the message can be controlled in the APJMLEAD model.

Checkout/Retrieve CA-Panvalet Members

Model Name: APJCPANV

Usage: Checkout/Retrieve Processing.

Purpose: Checkout/Retrieve members of CA-Panvalet libraries.

User keywords passed to the model:

LOCK

Specifies whether to lock the From Member in the From Library during the Checkout/Retrieve. The member is locked only when LOCK is specified as Y and the From Library is a normal target library and not a Backup or Back out library. The LOCK keyword must always be set to N or omitted for releases of CA-Panvalet before release 14.1.

If you are using version 14.2 of CA-Panvalet, you must alter the APJCPANV model before specifying Y for LOCK. The LOCKSTYLE keyword setting must be changed from 2 to 1. See the comments in the model regarding this.

Output: Optional JCL to first delete the member in the starting level test library, if you specified that the member is to be replaced.

JCL to copy the member to the test library using PAN#2 library-to-library transfer.

An optional control statement or job step (depending on your release of CA-Panvalet) to LOCK the member on the From Library.

Processing Overview: Status of the member in the target (Test) Library is taken from the RSTSTAT operand in the CA-Panvalet PVOPT generation macro.

A member that already exists on the Test Level Library is replaced or not, as specified on the Retrieve panel.

During Checkout processing, a step is run to update the database to update the status of the member.

Model specification notes: You should consider adding an INCLUDE to add a JOBLIB DD statement for model APJCJOBL prior to the INCLUDE of APJCLIBR.

If you want to send a message to the user who invoked the Checkout/Retrieve, you can add an INCLUDE for either model APJMSEND, APJCSEND, or APJCMSGS after the INCLUDE of APJCLIBR. APJMSEND is the best choice because the method it uses to send the message can be controlled in the APJMLEAD model.

Checkout/Retrieve PDS Members

Model Name: APJCPDS

Usage: Checkout/Retrieve Processing.

Purpose: Checkout/Retrieve members of PDS libraries.

User keywords passed to the model: None.

Output: JCL and control statements for IEBCOPY.

Processing Overview: A member is copied from the From Library to the starting level test library (Retrieve processing) or to a development level library (Checkout processing).

A member that already exists on the Test Level Library is replaced or not, as specified on the Retrieve panel.

During Checkout processing, a step is run to update the database to update the status of the member.

Model specification notes: You should consider adding an INCLUDE to add a JOBLIB DD statement for model APJCJOBL prior to the INCLUDE of APJCLIBR.

If you want to send a message to the user who invoked the Checkout/Retrieve, you can add an INCLUDE for either model APJMSEND, APJCSEND, or APJCMSGS after the INCLUDE of APJCLIBR. APJMSEND is the best choice because the method it uses to send the message can be controlled in the APJMLEAD model.

New Member Prototype

Model Name: APJCPRTO

Usage: Create a prototype member for a Checkout or Retrieve for Add.

User keywords passed to the model: None

Processing Overview: When you perform a Checkout or Retrieve for Add, a new member is added to the destination library. This model generates the contents of that member along with the control statements for IEBUPDTE (for PDS data sets), AFOLIBR (CA-Librarian), or PAN#1 (CA-Panvalet). You can customize this model to change the prototype to meet your site's standards.

Checkin Models

Checkin CA Librarian Members

Model Name: APJILIBR

Usage: Checkin or Check for Migration of CA Librarian members.

User keywords passed to the model:

LLIBR_PGM—This keyword indicates the name of the CA Librarian program name, which is usually AFOLIBR. This keyword must be set in the APJMLEAD model.

Processing Overview: The member is moved from the origin CA Librarian master to the destination unless they are the same data set. MCD security is supported using the APCS5921 utility. During the move, the status of the member is changed to the value specified in the security field for the destination master, if any. The database is then updated to reflect the member's new status.

Model specification notes: You should consider adding an INCLUDE to add a JOBLIB DD statement for model APJCJOBL prior to the INCLUDE of APJILIBR.

If you want to send a message to the user who invoked the Checkin, you can add an INCLUDE for model APJMSEND after the INCLUDE of APJILIBR.

Checkin CA Panvalet Members

Model Name: APJIPANV

Usage: Checkin or Check for Migration of CA Panvalet members.

User keywords passed to the model:

LOCK

This keyword specifies whether the destination member needs to be unlocked before it is replaced. This should correspond with the LOCK value used to Checkout the member.

Processing Overview: The member is moved from the origin CA Panvalet library to the destination unless they are the same data set. The move processes consists of unlocking the old version on the destination library (optional), deleting the old version on the destination library, transferring the new version to the destination library, and then deleting the new version from the origin library. The database is then updated to reflect the member's new status.

Model specification notes: You should consider adding an INCLUDE to add a JOBLIB DD statement for model APJCJOBL prior to the INCLUDE of APJIPANV.

If you want to send a message to the user who invoked the Checkin, you can add an INCLUDE for model APJMSEND after the INCLUDE of APJIPANV.

Checkin PDS Members

Model Name: APJIPDS

Usage: Checkin or Check for Migration of PDS members.

User keywords passed to the model: None.

Processing Overview: The member is moved from the origin PDS to the destination unless they are the same data set. The database is then updated to reflect the member's new status.

Model specification notes: You should consider adding an INCLUDE to add a JOBLIB DD statement for model APJCJOBL prior to the INCLUDE of APJIPDS.

If you want to send a message to the user who invoked the Checkin, you can add an INCLUDE for model APJMSEND after the INCLUDE of APJIPDS.

Cancel Checkout/Development Models

Cancel Checkout/Development of CA-Librarian Members

Model Name: APJCCLIB

Usage: Cancel Checkout or Development of CA-Librarian members.

User keywords passed to the model:

LLIBR_PGM

This keyword indicates the name of the CA-Librarian program name, which is usually AFOLIBR. This keyword must be set in the APJMLEAD model.

Processing Overview: The member is deleted from the library it is checked out to (Cancel Checkout) or checked in to (Cancel Development) unless that library also happens to be the same as a higher level data set. The database is then updated to reflect the member's new status.

Model specification notes: You should consider adding an INCLUDE to add a JOBLIB DD statement for model APJCJOBL prior to the INCLUDE of APJCCLIB.

If you want to send a message to the user who invoked the Checkin, you can add an INCLUDE for model APJMSEND after the INCLUDE of APJCCLIB.

Cancel Checkout/Development of CA-Panvalet Members

Model Name: APJCCPAN

Usage: Cancel Checkout or Development of CA-Panvalet members.

User keywords passed to the model: None

Processing Overview: The member is deleted from the library it is checked out to (Cancel Checkout) or checked in to (Cancel Development) unless that library also happens to be the same as a higher level data set. The database is then updated to reflect the member's new status.

Model specification notes: You should consider adding an INCLUDE to add a JOBLIB DD statement for model APJCJOBL prior to the INCLUDE of APJCCPAN.

If you want to send a message to the user who invoked the Checkin, you can add an INCLUDE for model APJMSEND after the INCLUDE of APJCCPAN.

Cancel Checkout/Development of PDS Members

Model Name: APJCCPDS

Usage: Cancel Checkout or Development of PDS members.

User keywords passed to the model: None

Processing Overview: The member is deleted from the library it is checked out to (Cancel Checkout) or checked in to (Cancel Development) using the CA-PanAPT APAS5900 utility unless that library also happens to be the same as a higher level data set. The database is then updated to reflect the member's new status.

Model specification notes: You should consider adding an INCLUDE to add a JOBLIB DD statement for model APJCJOBL prior to the INCLUDE of APJCCPDS.

If you want to send a message to the user who invoked the Checkin, you can add an INCLUDE for model APJMSEND after the INCLUDE of APJCCPDS.

Miscellaneous Development Facility and Retrieve Models

Reporting Outcome of Development Facility Action or Retrieve

Model Name: APJMSEND

Usage: Development Facility. Retrieve Processing.

Purpose: Generate JCL to report the outcome of a Development Facility action or Retrieve job, based on condition codes.

User keywords passed to the model:

COND_PFX

Contains the prefix, if any, for the CONDnn keywords.

<COND_PFX>COND1 through <COND_PFX>COND8

Contains the condition code values to test for success. These are passed along to the APJMCOND model.

GOODMSG

Contains the text of the message to send to indicate success. This keyword cannot contain any apostrophes and must be less than 60 characters long.

FAILMSG

Contains the text of the message to send to indicate failure. This keyword cannot contain any apostrophes and must be less than 60 characters long.

SENDMETHOD

Indicates how the message is sent. The valid values are SEND for the TSO SEND command and APAS0075 for the CA-PanAPT utility.

The APAS0075 utility uses the TPUT macro and runs faster than the TSO SEND command method. However, it can experience S522 abends if more than a screen's worth of messages are sent and the user does not clear the screen in a reasonable amount of time (there are comments regarding this in the model).

The TSO SEND command has the ability to queue up messages if the user is not logged on, until the user logs on again.

The default is SEND. You can change the default in the APJMLEAD model if you want.

Processing Overview: If the conditions pass, the GOODMSG text is sent to the user; otherwise, the FAILMSG text is sent to the user.

Reporting the Status of a Retrieve

Model name: APJCMSGS

Usage: Retrieve Processing.

Purpose: Generate JCL to report the status of a Retrieve based on return codes.

User keywords passed to the model:

EMLT

Set before invoking the model in the Retrieve Model Specifications of the Library Code definition or in a preceding model. The values of EMLT and $MEMBERCOUNT are used to generate a step name in a COND test.

For example, model APJCPDS is distributed with a stepname PDS. APJCPDS sets EMLT to PDS. Therefore, the following Retrieve Model Specification sends status messages back to the user who is Retrieving a member:

```
INC APJCPDS; INC APJCMSGS
```

CERTAIN

Set before invoking the model by the preceding Retrieve model. If CERTAIN=Y, it indicates that a return code of zero from the step named by EMLT guarantees that the Retrieve was successful.

If CERTAIN=N, it indicates that a return code of zero is not conclusive. In particular, IEBCOPY returns zero if the member was not copied because replace was not allowed.

Output: JCL and control statements to send messages to users. The program issues a TPUT to the user's user ID.

Processing Overview: APJCSEND is similar to APJCMSGS in function. APJCMSGS is faster, but does NOT deliver messages to the submitting user unless the user stays logged on.

Sending Messages During Retrieve

Model Name: APJCSEND

Usage: Retrieve Processing.

Purpose: Generate JCL to report the status of a Retrieve, based on return codes.

User keywords passed to the model:

EMLT

Set before invoking the model in the Retrieve Model Specifications of the Library Code definition or in a preceding model. The values of EMLT and $MEMBERCOUNT are used to generate a step name in a COND test.

For example, model APJCPDS is distributed with a stepname PDS. APJCPDS sets EMLT to PDS. Therefore, the following Retrieve Model Specification sends status messages back to the user who is retrieving a member:

```
INC APJCPDS; INC APJCMSGS
```

CERTAIN

Set before invoking the model by the preceding Retrieve model. If CERTAIN=Y, it indicates that a return code of zero from the step, named by EMLT, guarantees that the Retrieve was successful.

If CERTAIN=N, it indicates that a return code of zero is not conclusive. In particular, IEBCOPY returns zero if the member was not copied because replace was not allowed.

Output: JCL and SEND commands for IKJEFT01.

Processing Overview: APJCSEND is similar to APJCMSGS in function. APJCSEND is slower, but delivers messages to the submitting user even after logoff.

Retrieve Models

Generating a JOBLIB During Retrieve

Model name: APJCJOBL

Usage: Retrieve Processing. Development Facility.

Purpose: Generate a JOBLIB DD statement.

User keywords passed to the model: None.

Output: JOBLIB DD statement generated in the Retrieve JCL.

Processing Overview: Generating a JOBLIB concatenation is an alternative to generating STEPLIB DD statements in individual models or to including required libraries in the LINKLST.

The JOBLIB is generated only for the first member processed in preparing the job.

This model is an example of the use of the $MEMBERCOUNT keyword to identify the first member.

Retrieving CA-Panexec Members

Model Name: APJCPEX

Usage: Retrieve Processing.

Purpose: Retrieve members of CA-Panexec libraries.

User keywords passed to the model:

SELGRP

Selected group to be used. The group specified in the Library Code can be overridden by the From User data and To User data fields for the member on the Retrieve Processing Options panel. From User data applies to the sending library and To User data applies to the receiving library.

SELTYPE

Selected type to be used.

SELSTAT

Selected status to be used.

SELMODE

Selected mode to be used.

Specify all of the above parameters in the Library Code definition for every Library Code that uses the model. For example:

```
SELGRP='PAYROLL';SELTYPE='CNTL'
SELGRP=' ';SELSTAT='T';SELMODE='A';
 INCLUDE APJCPEX; INC APJCMSGS
```

$FROMDATA

The From Data field on the Retrieve panel, if it is specified, overrides the default CA-Panexec group specified in SELGRP for the sending library.

$TODATA

The To Data field on the Retrieve panel, if it is specified, overrides the default CA-Panexec group specified in SELGRP for the receiving library.

Output: JCL and control statements for executing CA-Panexec %COPY

Processing Overview: A member is copied from the From Library to the starting level test library.

A member that already exists on the Test Level Library is replaced or not, as specified on the Retrieve panel.

The values of user keywords EMLT and CERTAIN are set to PEX and Y, respectively, for use by models APJCMSGS or APJCSEND.

Retrieving CA PFF Members

Model Name: APJCPFF

Usage: Retrieve Processing.

Purpose: Retrieve members of CA PFF libraries.

User keywords passed to the model:

ALIAS

Specifies how CA PFF processes aliases on a PDS. Valid values are Y and N.

Output: JCL and control statements to copy a single member from a source CA-PFF library to a target CA-PFF library.

Processing Overview: A member is copied from the From Library to the starting level test library.

A member that already exists on the Test Level Library is replaced or not, as specified on the Retrieve panel.

The value of user keywords EMLT and CERTAIN are set to CA-PFF and Y, respectively, for use by model APJCMSGS or APJCSEND.

Retrieving CA-Telon Members

Retrieving CA-Telon Members from a PDS to a TDF

Model Name: APJCPD2T

Usage: Retrieve CA-Telon members from a PDS to a TDF via the CA-Telon Import facility.

User keywords passed to the model: None.

Processing Overview: The member is imported from the origin PDS to the starting level test TDF.

Changing the member name is not supported. The replace value specified on the Retrieve panel is ignored.

The EMLT and CERTAIN user keywords are not set in this model. Do not include APJCMSGS or APJCSEND in the Retrieve model specifications.

Retrieving CA-Telon Members from a CA-Panvalet library to a TDF

Model Name: APJCPV2T

Usage: Retrieve CA-Telon members from a CA-Panvalet library to a TDF via the CA-Telon Import facility.

User keywords passed to the model:

LOCK

Specifies whether to lock the From Member in the From Library during the Retrieve. The member is locked only when LOCK is specified as Y and the From Library is a normal target library, and not a Backup or Back out library. The LOCK keyword must always be set to N or omitted for releases of CA-Panvalet before release 14.1.

User keywords used within the model that require customization:

PANLOAD

The data set name of your CA-Panvalet load library.

TELONQUAL

The high-level qualifier for CA-Telon data sets. This is specified for the TLNLOAD parameter of the TLNUMPAN proc.

Processing Overview: The member is imported from the origin CA-Panvalet library to the starting level test TDF.

Changing the member name is not supported. The replace value specified on the Retrieve panel is ignored.

If LOCK is specified as Y and the From Library is not a Backup or a Back out library for the From Level, the origin CA-Panvalet member is locked. Do not specify **LOCK = 'Y'** if your release of CA-Panvalet is earlier than 14.1.

The EMLT and CERTAIN user keywords are not set in this model. Do not include APJCMSGS or APJCSEND in the Retrieve model specifications.

Retrieving CA-Telon Members from TDF to a TDF

Model Name: APJCT2TD

Usage: Retrieve CA-Telon members from a TDF to a TDF via a CA-Telon export and import.

User keywords passed to the model: None.

User keywords used within the model that require customization:

SRCPDS

The data set name of a PDS to use during the export/import.

TELONQUAL

The high-level qualifier for CA-Telon data sets. This is specified for the TLNLOAD parameter of the TLNUMPAN proc.

Models invoked by this model:

APJCJOBL

Include the CA-PanAPT load libraries in the JOBLIB DD. This is because the CA-PanAPT APAS5900 utility must be available to scratch the intermediate member from the SRCPDS data set.

Processing Overview: The member is exported from the origin TDF to the SRCPDS partitioned data set. Then the member is imported from SRCPDS to the starting level test TDF. Finally, the member is scratched from the SRCPDS data set.

Changing the member name is not supported. The replace value specified on the Retrieve panel is ignored.

The EMLT and CERTAIN user keywords are not set in this model. Do not include APJCMSGS or APJCSEND in the Retrieve model specifications.

Compile Models

The compile models distributed with CA-PanAPT are used by both the Development Facility and turnover.

When compile models are used by turnover, the JCL they create is run after the moves. This is called external processing. They can be used in conjunction with move models, performing compiles after members are moved; or they can be used without move models, performing compiles on members that are not moved but are affected by members that did move, such as copybooks.

Assembly JCL

Model Name: APJMASMB

Usage: Daily Processing Cycle. Development Facility.

Purpose: Build JCL to assemble and link edit a member.

User keywords passed to the model:

ASM_PGM

Lets you override the default for the assembler to be used. If set, the value must be either ASMA90 (the high-level assembler) or IEV90 (assembler H). This value would typically be set in the Library Code leading model specifications.

GASM_PGM

This is the default assembler to be used. It must be set in model APJMLEAD to either ASMA90 (the high-level assembler) or IEV90 (assembler H).

ASM_STD_COMPOPTS

This keyword lets you specify standard compiler options that are used for every assembly in addition to those specified in Inventory. If all of your Library Codes use the same assembler, this can be specified in the APJMLEAD model; otherwise, this should only be set in the Library Code, preferably in the leading model specifications. If this keyword is not set, it is given a default value in the model.

ASM_COMPOPTS

This keyword lets you define default compiler options to be used if none are defined in Inventory. If any are defined in Inventory, the model sets this keyword to the Inventory value.

ASM_SYSLIB_DSNnn

These keywords let you define additional SYSLIB data sets for the assembly. These data sets follow any SYSLIB data sets generated for Related Compile SYSLIB Library Codes that are defined in the current Library Code.

A list must be created using these keywords. Specify the first data set in the list in keyword ASM_SYSLIB_DSN01, the second in ASM_SYSLIB_DSN02, and so forth. The total number of data sets must be specified as a two-digit integer in keyword ASM_SYSLIB_DSN00. For example, if the last data set was specified in keyword ASM_SYSLIB_DSN07, set ASM_SYSLIB_DSN00 to '07'.

If any of the ASM_SYSLIB_DSNnn data sets are CA-Panvalet or CA-Librarian, set the corresponding ASM_SYSLIB_AMnn keyword to 'PV' (CA-Panvalet), 'L' (CA-Librarian), or 'LW' (CA-Librarian wide record). Otherwise 'PO' for PDS is assumed.

ASM_SYSLIB_MODEL

If the ASM_SYSLIB_DSNnn keywords do not give you enough flexibility in generating a SYSLIB DD for the assembly, you can code your own model to generate the SYSLIB DD statement.

Specify the name of the model in the ASM_SYSLIB_MODEL keyword, preferably in the Library Code leading model specifications. The APJMASMB model then calls this model instead of the APJMSLIB model to create the DD. Your model can either construct the DD statement or change the values of keywords used to pass information along to APJMSLIB and then invoke APJMSLIB. Model USERSLIB was distributed as an example of how to do this.

LINK_PGM

Lets you override the default for the linkage editor to be used. If set, the value must be either IEWL or HEWLKED (the name given to the old linkage editor when you have the binder). This value would typically be set in the Library Code leading model specifications.

GLINK_PGM

This is the default linkage editor to be used. It must be set in model APJMLEAD to either IEWL or HEWLKED. HEWLKED is the name given to the old linkage editor when you have the binder.

STD_LINKOPTS

This keyword lets you specify standard linkage editor options that are used for every link edit in addition to those specified in Inventory. This can be specified in the APJMLEAD model. If this keyword is not set, it is given a default value in the model.

LINKOPTS

This keyword lets you define default linkage editor options to be used if none are defined in Inventory. If any are defined in Inventory, the model sets this keyword to the Inventory value.

LINK_SYSLIB_DSNnn

These keywords are just like the ASM_SYSLIB_DSNnn keywords except they are used when generating a SYSLIB DD for the linkage editor.

These keywords let you define additional SYSLIB data sets for the link edit. These data sets follow any SYSLIB data sets generated for Related Link Edit SYSLIB Library Codes that are defined in the current Library Code.

A list must be created using these keywords. The first data set in the list must be specified in keyword LINK_SYSLIB_DSN01, the second in LINK_SYSLIB_DSN02, and so forth. The total number of data sets must be specified as a two-digit integer in keyword LINK_SYSLIB_DSN00. For example, if the last data set was specified in keyword LINK_SYSLIB_DSN12, set LINK_SYSLIB_DSN00 to '12'.

If any of the LINK_SYSLIB_DSNnn data sets are CA-Panexec, set the corresponding LINK_SYSLIB_Amnn keyword to 'PX'. Otherwise 'PO' for PDS is assumed.

LINK_SYSLIB_MODEL

If the LINK_SYSLIB_DSNnn keywords do not give you enough flexibility in generating a SYSLIB DD for the link edit, you can code your own model to generate the SYSLIB DD statement.

Specify the name of the model in the LINK_SYSLIB_MODEL keyword, preferably in the Library Code leading model specifications. The APJMASMB model then calls this model instead of the APJMSLIB model to create the DD. Your model can either construct the DD statement or change the values of keywords used to pass information along to APJMSLIB and then invoke APJMSLIB. Model USERSLIB was distributed as an example of how to do this.

LINK_PXCONCAT_GRPnn

These keywords are similar to the LINK_SYSLIB_DSNnn keywords. They are used during link edits involving CA-Panexec libraries to define CA-Panexec groups to be searched in addition to the groups generated for Related Link edit SYSLIB Library Codes that are defined in the current Library Code.

A list must be created using these keywords. The first group in the list must be specified in keyword LINK_PXCONCAT_GRP01, the second in LINK_PXCONCAT_GRP02, and so forth. The total number of groups must be specified as a two-digit integer in keyword LINK_PXCONCAT_GRP00. For example, if the last group was specified in keyword LINK_PXCONCAT_GRP12, set LINK_PXCONCAT_GRP00 to '12'.

LINK_PXCONCAT_MODEL

If the LINK_PXCONCAT_GRPnn keywords do not give you enough flexibility in generating CA-Panexec %CONCAT statements, you can code your own model to generate them.

Specify the name of the model in the LINK_PXCONCAT_MODEL keyword, preferably in the Library Code leading model specifications. The APJMASMB model then calls this model instead of the APJMSLIB model to create the statements. Your model can either construct the statements or change the values of keywords used to pass information along to APJMSLIB and then invoke APJMSLIB. Model USERSLIB was distributed as an example of how to do this.

CICS

If you want to include a CICS pre-compile in the assembly process, set this keyword to Y, preferably in the Library Code leading model specifications.

GJCLLIST

This keyword controls the SYSOUT class for the JCL listings. It is set in the APJMLEAD model.

GPANESRL

This keyword contains the data set name of the CA-Panexec PANESRL data set, if any. This is used when saving object, executable, or listings on CA-Panexec libraries.

GSYSTEMP

This keyword specifies the OS/390 unit for work files. It is set in the APJMLEAD model.

Processing Overview: Complete JCL is built to assemble and/or link edit the member. Assembly and link edit parms can be specified in Inventory and in the model specifications. When assembling, a CICS pre-compile is supported.

When run on behalf of the Development Facility, the JCL is submitted directly to OS/390 for execution. Depending upon the command you used to initiate the process, the member is either assembled, assembled and linked, or just link edited (assumes you performed the assembly separately). You can have messages sent to you regarding the outcome of the assembly and link edit by including the APJMSEND model after the APJMASMB model in the model specifications. You can also have a JOBLIB DD statement generated if necessary by including the APJCJOBL model before the APJMASMB model.

When run on behalf of turnover, the member is assembled. If the Library Code and Inventory support it, the member is also link edited. You have a choice of whether to lump all assemblies and other compiles into a single job or to run each as separate jobs. This is controlled in the APJMLEAD model using the GCOMPILES_GROUPED user keyword. The default is to run all assemblies and compiles as separate jobs.

In both cases, the assembly and link edit listings are either printed or saved to a CA-Panexec Library depending on whether there is a related listing Library Code. If the listings are to be saved but the assembly or link edit fail, the listings are printed rather than saved.

COBOL Compile JCL

Model Name: APJMCOMP

Usage: Daily Processing Cycle. Development Facility.

Purpose: Build JCL to compile and link edit a COBOL program.

User keywords passed to the model:

COB_PGM

Lets you override the default for the compiler to be used. If set, the value must be IGYCRCTL (LE/390). This value would typically be set in the Library Code leading model specifications.

GCOB_PGM

This is the default compiler to be used. It must be set in model APJMLEAD to IGYCRCTL (LE/390 COBOL).

COB_STD_COMPOPTS

This keyword lets you specify standard compiler options that are used for every compile in addition to those specified in Inventory. If all of your Library Codes use the same compiler, this can be specified in the APJMLEAD model; otherwise, this should only be set in the Library Code, preferably in the leading model specifications. If this keyword is not set, it is given a default value in the model.

COB_COMPOPTS

This keyword lets you define default compiler options to be used if none are defined in Inventory. If any are defined in Inventory, the model sets this keyword to the Inventory value.

COB_STD_CICSOPTS

This keyword lets you specify standard CICS precompiler options that are used for every CICS compile in addition to those specified in Inventory. If all of your COBOL Library Codes use the same compiler, this can be specified in the APJMLEAD model; otherwise, this should only be set in the Library Code, preferably in the leading model specifications. If this keyword is not set, it is given a default value in the model.

COB_CICSOPTS

This keyword lets you define default CICS pre-compiler options to be used if none are defined in Inventory. If any are defined in Inventory, the model sets this keyword to the Inventory value.

COB_SYSLIB_DSNnn

These keywords let you define additional SYSLIB data sets for the compile. These data sets follow any SYSLIB data sets generated for Related Compile SYSLIB Library Codes that are defined in the current Library Code.

A list must be created using these keywords. The first data set in the list must be specified in keyword COB_SYSLIB_DSN01, the second in COB_SYSLIB_DSN02, and so forth. The total number of data sets must be specified as a two-digit integer in keyword COB_SYSLIB_DSN00. For example, if the last data set was specified in keyword COB_SYSLIB_DSN07, set COB_SYSLIB_DSN00 to '07'.

If any of the COB_SYSLIB_DSNnn data sets are CA-Panvalet or CA-Librarian, set the corresponding COB_SYSLIB_AMnn keyword to 'PV' (CA-Panvalet), 'L' (CA-Librarian), or 'LW' (CA-Librarian wide record). Otherwise 'PO' for PDS is assumed.

COB_SYSLIB_MODEL

If the COB_SYSLIB_DSNnn keywords do not give you enough flexibility in generating a SYSLIB DD for the compile, you can code your own model to generate the SYSLIB DD statement.

Specify the name of the model in the COB_SYSLIB_MODEL keyword, preferably in the Library Code leading model specifications. The APJMCOMP model then calls this model instead of the APJMSLIB model to create the DD. Your model can either construct the DD statement or change the values of keywords used to pass information along to APJMSLIB and then invoke APJMSLIB. Model USERSLIB was distributed as an example of how to do this.

LINK_PGM

Lets you override the default for the linkage editor to be used. If set, the value must be either IEWL or HEWLKED (the name given to the old linkage editor when you have the binder). This value would typically be set in the Library Code leading model specifications.

GLINK_PGM

This is the default linkage editor to be used. It must be set in model APJMLEAD to either IEWL or HEWLKED. HEWLKED is the name given to the old linkage editor when you have the binder.

STD_LINKOPTS

This keyword lets you specify standard linkage editor options that are used for every link edit in addition to those specified in Inventory. This can be specified in the APJMLEAD model. If this keyword is not set, it is given a default value in the model.

LINKOPTS

This keyword lets you define default linkage editor options to be used if none are defined in Inventory. If any are defined in Inventory, the model sets this keyword to the Inventory value.

LINK_SYSLIB_DSNnn

These keywords are just like the COB_SYSLIB_DSNnn keywords except they are used when generating a SYSLIB DD for the linkage editor.

These keywords let you define additional SYSLIB data sets for the link edit. These data sets follow any SYSLIB data sets generated for Related Link Edit SYSLIB Library Codes that are defined in the current Library Code.

A list must be created using these keywords. The first data set in the list must be specified in keyword LINK_SYSLIB_DSN01, the second in LINK_SYSLIB_DSN02, and so forth. The total number of data sets must be specified as a two-digit integer in keyword LINK_SYSLIB_DSN00. For example, if the last data set was specified in keyword LINK_SYSLIB_DSN12, set LINK_SYSLIB_DSN00 to '12'.

If any of the LINK_SYSLIB_DSNnn data sets are CA-Panexec, set the corresponding LINK_SYSLIB_AMnn keyword to 'PX'. Otherwise 'PO' for PDS is assumed.

LINK_SYSLIB_MODEL

If the LINK_SYSLIB_DSNnn keywords do not give you enough flexibility in generating a SYSLIB DD for the link edit, you can code your own model to generate the SYSLIB DD statement.

Specify the name of the model in the LINK_SYSLIB_MODEL keyword, preferably in the Library Code leading model specifications. The APJMCOMP model then calls this model instead of the APJMSLIB model to create the DD. Your model can either construct the DD statement or change the values of keywords used to pass information along to APJMSLIB and then invoke APJMSLIB. Model USERSLIB was distributed as an example of how to do this.

LINK_PXCONCAT_GRPnn

These keywords are similar to the LINK_SYSLIB_DSNnn keywords. They are used during link edits involving CA-Panexec libraries to define CA-Panexec groups to be searched in addition to the groups generated for Related Link Edit SYSLIB Library Codes that are defined in the current Library Code.

A list must be created using these keywords. The first group in the list must be specified in keyword LINK_PXCONCAT_GRP01, the second in LINK_PXCONCAT_GRP02, and so forth. The total number of groups must be specified as a two-digit integer in keyword LINK_PXCONCAT_GRP00. For example, if the last group was specified in keyword LINK_PXCONCAT_GRP12, set LINK_PXCONCAT_GRP00 to '12'.

LINK_PXCONCAT_MODEL

If the LINK_PXCONCAT_GRPnn keywords do not give you enough flexibility in generating CA-Panexec %CONCAT statements, you can code your own model to generate them.

Specify the name of the model in the LINK_PXCONCAT_MODEL keyword, preferably in the Library Code leading model specifications. The APJMCOMP model then calls this model instead of the APJMSLIB model to create the statements. Your model can either construct the statements or change the values of keywords used to pass information along to APJMSLIB and then invoke APJMSLIB. Model USERSLIB was distributed as an example of how to do this.

CICS

If you want to include a CICS pre-compile in the compile process, set this keyword to Y, preferably in the Library Code leading model specifications.

GJCLLIST

This keyword controls the SYSOUT class for the JCL listings. It is set in the APJMLEAD model.

GPANESRL

This keyword contains the data set name of the CA-Panexec PANESRL data set, if any. This is used when saving object, executable, or listings on CA-Panexec libraries.

GSYSTEMP

This keyword specifies the OS/390 unit for work files. It is set in the APJMLEAD model.

Processing Overview: Complete JCL is built to compile and/or link edit the COBOL member. Compile and link edit parms can be specified in Inventory and in the model specifications. When compiling, a CICS pre-compile is supported.

When run on behalf of the Development Facility, the JCL is submitted directly to OS/390 for execution. Depending upon the command you used to initiate the process, the member is either compiled, compiled and linked, or just link edited (assumes you performed the compile separately). You can have messages sent to you regarding the outcome of the compile and link edit by including the APJMSEND model after the APJMCOMP model in the model specifications. You can also have a JOBLIB DD statement generated if necessary by including the APJCJOBL model before the APJMCOMP model.

When run on behalf of turnover, the member is compiled. If the Library Code and Inventory support it, the member is also link edited. You have a choice of whether to lump all compiles into a single job or to run each as separate jobs. This is controlled in the APJMLEAD model using the GCOMPILES_GROUPED user keyword. The default is to run all assemblies and compiles as separate jobs.

In both cases, the compile and link edit listings are either printed or saved to a CA-Panexec Library depending on whether there is a related listing Library Code. If the listings are to be saved but the compile or link edit fail, the listings are printed rather than saved.

Link Edit JCL

Model Name: APJMLINK

Usage: Daily Processing Cycle. Development Facility.

Purpose: Build JCL to link edit a program.

User keywords passed to the model:

LINK_PGM—Lets you override the default for the linkage editor to be used. If set, the value must be either IEWL or HEWLKED (the name given to the old linkage editor when you have the binder). This value would typically be set in the Library Code leading model specifications.

GLINK_PGM—This is the default linkage editor to be used. It must be set in model APJMLEAD to either IEWL or HEWLKED. HEWLKED is the name given to the old linkage editor when you have the binder.

STD_LINKOPTS—This keyword lets you specify standard linkage editor options that are used for every link edit in addition to those specified in Inventory. This can be specified in the APJMLEAD model. If this keyword is not set, it is given a default value in the model.

LINKOPTS—This keyword lets you define default linkage editor options to be used if none are defined in Inventory. If any are defined in Inventory, the model sets this keyword to the Inventory value.

LINK_SYSLIB_DSNnn—These keywords let you define additional SYSLIB data sets for the link edit. These data sets follow any SYSLIB data sets generated for Related Link Edit SYSLIB Library Codes that are defined in the current Library Code.

A list must be created using these keywords. The first data set in the list must be specified in keyword LINK_SYSLIB_DSN01, the second in LINK_SYSLIB_DSN02, and so forth. The total number of data sets must be specified as a two-digit integer in keyword LINK_SYSLIB_DSN00. For example, if the last data set was specified in keyword LINK_SYSLIB_DSN12, set LINK_SYSLIB_DSN00 to '12'.

If any of the LINK_SYSLIB_DSNnn data sets are CA-Panexec, set the corresponding LINK_SYSLIB_Amnn keyword to 'PX'. Otherwise 'PO' for PDS is assumed.

LINK_SYSLIB_MODEL—If the LINK_SYSLIB_DSNnn keywords do not give you enough flexibility in generating a SYSLIB DD for the link edit, you can code your own model to generate the SYSLIB DD statement.

Specify the name of the model in the LINK_SYSLIB_MODEL keyword, preferably in the Library Code leading model specifications. The APJMLINK model then calls this model instead of the APJMSLIB model to create the DD. Your model can either construct the DD statement or change the values of keywords used to pass information along to APJMSLIB and then invoke APJMSLIB. Model USERSLIB was distributed as an example of how to do this.

LINK_PXCONCAT_GRPnn—These keywords are similar to the LINK_SYSLIB_DSNnn keywords. They are used during link edits involving CA-Panexec libraries to define CA-Panexec groups to be searched in addition to the groups generated for Related Link edit SYSLIB Library Codes that are defined in the current Library Code.

A list must be created using these keywords. The first group in the list must be specified in keyword LINK_PXCONCAT_GRP01, the second in LINK_PXCONCAT_GRP02, and so forth. The total number of groups must be specified as a two-digit integer in keyword LINK_PXCONCAT_GRP00. For example, if the last group was specified in keyword LINK_PXCONCAT_GRP12, set LINK_PXCONCAT_GRP00 to '12'.

LINK_PXCONCAT_MODEL—If the LINK_PXCONCAT_GRPnn keywords do not give you enough flexibility in generating CA-Panexec %CONCAT statements, you can code your own model to generate them.

Specify the name of the model in the LINK_PXCONCAT_MODEL keyword, preferably in the Library Code leading model specifications. The APJMLINK model then calls this model instead of the APJMSLIB model to create the statements. Your model can either construct the statements or change the values of keywords used to pass information along to APJMSLIB and then invoke APJMSLIB. Model USERSLIB was distributed as an example of how to do this.

GJCLLIST—This keyword controls the SYSOUT class for the JCL listings. It is set in the APJMLEAD model.

GPANESRL—This keyword contains the data set name of the CA-Panexec PANESRL data set, if any. This is used when saving object, executable, or listings on CA-Panexec libraries.

GSYSTEMP—This keyword specifies the OS/390 unit for work files. It is set in the APJMLEAD model.

Processing Overview: Complete JCL is built to link edit the link deck member. Link edit parms can be specified in Inventory and the model specifications.

When run on behalf of the Development Facility, the JCL is submitted directly to OS/390 for execution. You can have messages sent to you regarding the outcome of the link edit by including the APJMSEND model after the APJMLINK model in the model specifications. You can also have a JOBLIB DD statement generated if necessary by including the APJCJOBL model before the APJMLINK model.

When run on behalf of turnover, the JCL is constructed in the APTMDLO data set. The APJMLEAD model coordinates the order of processing so that the link edit JCL follows any assembly or compile JCL. You have a choice of whether to lump all link edits into a single job or to run each as separate jobs. This is controlled in the APJMLEAD model using the GLINKS_GROUPED user keyword. The default is to run all link edits as separate jobs.

In both cases, the link edit listings are either printed or saved to a CA-Panexec library depending on whether there is a related listing Library Code. If the listings are to be saved but the link edit fails, the listings are printed rather than saved.

Generating SYSLIB DD Statements and Concatenating CA Panvalet and CA Librarian Data Sets

Model name: APJMSLIB

Usage: Daily Processing Cycle. Development Facility.

Purpose: Generates SYSLIB DD statements for compiles and link edits.

Generates CA-Panvalet and CA-Librarian concatenated library DD statements.

Generates CA-Panexec %CONCAT statements for link edits.

User keywords passed to the model:

SYSLIB_LIBLIST—This keyword contains the list of Library Codes used to determine the data sets (or CA-Panexec groups) to be included. This keyword can be set to one of the related Library Code keywords, such as $RELCOMPSYSLIB, and it can also be manually constructed.

Each Library Code name in the list must be formatted suitable for $ALTLIBCFORM to be set to it. Each Library Code name in the list must be padded on the right with spaces to occupy 9 positions if other Library Codes follow them. That is, if you specified four Library Codes, the first would start at position 1, the second at position 10, the third at position 19, and the fourth at position 28. You can specify as many as 8 Library Codes in the list.

Upon exit, this keyword is cleared so that it does not carry on to the next invocation of APJMSLIB.

SYSLIB_DDNAME—This keyword contains the DD name to be used for a PDS concatenation. Normally this should be set to SYSLIB. However, if you are generating additional DDs for the linkage editor, this should be set prior to each invocation of APJMSLIB to the linkage editor DD name. If you do not set this keyword, it is temporarily set to 'SYSLIB'.

Upon exit, this keyword is cleared so that it does not carry on to the next invocation of APJMSLIB.

SYSLIB_DDPFX—If you need your SYSLIB DD name to be prefixed with a procedure step name, such as //COMP.SYSLIB, specify the procedure step name (COMP for the example given) in this keyword.

Upon exit, this keyword is cleared so that it does not carry on to the next invocation of APJMSLIB.

APJMSLIX_PFX—If you specified any extra SYSLIB data sets in addition to those defined by the SYSLIB_LIBLIST keyword, the prefix for the keywords defining those data sets is specified in this keyword. For example, if you needed some additional keywords for an assembly, you might define them as follows:

```
ASM_SYSLIB_DSN01 = 'SYS1.MACLIB'
ASM_SYSLIB_DSN02 = 'SYS1.AMODGEN'
ASM_SYSLIB_DSN00 = '02'
```

To inform APJMSLIB that the extra data set keywords begin with ASM_SYSLIB, you would specify:

```
APJMSLIX_PFX = 'ASM_SYSLIB
```

Upon exit, this keyword is cleared so that it does not carry on to the next invocation of APJMSLIB.

<APJMSLIX_PFX>_DSNnn—These keywords let you define additional SYSLIB data sets. These data sets follow any SYSLIB data sets generated for SYSLIB_LIBLIST Library Codes.

A list must be created using these keywords. The first data set in the list must be specified in keyword <APJMSLIX_PFX>_DSN01, the second in <APJMSLIX_PFX>_DSN02, and so forth. The total number of data sets must be specified as a two-digit integer in keyword <APJMSLIX_PFX>_DSN00. For example, if the last data set was specified in keyword <APJMSLIX_PFX>_DSN07, set <APJMSLIX_PFX>_DSN00 to '07'.

If any of the <APJMSLIX_PFX>_DSNnn data sets are CA-Panvalet or CA-Librarian, set the corresponding or<APJMSLIX_PFX>_AMnn keyword to 'PV' (CA-Panvalet), 'L' (CA-Librarian), 'LW' (CA-Librarian wide record). Otherwise 'PO' for PDS is assumed.

If APJMSLIX_PFX has already been set, you can specify these keywords as either <APJMSLIX_PFX>_DSNnn or with the APJMSLIX_PFX value substituted. For example, if APJMSLIX_PFX is set to ASM_SYSLIB, the DSN00 keyword can be specified either as <APJMSLIX_PFX>_DSN00 or ASM_SYSLIB_DSN00.

Upon exit, the <APJMSLIX_PFX>_DSN00 keyword is cleared so that it does not carry on to the next invocation of APJMSLIB.

<APJMSLIX_PFX>_AMnn—This keyword is covered in the description of <APJMSLIX_PFX>_DSNnn.

<APJMSLIX_PFX>_GRPnn—If you are generating CA-Panexec %CONCAT statements, you can specify additional CA-Panexec group names to be concatenation using these keywords. These groups follow any generated for SYSLIB_LIBLIST Library Codes.

A list must be created using these keywords. The first group in the list must be specified in keyword <APJMSLIX_PFX>_GRP01, the second in <APJMSLIX_PFX>_GRP02, and so forth. The total number of groups must be specified as a two-digit integer in keyword <APJMSLIX_PFX>_GRP00. For example, if the last group was specified in keyword <APJMSLIX_PFX>_GRP07, set <APJMSLIX_PFX>_GRP00 to '07'.

If APJMSLIX_PFX has already been set, you can specify these keywords as either <APJMSLIX_PFX>_GRPnn or with the APJMSLIX_PFX value substituted. For example, if APJMSLIX_PFX is set to LINK_SYSLIB, the GRP00 keyword can be specified either as <APJMSLIX_PFX>_GRP00 or LINK_SYSLIB_GRP00.

Upon exit, the <APJMSLIX_PFX>_GRP00 keyword is cleared so that it does not carry on to the next invocation of APJMSLIB.

PXCONCAT—If you are invoking APJMSLIB to generate CA Panexec %CONCAT statements, set this keyword to Y. A %CONCAT is generated for each Library Code in the SYSLIB_LIBLIST keyword with a group name matching the Library Code name without the slash (/).

Upon exit, this keyword is cleared so that it does not carry on to the next invocation of APJMSLIB.

Extracting Source from CA Panvalet, CA Librarian, and Partitioned Data Sets

Model name: APJMEXTR

Usage: Daily Processing Cycle. Development Facility.

Purpose: Extract a source member from a concatenation of libraries to a sequential data set for a compile or link edit.

User keywords passed to the model:

APJMEXTR_INCLIST—This keyword contains the list of Library Codes used to determine the data sets to be included for CA Panvalet ++INCLUDE or CA-Librarian -INC statement expansion. This keyword can be set to one of the related Library Code keywords, such as $RELCOMPINC or $RELLKEDINC, and it can also be manually constructed.

Each Library Code name in the list must be formatted suitable for $ALTLIBCFORM to be set to it. Each Library Code name in the list must be padded on the right with spaces to occupy 9 positions if other Library Codes follow them. That is, if you specified four Library Codes, the first would start at position 1, the second at position 10, the third at position 19, and the fourth at position 28. You can specify as many as 8 Library Codes in the list.

For CA-Panvalet data sets, take care that the result does not cause more than 10 data sets to be included because CA-Panvalet cannot concatenate more than that.

APJMINCP_DSNnn—These keywords let you define additional CA Panvalet data sets for extracting ++INCLUDE members. These data sets follow any data sets generated on behalf of the APJMEXTR_INCLIST keyword.

A list must be created using these keywords. The first data set in the list must be specified in keyword APJMINCP_DSN01, the second in APJMINCP_DSN02, and so forth. The total number of data sets must be specified as a two-digit integer in keyword APJMINCP_DSN00. For example, if the last data set was specified in keyword APJMINCP_DSN07, set APJMINCP_DSN00 to '07'.

APJMINCL_DSNnn—These keywords let you define additional CA-Librarian data sets for extracting -INC members. These data sets follow any data sets generated on behalf of the APJMEXTR_INCLIST keyword.

A list must be created using these keywords. The first data set in the list must be specified in keyword APJMINCL_DSN01, the second in APJMINCL_DSN02, and so forth. The total number of data sets must be specified as a two-digit integer in keyword APJMINCL_DSN00. For example, if the last data set was specified in keyword APJMINCL_DSN07, set APJMINCL_DSN00 to '07'.

SIMPLE_LINK—This keyword informs the model whether it is extracting the member being processed by modeling or its related link stream member. A value of 0 means the link stream defined to the members Inventory is being extracted. If the keyword is not defined or if it has any other value, then the member being processed by modeling is to be extracted. This keyword must be set by the invoking compile/link model.

PANEXEC—This keyword indicates if the extract has to run under CA-Panexec. A value of 1 indicates to run the extract under CA-Panexec and generate any DD statements required. A value of 2 indicates to run the extract under CA-Panexec and generate the PANEIN control statements that run the extract. If the keyword is not defined or the value is blank, this indicates to not run the extract under CA-Panexec. Any other value is invalid and causes incorrect JCL to be generated. This keyword must be set by the invoking compile/link model.

LLIBR_PGM—If the member is to be extracted from CA-Librarian, this keyword indicates the name of the CA-Librarian program name, which is usually AFOLIBR. This keyword should be set in the APJMLEAD model.

NUMCONDS—This two-digit integer should contain the suffix of the most recently added CONDnn keyword that is used for the APJMCOND condition code statement model. If there are no CONDnn keywords set yet, NUMCONDS should contain 00.

CONDnn—If there were any steps preceding the extract that must run successfully for the extract to run, there must be a CONDnn keyword set for each step. See the APJMCOND condition code statement model for details.

GJCLLIST—This keyword controls the SYSOUT class for the JCL listings. It is set in the APJMLEAD model.

GSYSTEMP—This keyword specifies the OS/390 unit for work files. It is set in the APJMLEAD model.

User keywords returned from the model:

APJMEXTR_SOURCE_DD—This keyword returns to you the DD name of the sequential data set to which the source is extracted.

NUMCONDS—This keyword is incremented to reflect the additional CONDnn statements added for testing if all steps were successful so far. This keyword is not incremented if the extract is run under CA-Panexec.

CONDnn—Additional CONDnn keywords are created for testing if all steps were successful so far. This keyword is not incremented if the extract is run under CA Panexec. Refer to the APJMCOND model for details on CONDnn keywords.

Processing Overview:

The means by which the source is extracted varies, depending on the Access Method (CA-Panvalet, CA-Librarian, or PDS) and if it is being extracted under CA-Panexec.

CA Panvalet Standalone

A PAN#1 step is run to perform a ++WRITE to work of the member.

CA Panvalet under CAPanexec

This model must be invoked twice under CA-Panexec.

First when all the DD statements are being generated, this model must be invoked with a value of 1 in the PANEXEC keyword. At that time the DD statements necessary to extract the member are generated.

Next when the PANEIN control statements are being generated, this model must be invoked with a value of 2 in the PANEXEC keyword. A %WRITE of the member is done with the MS operands, causing the member to be scanned to collect CA-Panexec XIR information. Then PAN#1 is invoked to do a ++WRITE to work to extract the member.

CA Librarian Standalone

The APCS5923 utility is executed to locate the member, starting at the destination data set (turnover) or the compile level data set (Development Facility), and continuing with the data sets implied by the APJMEXTR_INCLIST and APJMINCL_DSNnn keywords. APCS5923 allocates the data set the member was found on to DD name MASTER. Then CA-Librarian is invoked from APCS5923 to extract the member. The -EXTRACT command with the VAR operand is used to extract the member, expanding SLAT (source load audit trail) variables. If an MCD code is required, CA-Librarian is invoked using the APCS5921 utility to resolve the MCD code. When CA-Librarian is invoked, the AUXINC output exit is used to support concatenated -INC libraries.

CA Librarian under CAPanexec

This model must be invoked twice under CA-Panexec.

First when all the DD statements are being generated, this model must be invoked with a value of 1 in the PANEXEC keyword. At that time the DD statements necessary to extract the member are generated.

Next when the PANEIN control statements are being generated, this model must be invoked with a value of 2 in the PANEXEC keyword. A %WRITEL of the member is done with the MS operands, causing the member to be scanned to collect CA-Panexec XIR information. Then the APCS5923 utility is executed to locate the member, starting at the destination data set (turnover) or the compile level data set (Development Facility), and continuing with the data sets implied by the APJMEXTR_INCLIST and APJMINCL_DSNnn keywords. APCS5923 allocates the data set the member was found on to DD name MASTER. If an MCD code is required, APCS5921 is run to resolve it. Then CA-Librarian is run to extract the member, using the AUXINC output exit to support concatenated -INC libraries. The -EXTRACT command with the VAR operand is used to extract the member, expanding SLAT (source load audit trail) variables.

PDS Standalone

The APCS5924 utility is executed to locate the member, starting at the destination data set (turnover) or the compile level data set (Development Facility), and continuing with the higher level data sets for the Library Code. When found, the data set and member are allocated as a sequential data set. Then IEBGENER is invoked from APCS5924 to copy the member out of the PDS.

PDS under CAPanexec

This model must be invoked twice under CA-Panexec.

First when all the DD statements are being generated, this model must be invoked with a value of 1 in the PANEXEC keyword. At that time the DD statements necessary to extract the member are generated.

Next when the PANEIN control statements are being generated, this model must be invoked with a value of 2 in the PANEXEC keyword. The APCS5924 utility is executed to locate the member, starting at the destination data set (turnover) or the compile level data set (Development Facility), and continuing with the higher level data sets for the Library Code. When found, the data set is allocated for the %WRITEP command. The member is then extracted using %WRITEP, which also collects CA-Panexec XIR information.

Miscellaneous Models

Leading Move Model

Model Name: APJMLEAD

Purpose: To manage and oversee the entire modeling process.

Processing Overview: This model is a focal point in managing the entire modeling process. It is invoked automatically by CA-PanAPT before any modeling specifications for all modeling phases, including the START and END phases.

Within this model are many places that require customization for your site. User keywords are set to maintain miscellaneous data, such as unit names for dasd and tape, sysout classes, etc. Other user keywords are also specified to perform customization for specific types of moves, such as PDS, CA-Panexec, and CA-Telon. Comments instruct you on what to modify.

This model also ties separate move jobs constructed on APTMDLO together so that the last step of one job submits the next job in a specific order. Moves are done first, followed by compiles, followed by DB2 BINDs. This model also appends JCL to the end of the last move job to run the APJP5395 and APJP6930 procs, which must be run at the end of a move cycle.

Trailing Move Model

Model Name: APJMTAIL

Purpose: To manage and oversee the entire modeling process.

Processing Overview: This model is invoked automatically by CA-PanAPT after processing the model specifications for all modeling phases, including the START and END phases.

During move modeling, the APJMTAIL model just sets $OUTPOST to $LIBCODE if it is currently blank. This is done to allow models written before version 2.0 to continue to work without modifications. If you have any post model processing you would like performed you should include it in this model.

COND Parameter

Model Name: APJMCOND

Usage: All model facilities.

Purpose: Generate a COND parameter for an EXEC statement.

User keywords passed to the model:

COND_PFX

This optional keyword contains prefixes for the following COND keywords. It allows multiple sets of COND keywords to exist and uses only one of those sets for any one invocation of APJMCOND. This keyword must not exceed 12 characters. This keyword is cleared upon exit of this model.

<COND_PFX>CONDCONT

This keyword should contain a Y if the EXEC statement is continued after the COND parameter. This causes the COND parameter to end with a comma.

<COND_PFX>COND1 through <COND_PFX>COND8

These are the conditions to be specified in the COND parameter. Each condition code specification must be enclosed in parenthesis. The generation of the COND parameter ends as soon as a blank or missing <COND_PFX>CONDn keyword is encountered. Only 8 conditions can be specified because that is the JES limitation.

Processing Overview:

A COND parameter is generated unless the <COND_PFX>COND1 keyword is missing or empty.

Model specification notes:

Many distributed models use a counter keyword to increment the last character of the <COND_PFX>CONDn keywords. This technique simplifies the setting of these keywords. Model APJCPDS is a rather simple model that gives an example of this.

External Processing Move Models

External Processing models are used during turnover. They perform processes other than movement, such as assemblies and compiles. They can be used in conjunction with move models, performing compiles after members are moved; or they can be used without move models, performing compiles on members that are not moved but are affected by members that did move, such as copybooks. The compile models (APJMASMB and APJMCOMP) are described earlier in this chapter.

Managing External Processing Jobs

Model Name: APJMCOMJ

Purpose: Creates leading job start JCL for External Processing models.

Processing Overview: This model is invoked by other external processing models to construct leading JCL for each member's external processing.

The operation of this model is controlled by the GCOMPILES_GROUPED user keyword, which is specified in the APJMLEAD model. A value of N means that each member's external processing is to be performed as a separate job. This is the default. A value of Y means that all external processing must be grouped together in a single job if possible.

When separate jobs are to be used, this model appends a step the previous external processing job (which is in its own member on APTMDLO) to submit the current member's external processing job upon completion. Then it constructs the start of the current member's external processing in a new member on APTMDLO. This member begins with a JOB statement and might contain other statements, such a JOBLIB DD's. The member name of the new member is returned to the external processing model through the GCOMPJCL_CURR user keyword.

When a single job is to be used, this model builds the leading job control only for the first member presented to modeling for external processing. The name of this member is always returned to the external processing model regardless of whether it was just created through the GCOMPJCL_CURR user keyword.

The actual leading JCL is constructed in the APJMJBST model, which this model invokes.

Miscellaneous Move Models

Blank Out $MSG Move Model

Model Name: APJMMSG

Usage: Daily Processing Cycle.

Purpose: Clears system keyword $MSG to blanks. This model assumes that the $MSG system keyword has been set in the Library Code model specification field to be printed once on the APCS5320-01 report, Today's moves by Library Code.

Members of the APTMDLO PDS:

$OUTDD=	None	
$OUTMEM	=	None
$OUTPOST	=	None

User keywords passed to the model: None

Output: $MSG system keyword set to blanks.

Processing Overview: $MSG system keyword set to blanks.

Job Start Move Model

Model Name: APJMJBST

Purpose: To construct a JOB statement and other leading JCL for a job.

Processing Overview: This model is invoked by other move models when they need leading JCL constructed for a JOB.

User keywords can be set by the invoking model to control the JOB statement as follows:

JOBNAME

The job name. If not specified, a job name of APJJMOVE is used.

JOBID

The programmer name portion of the job statement. If not specified, the literal 'GENERATED MOVE JOB' is used.

JOBCLASS

The class the job is to run in. If not specified, P is used.

JOBMSGCLASS

The MSGCLASS for the jobs SYSOUT. If not specified, P is used.

You can modify this model to change the defaults for these parameters to conform to your site's standards. You should also make any other modifications to conform to your site's standards.

Any load and JCL libraries needed for turnover that are not available without a JOBLIB DD or JCLLIB statement must be placed in this model. If your level of OS/390 does not support the JCLLIB statement you must modify this model to use an alternative, such as a PROCLIB DD or a JOBPARM statement.

Null Move Model (Support of Status Posting)

Model Name: APJMNULL

Usage: Daily Processing Cycle.

Purpose: Provides support for Library Codes that do not move members as part of daily CA-PanAPT Move processing (for example, IBM maintenance or IPLs).

Members of the APTMDLO PDS:

$OUTDD=	APTMDLO	
$OUTMEM	=	None
$OUTPOST	=	NULL#

Note: This is the output member name of the modeling statement $OUTPOST= NULL. The # is not coded on the modeling statement. It is automatically added when the model is processed.

User keywords passed to the model: None

Output: This model builds only the posting information required to set the proper status for Library Code members.

Processing Overview: No move processing is done. JCL executing as part of APJJ5320 processing uses the status member as input to program APCS5391 to set proper status for the members. Unless this is done, the members are never marked as moved and the Move Request status never indicates a completed move.

Turnover Move Models

CA-Librarian Moves

Model Name: APJMLIBR

Usage: Daily Processing Cycle.

Purpose: Move members of CA-Librarian masters as part of daily move processing (APJJ5310-APJJ5320).

Members of the APTMDLO PDS:

- $OUTDD = APTMDLO
- $OUTMEM =
  - <$OUTPFX>—main CA-Librarian move JCL
  - <$OUTPFX>A—control statements
  - <$OLITPFX>H—control statements for history information
  - <$OUTPFX>C—control statements
  - <$OUTPFX>E—control statements
- $OUTPOST = <$OUTPFX>#

Note: This is the output member name of the modeling statement $OUTPOST= <$OUTPFX>. The # is not coded on the modeling statement. It is automatically added when the model is processed.

Library Code Model Specifications:

See CA-Librarian Moves in the "Setups for Different Types of Moves" chapter in the *CA-PanAPT Reference Guide* for more information on using this model.

CA-Panexec Moves

Model Name: APJMPEX

Usage: Daily Processing Cycle

Purpose: Move members of CA-Panexec libraries as part of daily move processing (APJJ5310-APJJ5320).

Members of the APTMDLO PDS:

- $OUTDD = APTMDLO
- $OUTMEM =
  - <$OUTPFX>A—remove to destination2 protection file
  - <$OUTPFX>B—transfer from destination1 to destination2
  - <$OUTPFX>C—remove to destination1 protection file
  - <$OUTPFX>D—transfer from original to destination1
  - <$OUTPFX>E—remove from original
  - <$OUTPFX>—contains JCL
- $OUTPOST = <$OUTPFX>#

Note: This is the output member name of the modeling statement $OUTPOST= <$OUTPFX>. The # is not coded on the modeling statement. It is automatically added when the model is processed.

User keywords passed to the model:

CMDOPT

Command option scope (F,G,E, or blank).

SELGRP

Selected group to be used. The group specified in the Library Code can be overridden by the From User data field for the member on the Move Request.

SELTYPE

Selected type to be used.

SELSTAT

Selected status to be used.

SELMODE

Selected mode to be used.

Specify all of the above parameters in the Library Code definition for every Library Code that uses the model. For example, use the following specification when no CA-Panexec protection files are created:

CMDOPT=' ';SELGRP='PAYROLL';SELTYPE='T'

SELSTAT=' '; SELMODE=' '; INCLUDE APJMPEX

This model also uses the User Data field from the Move Request as the CA-Panexec group name. $FROMDATA overrides the group name for the sending library, and $TODATA overrides the group name for the receiving library.

Output: This model creates a set of members in model output library APTMDLO. Each member name consists of an output prefix ($OUTPFX) followed by a character to provide uniqueness. The output prefix is a combination of the model base plus the library level abbreviation.

Processing Overview: Builds control cards for use by CA-Panexec.

The access and override code for any Library Code can be specified in the security field for each data set in the Library Code definition. The first four bytes are the access code and the second four bytes are the override code.

The same model (APJMPEX) is used to process CA-Panexec moves with protection files and without them. Protection files must be specified in the library code specification before the model is included in order to alert CA-PanAPT to their existence. To use a protection file for any CA-Panexec type move, set the variable PF_DDN to the data set name of the protection file, excluding the GDG generation reference at the end of the name. Substitute DDN with the ddname of the library this protection file is for. For example, if you want to use a protection file name, PAYROLL.PROD.PEXBKUP for library PAYROLL.PROD.PEXLIB and the ddname for the latter is PEXPP, designate the following in the library code specification:

```
PF_PEX$P$P='PAYROLL.PROD.PEXBKUP;
 CMDOPT=' ';SELGRP='PAYROLL';SELTYPE='T'
SELSTAT=' '; SELMODE=' '; INCLUDE APJMPEX
```

CA Panvalet Moves

Model Name: APJMPANV

Usage: Daily Processing Cycle.

Purpose: Move members of CA-Panvalet libraries as part of daily move processing (APJJ5310-APJJ5320).

Members of the APTMDLO PDS:

- $OUTDD = APTMDLO
- $OUTMEM when using REXX:
  - <$OUTPFX>—main CA-Panvalet move JCL
  - <BASEPFX>G—move group initialization date
  - <$OUTPFX>I—list of move group init members
  - <BASEPFX>M—list of member data

    Where BASEPFX is the $OUTPFX value for the first member of the move grouping.
- $OUTMEM when not using REXX:
  - <$OUTPFX>—main CA-Panvalet move JCL
  - <$OUTPFX>A—control statements
  - <$OUTPFX>B—control statements
  - <$OUTPFX>D—control statements
  - <$OUTPFX>E—control statements
  - <$OUTPFX>G—control statements
  - <$OUTPFX>H—control statements
  - <$OUTPFX>I—control statements
- $OUTPOST:
  - <BASEPFX>#—when using REXX
  - <$OUTPFX>#—when not using REXX

  Note: These are the output member names of the modeling statements.

- $OUTPOST = '<$OUTPFX>'
  and
  $OUTPOST = '<BASEPFX>'

The # is not coded on the modeling statements; it is automatically added when the model is processed.

Library Code Model Specifications:

See CA-Panvalet Moves in the "Setups for Different Types of Moves" chapter in the *CA PanAPT Reference Guide* for information on using this model.

CA-PFF Moves

Model Name: APJMPFF

Usage: Daily Processing Cycle.

Purpose: Move members of CA-PFF libraries as part of the daily processing cycle (APJJ5310-APJJ5320).

Members of the APTMDLO PDS:

- $OUTDD = APTMDLO
- $OUTMEM =
  - <$OUTPFX>A—backup to destination2 protection file
  - <$OUTPFX>B—transfer from destination1 to destination2
  - <$OUTPFX>C—backup to destination1 protection file
  - <$OUTPFX>D—transfer from original to destination1 (purge included if desired)
  - <$OUTPFX>—contains JCL
- $OUTPOST = <$OUTPFX>#

Note: This is the output member name of the modeling statement $OUTPOST=<$OUTPFX>. The # is not coded on the modeling statement, It is automatically added when the model is processed.

User keywords passed to the model:

ALIAS

Specifies how CA-PFF is to process aliases on a PDS. Valid values are Y and N.

LIBTYPE

Specifies the type of PDS being processed. This model validates the value for the correct content. Valid values are ",", SOURCE, EXEC, and OBJECT.

Specify each of these parameters in the Library Code definition for every Library Code that uses the model. For example, use the following specification:

```
ALIAS='N'; LIBTYPE= ''; INCLUDE APJMPFF
```

Output: Several members are built in the model output library, APTMDLO. These members contain control statements that govern the members' movement for one step of the complete move procedure. The MOVEJCL member is the driver which includes the customized JCL in the <$OUTPFX> member on APTMDLO. That member does the actual move.

Processing Overview:

Moves are done in several stages depending on the number of library levels, the following is true for all:

- First, all members whose Library Codes require the use of a Backup Library are moved from the destination1 library to the destination2 or Backup Library.
- Second, members are moved between the originating library and the destination1 or target library. The following are possible moves:
  - From a lower level to a higher target level
  - From the current level to its backout library
- Third, all members that must be deleted (according to Library Code definitions) are deleted. This step is part of the copy/purge step.

The same model (APJMPFF) is used to process CA-PFF moves with protection files and without them. Protection files must be specified in the library code specification before the model is included in order to alert CA-PanAPT to their existence. To use a protection file for any CA-PFF type move, set the variable PF_DDN to the data set name of the protection file, excluding the GDG generation reference at the end of the name. Substitute DDN with the ddname of the library this protection file is for. For example, if you want to use a protection file name, PAYROLL.PROD.PFFBKUP for library PAYROLL.PROD.PFFLIB and the ddname for the latter is PFFPP, designate the following in the library code specification:

```
PF_PFF$P$P='PAYROLL.PROD.PFFBKUP;
 ALIAS='N'; LIBTYPE= ''; INCLUDE APJMPFF
```

CA-Telon Moves

Many CA-Telon move models exist. These are described under CA-Telon Moves in the "Setups for Different Types of Moves" chapter in the *CA-PanAPT Reference Guide*.

PDS Moves

Model Name:

APJMPDS

Usage:

Daily Processing Cycle.

Purpose:

Move PDS members as part of the daily processing cycle (APJJ5310-APJJ5320).

Members of the APTMDLO PDS:

- $OUTDD = APTMDLO
- $OUTMEM =
  - <$OUTPFX>J—main PDS move JCL
  - <$OUTPFX>1—DD statements for Backup step
  - <$OUTPFX>B—IEBCOPY control statements for Backup step
  - <$OUTPFX>2—DD statements for Move step
  - <$OUTPFX>M—IEBCOPY control statements for Move step
  - <$OUTPFX>3—DD statements for Scratch step
  - <$OUTPFX>S—APAS5900 control statements for Scratch step

  Note: When moves are being grouped, $OUTPFX is altered for each Library Code the $OUTPFX value of the first PDS Library Code processed for the group. The group consists of all members being moved or backed out to the same level. Moves and Backouts are grouped separately.
- $OUTPOST = <$OUTPFX>#

  Note: This is the output member name of the modeling statement $OUTPOST = '<$OUTPFX>'. The # is not coded on the modeling statement, it is automatically added when the model is processed.

Library Code Model Specifications:

See Partition Data Set Moves in the "Setups for Different Types of Moves" chapter in the *CA-PanAPT Reference Guide* for information on using this model.

Development Facility Utility Models

Compare Members

Model Name: APJDCP

Usage: Compare two members.

User keywords passed to the model: None

Processing Overview: The two members are compared. Depending upon the Access Method, the compare utility is either IBM SUPERC (PDS), CA-Panvalet PCOMPARE, or CA-Librarian COMPARITOR. Mixed Access Methods (for example, comparing a PDS member to a CA-Panvalet member) are not supported. The report is saved in a library, viewable by using the OCP line command in the Development Facility. A message is then sent to the user who initiated the compare, reporting the status of the compare job.

Merge Members

Model Name:

APJDMG

Usage:

Merge two changed members.

User keywords passed to the model:

None

Processing Overview:

The two changed members are merged, using the original unchanged member as a base. The merge is done using CA-Pan/Merge. If the members are on CA-Librarian masters, they are first extracted to a PDS because CA-Pan/Merge does not support CA-Librarian. The resulting merged member is added to the output library, and the report is saved in a library, viewable by using the OMG line command in the Development Facility. Mixed Access Methods (for example, merging a PDS member to a CA-Librarian member) are not supported. A message is then sent to the user who initiated the merge, reporting the status of the merge job.

Chapter 5: File Maintenance

This section contains the following topics:

About File Maintenance

The jobs described in this chapter report on:

- File backup to GDG tape (APJJBKUP)
- File restore from GDG tape or other medium (APJJREST)
- Initialize backup library (APJJINIT)
- Time sensitive processing (APJJNEXT)
- Create an Inventory Load file from the directories of libraries managed by CA-PanAPT (APJJ6910)
- Load the CA-PanAPT Inventory file (APJJ6920)
- Create an Inventory Load file from existing Inventory Records (APJJ6930)
- Purge Move Requests for the History file to the History file (APJJ5950)
- Create a Sequential file of Move Requests from the History file (APJJ5955).

File Backup to GDG Tape

Using standard utilities, back up CA PanAPT system files to a cataloged GDG tape (or other medium). This tape or other medium provides a mechanism for you to use in performing regular backup of the CA PanAPT system files.

Activity

Uses IDCAMS to back up the CA PanAPT database, and IEBCOPY to back up the Model Library. Optionally uses IDCAMS to back up the History file.

Output

Standard output produced by IDCAMS and IEBCOPY.

PROC

APJPBKUP

Programs

IDCAMS and IEBCOPY

APCS5910 is used to copy control cards to a temporary disk file (allows in-stream creation of SYSIN input for utilities).

PROC APJPBKUP

Parameter	Description	Default in PROC
VSAMPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT files. Change this to your site's high-level VSAM index.	none
TAPE =*x*	Designates the TAPE device. Change this to your site's standard for TAPE in JCL.	TAPE
TMSPARM=*x*	*x* is the keyword and value for the LABEL parameter. Valid values are EXPDT=? or REPDT=?. When this parameter is used, it MUST be preceded with a leading comma. DO NOT code a trailing comma. Example: TMSPARM=',EXPDT=99365'	none
SOFTPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT files. Change this to your site's high-level software library index.	none

Parameter	Description	Default in PROC
OTHRPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT files. Change this to your site's high-level sequential index.	none
SYSDA=*x*	*x* indicates a disk type.	SYSDA
MODDSCB=*x*	Model DSCB used for tape GDGs. This **must** be valid DSCB. If you do not have a system DSCB defined, you can use any cataloged data set name.	specified during installation
JCLLIST=*x*	*x* must be suitable to include in SYSOUT=*x*. Utility messages, control totals, and so forth are routed to this destination.	*
DUMPS=*x*	*x* must be suitable to include in SYSOUT=*x*. Any SYSUDUMP output is routed to this destination.	*
HISTFL=*x*	*x* indicates whether to back up the Pending History file. 0 = Execute this step 1 = Bypass this step	0

File Restore

Using standard utilities, restore CA-PanAPT processing Control files from Backup.

Activity

Uses IDCAMS to reallocate and restore the CA-PanAPT database or History. Uses IEFBR14 to reallocate Model file and IEBCOPY to restore Model file. The PROC APJPRSTV is executed once for each CA-PanAPT VSAM file.

Output

Standard output produced by IDCAMS and IEBCOPY.

PROC

APJPRSTV

For the VSAM files, executed once for each VSAM file.

APJPRSTP

For the PDS file (Model file).

Programs

IDCAMS and IEBCOPY.

APCS5910 is used to copy control cards to a disk file (allows in-stream creation of SYSIN input for utilities).

PROC APJPRSTV - VSAM

Parameter	Description	Default in PROC
VSAMDSN=*x*	*x* is the name of the data set that is restored. It should be the CA-PanAPT Model library, APTMODEL.	none
BKUPDSN=*x*	*x* must be the name given to the backup GDG tape.	none
SOFTPFX=*x*	*x* is concatenated ahead of the Standard data set name for CA-PanAPT files. Change this to your site's high-level software library index.	none

Parameter	Description	Default in PROC
DELDEF=*x*	Indicates the name of the CA-PanAPT PARMLIB member. When restoring the database, specify: DELDEF=APJRDB When restoring the History file, specify: DELDEF=APJRHIST	none
SYSDA=*x*	*x* is the DASD unit name used to store the APTMODEL file.	
JCLLIST=*x*	*x* must be suitable to include as a SYSOUT class. Utility messages, control totals, JCL, and so forth are routed to this destination.	*
DUMPS=*x*	*x* must be suitable to include as a SYSOUT class. Any SYSUDUMP output is routed to this destination.	*

PROC APJPRSTP

Parameter	Description	Default in PROC
DISKDSN=*x*	*x* is the name of the data set that is restored. It should be the CA-PanAPT Model library, APTMODEL.	none
TAPEDSN=*x*	*x* is the name of the backup data set that is restored to the Model library. It is normally a GDG.	none
DELETE=*x*	*x* is the Condition Code to control Deletion of the old APTMODEL PDS before its being restored. 0 = Yes, delete before restore 1 = No, do not delete before restore	0
ALLOC=*x*	*x* is the Condition Code to control reallocation of the old APTMODEL PDS before its being restored. 0 = Yes, allocate after delete and before restore. 1 = No, do not reallocate because it was not deleted. If DELETE=0, ALLOC must be 0. If DELETE=1, ALLOC must be 1.	0

Parameter	Description	Default in PROC
SYSDA=*x*	*x* is the DASD unit name used to store the APTMODEL file.	SYSDA
DISKVOL=*x*	*x* is the DASD VOL=SER on which the APTMODEL file is stored.	none
DISKSPA=*x*	*x* is the SPACE parameter used to reallocate the APTMODEL file after it was deleted and before being stored	3120,(120,15,20),RLSE
DISKDCB=*x*	*x* is the DCB parameter used to reallocate the APTMODEL file after it was deleted and before being stored.	RECFM=FB, LRECL=80, BLKSIZE=3120
JCLLIST=*x*	*x* must be suitable to include as a SYSOUT class. Utility messages, control totals, JCL, and so forth are routed to this destination.	*
DUMPS=*x*	*x* must be suitable to include as a SYSOUT class. Any SYSUDUMP output is routed to this destination.	*

Initialize Backup Library

Remove all members from the backup PDS that contains old versions of Production members. Delete and reallocate the data set.

Activity

This job assumes that all old members have been copied to an archive tape already.

The backup data set is deleted without first copying the contents to an archive medium. The backup data set is then reallocated using the overriding space parameter and the DCB information from the Production data set with which it is associated.

Output

An empty backup data set.

Program

IEFBR14

PROC APJPINIT

Parameter	Description	Default in PROC
BKUPDSN=*x*	*x* is the name of the data set that is deleted and reallocated. The data set contains copies of past Production members.	none
PRODDSN=*x*	*x* is the name of the Production data set that provides DCB information for the reallocation of the backup data set.	none
BKUPSPA=*x*	*x* is the SPACE parameter used to reallocate the backup file after it has been deleted.	(6356,(200, 200,135))
SYSDA=*x*	*x* is the DASD unit name used to store the BACKUP PDS.	SYSDA

Time Sensitive Processing

Move Next Day Library to Production Library.

This procedure is coded to handle PDSs. The concept can be modified to handle any type of library structure by using an appropriate utility to move the members from the Next Day Library to the Production Library.

Activity

Copy all members from the Next Day Library to the corresponding Production Library. Then delete and reallocate the Next Day Library.

Output

An updated Production Library and an empty Next Day Library. Standard IEBCOPY messages are produced.

PROC

APJPNEXT - Time sensitive moves.

Programs

APCS5910

Creates control statement for IEBCOPY.

IEBCOPY

Copies members.

IEFBR14

Deletes and reallocates Next Day library.

PROC APJPNEXT

Parameter	Description	Default in PROC
NEXTDSN=*x*	*x* is the name of the library that contains Production members that are copied from the Next Day Library to the Production Library. This library is deleted and reallocated after the members are successfully copied. **Note:** This library must be reallocated for Job APJJ5320 to execute (when a Next Day Library is being used). The default parameters are placed in this library on a scratch pack and therefore, it can be scratched before Job APJJ5320 is run.	none
PRODDSN=*x*	*x* is the name of the Production Library that receives the members from the Next Day Library and provides the DCB information for the reallocation of the Next Day Library.	none
NEXTSPA=*x*	*x* is the SPACE parameter used to reallocate the Next Day Library after it has been deleted.	(6356,(100,100,90))
SYSDA=*x*	*x* is the DASD unit name used to store both the Next Day Library and the temporary data set that contains the control statement passed to IEBCOPY. If SYSDA points to a pack that is cleaned up daily, specify a Volume parameter that points to a non-scratch pack. For example, **SYSDA=SYSDA,VOL=SER=PACK01,** or **//libcode.ALLOCATE DD VOL=SER=PACK01**	SYSDA
JCLLIST=*x*	*x* is a SYSOUT class and other SYSOUT parameters. Utility messages, control totals, JCL, and so forth are routed to this destination.	*
DUMPS=*x*	*x* is a SYSOUT class and other SYSOUT parameters. Any SYSUDUMP output is routed to this destination.	*

Create an Inventory Load File from Library Directories

Provides a way to populate the CA-PanAPT Inventory file from the directories of libraries managed by CA-PanAPT.

You can create an Inventory load file from a PDS, CA-Panvalet, or CA-Librarian directory. A sequential file is created that is input to job APJJ6920 for the actual file load. The record layout for this file is provided as member APCCDIB2 in the CA-PanAPT SRCELIB data set. See Appendix C for a listing of this member.

You must specify a valid Library Code/Subcode and the type of directory in use (PDS, CA-Panvalet, or CA-Librarian). In addition, specify a member name range on instream control statements.

For PDS directories, only the member name is supplied from the directory. All other inventory information is supplied from the defaults on the Library Code file.

For CA-Panvalet directories, the member name, description, and language are supplied from the directory. The comment data from the CA-Panvalet directory is copied to the inventory record description field. All other inventory information is supplied from the defaults on the Library Code file. (If description or language is blank on the directory, the defaults on the Library Code file are used.) Additionally, the member name range can be further restricted by specifying selection parameters on the ++PRINT 2-UP,COMMENT statement (,TYPE=BAL). See your *CA-Panvalet System Management Guide* for more details.

For CA-Librarian directories, the member name, description, and language are supplied from the directory. All other inventory information is supplied from the defaults on the Library Code file. (If description or language is blank on the directory, the defaults on the Library Code file are used.)

Activity

For CA-Panvalet and CA-Librarian directories, produce a temporary sequential file with directory information.

Use the directory file or a PDS index to produce a sequential file that is loaded to the Inventory (DIBS) file by job APJJ6920. APJJ6920 gets default values from the CA-PanAPT Library Code file before loading the Inventory Records to the DIBS file.

Output

A sequential Inventory Load file.

APCS6910-00

Execution Log and program messages. This report shows the library type, Library Code/Subcode, and the number of records processed.

APCS6910-01

Inventory Load Exception Report.

PROC

APJP6910—Create an inventory load file.

Program

APCS6910

PROC APJP6910

The first three parameters in the table below are mutually exclusive (PDSDSN, PANINDX, LIBINDX.) Delete or comment out the two parameters that are not used in job APJJ6910.

Parameter	Description	Default in PROC
PDSDSN=*x*	*x* is the name of a data set (usually production) that is to be used for an index listing. Inventory Records are produced for each member in this library, but are limited to those members who meet the criteria indicated on the optional From-member and To-member control statements. **Note:** Delete or comment out this parameter in job APJJ6910, if you are not using a PDS.	NULLFILE

Parameter	Description	Default in PROC
PANINDX=*x*	*x* is the name of the temporary sequential data set that contains an index listing that is created in the first step. Inventory Records are produced for each member in the directory listing but are limited to those members that meet the criteria indicated on the optional From-member and To-member control statements. **Note:** Delete or comment out this parameter in job APJJ6910, if you are not using a CA-Panvalet directory.	NULLFILE
LIBINDX=*x*	*x* is the name of the temporary sequential data set that contains an index listing that is created in the first step. Inventory Records are produced for each member in the directory listing, but are limited to those members that meet the criteria indicated on the optional From-member and To-member control statements. **Note:** Delete or comment out this parameter in job APJJ6910 if you are not using a CA-Librarian directory.	NULLFILE
VSAMPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT files. Change *x* to your site's high-level VSAM index.	none
OTHRPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT passed and temporary files (APT6910.DIBSFILE.) Change *x* to your site's high-level index for sequential CA-PanAPT data sets.	none
SYSDA=*x*	*x* is the DASD unit name used to store the sequential data set.	SYSDA
JCLLIST=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Utility messages, control totals, JCL, and so forth are routed to this destination.	*
REPORTS=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Reports are routed to this destination. Change this in the JCL to your site's standards.	*
DUMPS=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Any SYSUDUMP output is routed to this destination.	*

Input Selection Criteria from DD APTSYSIN

You must specify at least two control statements:

1. INPUT-TYPE = '*X*' ,

   where *X* is PDS, PANV, or LIBRA

2. LIBCODE = 'LLLL/SSS'.

   This LIBCODE must already exist on the APTLIBC file.

   You can restrict the number of members selected from the directory list and subsequent Inventory Records created by coding two optional control statements:
   `FROM-MEMBER='X'`

   and
   `TO-MEMBER='X'`

Each keyword can be specified only once. If any keyword is encountered more than once, an error message is issued and the program abends.

Multiple control statements can be included on a single SYSIN record if they are separated by a semicolon. For example:

```
FROM-MEMBER='GL0000' ; TO-MEMBER='GL9999'
```

Input Selection Criteria from DD PANV.SYSIN

For CA-Panvalet libraries that contain multiple types of members, it might be necessary to restrict the members selected for each Library Code beyond the From-member/To-member control statements. Additional selection criteria can be specified on the ++PRINT 2-UP,COMMENT control statement in step PANV in job APJJ6910.

For example, ++PRINT 2-UP,COMMENT,TYPE=COBOL produces a partial list of members whose language type is COBOL.

This partial list can then be further restricted by using the From-member/To-member control statements. See the *CA-Panvalet MVS, VSE System Management Guide* for further details and other options.

Load the Inventory File

Load the CA-PanAPT Inventory (APTDIBS) file using the file created in job APJJ6910. Inventory field values that are not filled with data from the directory get the default values specified on the Library Code. New records can optionally replace existing records.

The Inventory Approval Flag can be set to YES or NO for the records added to the file.

These records might not have complete and accurate information. After the records are loaded, review the records online or run jobs APJJ6111 and APJJ6113 to produce Inventory reports.

Activity

Use the CA-PanAPT Library Code file and the sequential file created in job APJJ6910 to create Inventory Records. These records are then loaded to the CA-PanAPT Inventory file.

Output

APTDIBS

An updated CA-PanAPT Inventory file.

APCS6920-00

Execution Log and program messages. This report shows the control statements used and the number of records processed.

APCS6920-01

Inventory Creation Report. For each record added to the Inventory file, the Library Code, the member name, and the action message are displayed.

APCS6920-02

Inventory Creation Error Report.

PROC

APJP6920

Program

APCS6920

PROC APJP6920

Parameter	Description	Default in PROC
VSAMPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT files. Change *x* to your site's high-level VSAM index.	none
OTHRPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT passed and temporary files (APT6910.DIBSFILE.) Change *x* to your site's high-level index for sequential CA-PanAPT data sets.	none
JCLLIST=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Utility messages, control totals, JCL, and so on are routed to this destination.	*
REPORTS=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Reports are routed to this destination. Change this in the JCL to your site's standards.	*
DUMPS=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Any SYSUDUMP output is routed to this destination.	*

Input Selection Criteria from DD APTSYSIN

You can specify two optional control statements:

1. REPLACE='*x*',

   (where *x* can be Y or N)

2. APPROVED-FLAG='*x*',

   (where *x* can be Y or N)

The default value for each control statement is N; existing records are not replaced and the Approval Flag is set to No.

Specify each keyword only once. If any keyword is encountered more than once, an error message is issued and the program abends.

Create an Inventory Load File from Inventory Records

Provide a simplified way to populate the CA-PanAPT Inventory file from existing Inventory Records.

Activity

APCS6930 reads the Inventory Records of selected members and writes them to a sequential file in the format defined in COBOL COPY library member APCCDIB2. You can use the first step in the CA-PanAPT Inventory Load process, APCS6910, to create records in APCCDIB2 format from PDS, CA-Panvalet, or CA-Librarian directories.

The output of PROC APJP6930 is suitable to be loaded by PROC APJP6920. You can write programs to change some inventory values from the output of APJP6930 and then reload the inventory, or you can change the member names to create brand new inventory.

Extract Output

Output from APCS6930 is written to a sequential file with the DDname: DIBSFILE.

APCS6930 issues one of the following return codes:

0

One or more records were selected.

4

No records meeting the specified selection criteria were found.

PROC

APJP6930 - Create an Inventory load file.

Program

APCS6930

PROC APJP6930

Parameter	Description	Default in PROC
VSAMPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT files. Change *x* to your site's high-level VSAM index.	none
OTHRPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT passed and temporary files (APT6910.DIBSFILE.) Change *x* to your site's high-level index for sequential CA-PanAPT data sets.	none
SYSDA=*x*	*x* is the DASD unit name used to store the sequential data set.	SYSDA
JCLLIST=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Utility messages, control totals, JCL, and so forth are routed to this destination.	*
DUMPS=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Any SYSUDUMP output is routed to this destination.	*

Input Selection Criteria from DD APTSYSIN

Selection criteria are entered with optional keyword parameters through an APTSYSIN file. If the parameters are not specified, all Inventory Records on the APTDIBS file are selected. If a parameter is specified more than once, the last parameter value specified is used (for example, if you enter DB2='SOURCE' and DB2='PLAN', DB2='PLAN' is the value used by APCS6930).

Columns 73-80 of the APTSYSIN file must be numeric or blank. Parameters must be entered in columns 1-72.

The parameters can be used in any combination. Only members meeting all criteria are selected. For example, if you specify **LIBC='SRCE/DB2';LANGUAGE='COBOL',** only Inventory information for those members in Library Code SRCE/DB2 whose LANGUAGE field contains COBOL is chosen. See member APJJ6930 in your JCL library for more information about entering the selection criteria.

The keyword parameters are listed below.

Note: The DB2 keyword parameter is used with the CA-PanAPT DB2 option.

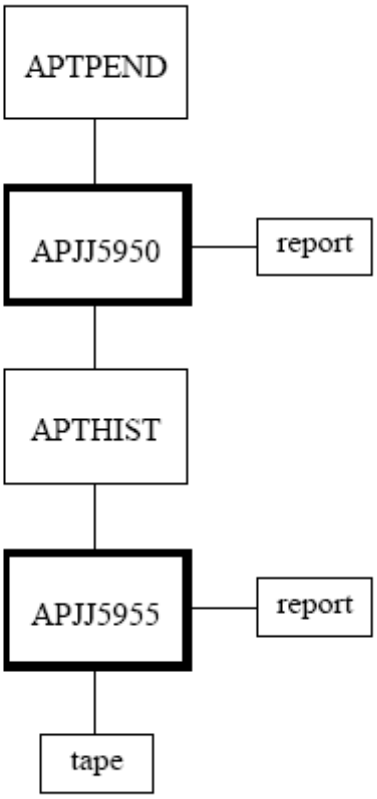
Keyword Parameter	Description
LIBC='*xxxx*/*xxx*'	Selects members in the specified Library Code.
DB2='Y'	Selects all DB2 members. That is, members whose inventory qualifiers match those for any Library Code with Type values starting with 'DB2'. If the CA-PanAPT DB2 option is not installed, the DB2 keyword cannot be specified.
DB2='N'	Bypasses all DB2 members.
DB2='SOURCE'	Selects only members for DB2 SOURCE Library Codes. For compatibility with prior releases of CA-PanAPT this can also be specified as **DB2='S'**. DB2 SOURCE Library Codes are those with Type values of **'DB2SRC'**.

Keyword Parameter	Description
DB2='DBRM'	Selects only members for DB2 DBRM Library Codes. For compatibility with prior releases of CA-PanAPT, this can also be specified as **DB2='D'**. DB2 DBRM Library Codes are those with Type values of **'DB2DBRM'**.
DB2='SRCEDBRM'	Selects only members for DB2 SOURCE/DBRM combined Library Codes. For compatibility with prior releases of CA-PanAPT, this can also be specified as **DB2='SD'**. DB2 SOURCE/DBRM Library Codes are those with Type values of **'DB2SDB'**.
DB2='PACKAGE'	Selects only members for DB2 PACKAGE Library Codes. For compatibility with prior releases of CA-PanAPT, this can also be specified as **DB2='K'**. DB2 PACKAGE Library Codes are those with Type values of **'DB2PACK'**.
DB2='PLAN'	Selects only members for DB2 PLAN Library Codes. For compatibility with prior releases of CA-PanAPT, this can also be specified as **DB2='P'**. DB2 PLAN Library Codes are those with Type values of **'DB2PLAN'**.
APPLICATION='*xxxxxxxx*'	Selects members with the specified application.
LANGUAGE='*xxxxxxxx*'	Selects members with the specified language.
QUALIFIER='*xxxxxxxx'*	Selects only those members with the specified Inventory Qualifier.
MEMBER='*xxxxxxxxxx*'	Selects only those members with the specified member name.

Purging to the History File

Job APJJ5950 deletes Move Requests from the Pending file and adds these Move Requests to the History file. The creation of the History file is included as part of the CA-PanAPT installation job. If this file was not created, you can modify and run just those parts of the CA-PanAPT installation JCL that perform an IDCAMS delete, define, and REPRO of the History file.

Job APJJ5950 is usually run with job APJJ5955. A processing flowchart is shown next.



Activity

Before a Move Request can be physically removed from the Pending file, it must be in one of three statuses:

- Moved To *level* (MV*?*)
- *level* Bkot Complete (MV*?*-B)
- Deleted (DEL)

The selection of Move Requests is based on selection criteria keywords in the APTSYSIN file. The selection criteria allow a number of Move Request fields to be specified. These include single Move Request numbers, ranges of Move Request numbers, and the service request and age of the Move Request.

See the topic Input Selection Criteria from DD APTSYSIN presented later in this subject for a list of eligible keywords.

PROC

APJP5950

Program

APCS5950

Input

The ACTION="PURGEPEND" keyword is required in the APTSYSIN file. All other keywords are optional. If the ACTION keyword is not specified or if an error is encountered with any of the keywords, the program ABENDs with a User 1000, and an error is logged on the APCS5950-00 Execution Log.

Keywords can be entered in one or more SYSIN records. If more than one keyword appears on a single SYSIN record, the keywords must be separated by a comma. The last keyword on a SYSIN record does not have to end with a comma. If there is only one keyword on a SYSIN record, then the commas are not required.

The keyword name must be entered in its entirety on a SYSIN record. For example, you cannot start a keyword name on one SYSIN record and then continue that name onto the next SYSIN record. If this is done, the program ABENDs with a User 1000 and an error is generated on the APCS5950-00 Execution Log.

Move Requests can be selected or bypassed by status. Specify selection by the **STATUS=** keyword, and bypass by the STATUS-OMIT= keyword. Specify multiple statuses either by separating them by commas in the value, as in:

```
STATUS="MVP,MVP-B,DEL"
```

Or specify multiple status keywords, as in:

```
STATUS="MVP",STATUS="MVP-B"
STATUS="DEL"
```

Any statuses specified that are ineligible for being purged or non-existent, are ignored.

Move Requests can be selected or bypassed by Move Request Number. Specify selection by the **MR=** keyword and bypass by the MR-OMIT keyword. Like the STATUS and STATUS-OMIT keywords, you can specify multiple Move Requests within a keyword value, and you can specify the keyword multiple times. Also, the Move Request value can be a single Move Request number, or a range in the format LOWER:HIGHER as in the following example:

```
MR="5,10,15:200,300",MR-OMIT="50:60,45"
```

Note that in the overlap of MR-OMIT 50:60 and MR 15:200, the MR-OMIT takes precedence, selecting 15 through 49 and 61 through 200.

Selection by Move Requests can improve performance of this program if a significant number of Move Requests do not have to be read to determine whether they are eligible to be purged.

Move Requests can be selected or bypassed by Service Request. Specify selection by the **SR=** keyword, and bypass by the SR-OMIT keyword. Like the STATUS and STATUS-OMIT keywords, you can specify the keyword multiple times.

Restrict the selection of Move Requests to those that have certain Move Types by the MPC= (Move Processing Cycle) keyword. Specify all Move Types to be selected in the value, with no separating commas. For example, to select Move Types C, D, and M, specify **MPC="CDM"**. Do not specify multiple MPC keywords, only the last one specified is in effect.

Select Move Requests by age, using the **MR-AGE=** keyword. The value of the keyword can be a single number, specifying the exact number of days ago the Move Request was last moved, or a range of ages separated by colons. This is just like the MR selection keyword. You can select those Move Requests moved yesterday and those moved a year ago or more with the following:

```
MR-AGE="1,365:99999"
```

See the topic Input Selection Criteria from DD APTSYSIN presented later in this subject for some sample selection criteria.

A Move Request must satisfy ALL selection criteria to be deleted from the Pending file. For example, if the following selection criteria is entered as two SYSIN records in the APTSYSIN file:

```
ACTION="PURGEPEND",STATUS="DEL"
MR="10:99",SR="PR000025"
```

Only a Move Request on the Pending file that is in a Deleted status, has a Move Request number between 10 and 99, **and** has a Service Request of PR000025 is deleted from the Pending file.

Output

Each Move Request on the Pending file that meets all selection criteria is written first to the History file. Next, the Move Request is deleted from the database, and finally, an entry is logged on the APCS5950-01 Selection Report. If after purging the Move Requests there are any Move Levels defined to your database that are pending deletion, APCS5950 evaluates the database to determine if there are no longer any references to those levels, and if not then remove those levels.

After purging the Move Requests, all move levels defined to your database are also defined to your History file. Any move levels that were already defined to your History file that are no longer defined to your database are marked as pending deletion. Those levels are not removed from your History file until the APCS5955 program determines that they are no longer referenced. The Move Requests on the History file are in the same format as those on the database.

There is an internal identifier on the History file to indicate that it is not the actual database. This ensures that the database is not used inadvertently as the History file in this job.

If an abend or system failure occurs while job APJJ5950 is executing, just rerun this job because the job has been designed to be self-correcting.

Output Report

APCS5950-01 - Pending File Purge Selection Report

PROC APJP5950

Parameter	Description	Default in PROC
VSAMPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT files. Change this to your site's high-level VSAM index.	none
JCLLIST=*x*	*x* must be suitable to be included in SYSOUT=*x*. Utility messages, control totals, and similar information are routed to this destination.	*
REPORTS=*x*	*x* must be suitable to be included in SYSOUT=*x*. Output reports are routed to this destination. Change this in the JCL to your site's standards.	*
DUMPS=*x*	*x* must be suitable to be included in SYSOUT=*x*. Any SYSUDUMP output is routed to this destination.	*

Input Selection Criteria from DD APTSYSIN

Some examples of specifying selection criteria are listed below. Each example represents a single execution of job APJJ5950.

- This example purges all Move Requests between 100 and 199 that are in Moved to QA status (they are not moving to any higher levels). These Move Requests are added to the History file.

  ```
  ACTION="PURGEPEND",STATUS="MVQ",MR="100:199"
  ```

- This example purges all Move Requests in a Deleted or Moved to Prod status that are 60 days old or older. These Move Requests are added to the History file.

  ```
  ACTION="PURGEPEND"
  STATUS="DEL,MVP"
  MR-AGE="60:99999"
  ```

- This example purges all Move Requests between 10 and 100 that are currently Backed Out from Production. These Move Requests are added to the History file.

  ```
  ACTION="PURGEPEND",STATUS="MVB-B",MR="10:100"
  ```

  This example purges Move Request 25. This Move Request is added to the History file.

  ```
  ACTION="PURGEPEND",MR="25"
  ```

Create Sequential File from the History File

APJJ5955 creates a sequential file of Move Requests from the History file and optionally deletes these Move Requests from the History file. These Move Requests were deleted from the Pending file (APTPEND) by running job APJJ5950.

The record layouts of the Move Requests on the sequential file are supplied so that user-written reports can be run against this file. These record layouts can be found in the CA-PanAPT SRCELIB as members APCCMDES, APCCMMBR, and APCCMAPP. See Appendix C for a listing of these members.

APJJ5955 uses the same selection criteria keywords as job APJJ5950 that was described in the previous subsection, but the ACTION keyword supports additional values.

Job APJJ5955 is usually run with job APJJ5950. See the processing flowchart shown earlier under Purging to the History File.

PROC

APJP5955

Program

APCS5955

Input

Keywords can be entered in one or more APTSYSIN records. If more than one keyword appears on a single record, the keywords must be separated by a comma. The keyword name and value must fit in its entirety on a single record.

The keyword ACTION is required. The values that can be specified for ACTION are **PURGEHIST**, **DUMPHIST**, **PRINTHIST**, **DUMPPEND**, and **PRINTPEND**.

ACTION="PURGEHIST"

Selected Move Requests are deleted from the History file and written to the sequential file.

ACTION="DUMPHIST"

Selected Move Requests on the History file are written to the sequential file, but are not deleted.

ACTION="PRINTHIST"

Selected Move Requests on the History file are simply reported. They are not deleted or written to a sequential file.

ACTION="DUMPPEND"

Selected Move Requests on the database are written to the sequential file, but are not deleted.

ACTION="PRINTPEND"

Selected Move Requests on the database are simply reported. They are not deleted or written to a sequential file.

Move Requests can be selected or bypassed by status. Specify selection by the **STATUS=** keyword, bypass by the STATUS-OMIT= keyword. Specify multiple statuses either by separating them by commas in the value, as in:

```
STATUS="MVP,MVP-B,DEL"
```

Or specify multiple status keywords, as in:

```
STATUS="MVP",STATUS="MVP-B"
STATUS="DEL"
```

Move Requests can be selected or bypassed by Move Request Number. Specify selection by the **MR=** keyword, bypass by the MR-OMIT keyword. Like the STATUS and STATUS-OMIT keywords, you can specify multiple Move Requests within a keyword value, and you can specify the keyword multiple times. Also, the Move Request value can be a single Move Request Number, or a range in the format LOWER:HIGHER. An example follows:

```
MR="5,10,15:200,300",MR-OMIT="50:60,45"
```

Note that in the overlap of MR-OMIT 50:60 and MR 15:200, the MR-OMIT takes precedence, selecting 15 through 49 and 61 through 200.

Selection by Move Requests can improve performance of this program if a significant number of Move Requests do not have to be read to determine whether they are eligible to be purged.

Move Requests can be selected or bypassed by Service Request. Specify selection by the **SR=** keyword, bypass by the SR-OMIT keyword. Like the STATUS and STATUS-OMIT keywords, you can specify the keyword multiple times.

Restrict the selection of Move Requests to those that have certain Move Types by the MPC= (Move Processing Cycle) keyword. Specify all Move Types to be selected in the value, with no separating commas. For example, to select Move Types C, D, and M, specify **MPC="CDM"**. Do not specify multiple MPC keywords, only the last one specified is in effect.

Select Move Requests by age, using the **MR-AGE=** keyword. The value of the keyword can be a single number, specifying the exact number of days ago the Move Request was last moved, or a range of ages separated by colons. This is just like the MR selection keyword. You can select those Move Requests moved yesterday and those moved a year ago or more with the following:

```
MR-AGE="1,365:99999"
```

See the topic Input Selection Criteria from DD APTSYSIN presented later in this subject for sample selection criteria.

For a Move Request to be processed, it must satisfy ALL selection criteria. For example, if the following selection criteria are entered in the APTSYSIN file:

```
ACTION="PURGEHIST",STATUS="DEL"
MR="10:99",SR="PR000025"
```

Only Move Requests on the History file that are in Deleted status in the Move Request Number range 10 through 99 with a Service Request of PR000025 are purged to the sequential file.

Output

If the ACTION keyword is PURGEHIST, each Move Request on the History file that satisfies the selection criteria is written to the sequential file, reported on, deleted from the History file, and an entry is logged on the APCS5955-01 Selection Report.

If, after purging the Move Requests, there are any Move Levels defined to your History file that are pending deletion, APCS5955 evaluates the History file to determine if there are any references to those levels, and if not, then removes those levels.

If the ACTION keyword is DUMPHIST, each Move Request on the History file that satisfies the selection criteria is written to the sequential file, reported on, and an entry is logged on the APCS5955-01 Selection Report.

If the ACTION keyword is PRINTHIST, each Move Request on the History file that satisfies the selection criteria is reported on, and an entry is logged on the APCS5955-01 Selection Report.

If the ACTION keyword is DUMPPEND, each Move Request on the database that satisfies the selection criteria is written to the sequential file, reported on, and an entry is logged on the APCS5955-01 Selection Report.

If the ACTION keyword is PRINTPEND, each Move Request on the database that satisfies the selection criteria is reported on, and an entry is logged on the APCS5955-01 Selection Report.

Move Request Deletion

The recommended method for deleting Move Requests is to first run APJJ5950, which transfers Move Requests from the Pending file to the History file, and then run APJJ5955 to create a sequential file from the History file. In the event that an ABEND or system failure occurs while job APJJ5950 is deleting from the Pending file and creating the History file, preserve the Move Requests that have been deleted before the system failure or ABEND.

You might be able to recover the sequential data set by using a utility that writes a final EOF to the partially created file. In any case, if such a failure occurs, rerun PROC APJJ5955 immediately after the system is recovered. Program APCS5955 detects that a failure occurred previously, and it completes any outstanding purge operations, thus leaving the History file in proper shape for normal CA-PanAPT operations.

Output Report

APCS5955-01 - History File Purge Selection report

PROC APJP5955

Parameter	Description	Default in PROC
VSAMPFX=*x*	*x* concatenates before the standard data set name for CA-PanAPT files. Change this to your site's high-level VSAM index.	none
OTHRPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT files. Change this to your site's high-level sequential index.	none
TAPE=*x*	*x* is the UNIT name that contains the newly created Move Request file. Tape is the normal medium, but you might want to store it on disk. If you specify a unit group (SYSDA) or type (3380), you must also specify a SPACE parameter. For example: **TAPE="SYSDA,SPACE=(TRK,(60,15))"**	TAPE

Parameter	Description	Default in PROC
TMSPARM=*x*	*x* is the keyword and value for the LABEL parameter. Values can be EXPDT=? or REPDT=?. When this parameter is used, it MUST be preceded with a leading comma. DO NOT code the trailing comma. For example: **TMSPARM=",EXPDT=99365"**	none
JCLLIST=*x*	*x* must be suitable to be included in SYSOUT=*x*. Utility messages, control totals, and so forth are routed to this destination.	*
REPORTS=*x*	*x* must be suitable to be included in SYSOUT=*x*. Output reports are routed to this destination. Change this in the JCL to your site's standards.	*
DUMPS=*x*	*x* must be suitable to be included in SYSOUT=*x*. Any SYSUDUMP output is routed to this destination.	*
MODDSCB=*x*	*x* is the model DSCB used for tape GDGs. This must be a valid DSCB. If you do not have a system DSCB defined, you can use any cataloged data set name.	specified during installation

Input Selection Criteria from DD APTSYSIN

Some examples of specifying the selection criteria are listed below. Each example represents a single execution of the purge job APJJ5955.

- This example purges from the History file all Move Requests between 100 and 199 that are in Moved to QA status. These Move Requests are written to the sequential output file.

  ```
  ACTION="PURGEHIST",STATUS="MVQ",MR="100:199"
  ```

- This example purges from the History file all Move Requests in Deleted or Moved to Production status that were moved 90 or more days ago. These Move Requests are written to the sequential output file.

  ```
  ACTION="PURGEHIST"
  STATUS="DEL,MVP"
  MR-AGE="90:99999"
  ```

- This example purges from the History file all Move Requests in Deleted or Moved to Production status that were moved 90 or more days ago. These Move Requests are written to the sequential output file.

  ```
  ACTION="DUMPPEND"
  MR="190"
  ```

Chapter 6: Batch Interface

This section contains the following topics:

About the Batch Interface

The batch interface consists of three programs:

- APCS5960 lets the user add multiple move requests to the Pending file.
- APCS5970 lets the user:
  - Add
  - Change
  - Delete
  - Copy
  - Verify
  - Run a verification procedure
  - Close a Move Request
  - One Move Request change is processed at a time by this program.
- APCS5970 allows the user to produce a LIBCODE extract.

Add Move Requests to the Pending File

Add Move Requests based on records created by user-supplied programs or the output created from APJJ5955. If you are using the dumped records of an APJJ5955 job, no modification should be necessary as APJJ5960 ignores Approval and Verification records and just adds description and member records. This effectively lets you copy Move Requests in a batch environment. You can also write a program that takes information from other products and creates the Add-Move-Request records for this job. This eliminates any need to reenter data, such as member names, into the online CA-PanAPT system when the information is available elsewhere.

Activity

This job reads records from the batch Add-Move-Request file, creates Move Requests using information from the input records, and adds the new Move Request(s) to the Pending file. The record format for the input file is the same one used by the batch Purge Move Request job (APJJ5955).

A report is produced to list each Move Request created and added to the Pending file. Use this report to obtain the Move Request numbers for Move Requests created by this job.

APJP5960 looks for groups of Add-Move-Request records on the file to create a Move Request. Each group of records consists of a Move Request Description record and a Member Name/Library Code record.

The first record in the group is the Description record. This record has a record ID of 01, and it is required to add a Move Request. The information in this record corresponds to the online Description of Move Request panel.

The next record is the Member Name/Library Code record that has a record ID of 02. The information in this record corresponds to the online Member Moves panel. There must be one Member Name/Library Code record for every member that is to be included in the Move Request.

The Description record indicates that a new Move Request is to be added. The data for the Service Request, Final, Next, and First Run Dates, Move Type, Early Stop level, Special Handling indicator, Short Description and the Expanded Description fields is taken directly from this record.

Validation is performed on these fields. If errors are detected, the Move Request is not added or is added as an invalid Move Request, depending on whether the invalid data was valid at some other time. Examples that cause Move Requests to be added as invalid are:

- Move dates prior to current date
- No expanded description when required by control file option

Note: The MR-VERSION-STAMP must be 03.1 or the entire Move Request is not added to the Pending file.

The Member Name/Library Code record contains the member data information for the Move Request. Use this to specify:

- From Member name on the Test-Level Library
- From User Data for the Test-Level member
- To Member name on the Production Level library
- To User Data for the Production Level member
- Library Code/Subcode associated with the member

Validation is performed on this record to determine if the Library Code listed for the member is found on the Library Code file and to verify that the From or To Member name is not blank. Also, all members must be presented in order by Library Code and To Member name, the same order they occur in a Move Request. If these errors occur, the entire Move Request is not added to the Pending file.

In addition, all online edits are performed. If applicable to the Library Code, Inventory is automatically added and assigned. Member existence exits are called. Online edits that would have prevented you from adding a member online, such as member names not conforming to the minimum and maximum lengths defined in the Library Code, causes the Move Request to be added as invalidated.

APJJ5960 creates a Move Request for each group of Description and Member Name/Library Code records. For Move Requests added as invalidated, you must use the Move Request Online Change option to validate all the input fields. For all Move Requests added, a new Move Request number is assigned.

APJJ5960 also produces the Batch Add-Move-Request Report. This report lists:

- All Move Requests created
- The Move Request's description and member data
- All invalid Library Codes and the member data to which they belong

For an example, see Appendix A, "Reports," in the *CA-PanAPT Reference Guide*.

Move Requests created by this job can be changed and deleted only by the CA-PanAPT online Move Request option.

Input

Input to this job consists of the user-defined Batch Add-Move-Request file. This file is typically created by a user-written program. The data set name must be overridden in APJJ5960, INPUT='<SYSIN>'.

Records on the input file are formed in groups consisting of the Move Request Description and Member Name/Library Code records. The Description record is identified by record type 01, and it must be the first record in the group. The Member Name/Library Code record is identified by record type 02, it must be provided for each Move Request member.

APJJ5960 uses the same record layouts, the Description Record and the Member Data Record, as are produced by the Purge Move Request Job (APJJ5955). These record layouts are provided as members APCCMDES and APCCMMBR in the CA-PanAPT SRCELIB data set. Also, see Appendix C for displays of these members.

The following two tables show the input fields used by APJJ5960 that are taken from input records APCCMDES and APCCMMBR. All other record fields are initialized to blanks or zeros.

Description Record

This record is required. Basic information about the Move Request is described here. All fields listed in the following table except for MR-RECORD-TYPE, correspond to information displayed on the Description of Move Request panel.

See the "Move Requests" chapter in the *CA-PanAPT Reference Guide* for editing requirements for these fields. Unless stated otherwise, all fields can be left blank.

Field Name	Value
MR-RECORD-TYPE	must be 01
MR-VERSION-STAMP	must be 03.1
MR-SERVICE-REQUEST	first character must not be blank
MR-FINAL-MOVE-DATE	initialized to today's date, if blank
MR-NEXT-MOVE-DATE	
MR-SCHEDULED-RUN-DATE	
MR-MOVE-TYPE	initialized to M, if blank
MR-DESCRIPTION	first character must not be blank
MR-EXPANDED-DESCRIPTION	
MR-SPCL-HANDLING	Y or N, if blank, defaults to N
MR-EARLY-STOP-SNAME	the short name for the early stop level, or blank for no early stop.
MR-ADD-USER-ID	APCS5960, if user ID is not specified

Member Data Record

This record is required. There must be one Member Data record for every member that is included in the Move Request. All fields listed in the table below, except for MR-RECORD-TYPE, correspond to information displayed on the Member Moves panel.

See Entering Member Data for Any Library Codes, in the "Move Requests" chapter in the *CA-PanAPT Reference Guide* for the editing requirements for these fields. Unless stated otherwise, all fields can be left blank.

Field Name	Value
MR-RECORD-TYPE	must be 02
MR-VERSION-STAMP	must be 03.1
MR-FROM-MEMBER	if blank, defaults to the MR-TO-MEMBER
MR-FROM-USER-DATA	
MR-TO-MEMBER	if blank, defaults to the MR-FROM-MEMBER
MR-TO-USER-DATA	
MR-LIB-CODE	must be specified
MR-LIBSUBCODE	must be specified, if appropriate

Note: The MR-FROM-MEMBER or the MR-TO-MEMBER field must be specified or the entire Move Request is rejected. If one is left blank the value of the other is copied into it.

Output

APJP5960 produces the Batch Add-MOVE-Request Report. This report lists each Move Request added to the Pending file.

An example of the Batch Add-Move-Request Report can be found in Appendix A, "Reports" in the *CA PanAPT Reference Guide*.

Program

APCS5960

PROC APJP5960

Parameters	Description
INPUT=*x*	*x* must be the name of the input file that contains records used to create a Move Request. Default in PROC: none.
VSAMPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT VSAM files. Change this to your installation's high-level VSAM index. Default in PROC: none.
JCLLIST=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Utility messages, control totals, JCL, and similar information are routed to this destination. Default in PROC: *.
REPORTS=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Reports are routed to this destination. Change this in the JCL to your standards. Default in PROC: *.
DUMPS=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Any SYSUDUMP output is routed to this destination. Default in PROC: *.
SYSDA=*x*	*x* indicates a disk type to be used for the temporary work file. Default in PROC: SYSDA.
PARMS=*x*	*x* is the parameter passed to APCS5960 that specifies which user ID to use in MR-ADD-USER-ID. Valid parameters are *,user ID; *; or user ID. An * (asterisk) uses the user ID in MR-ADD-USER-ID. If MR-ADD-USER-ID is blank, APCS5960 uses the default user ID. The default user ID is the user ID specified in the parameter passed to APCS5960. If the user ID is not specified or is blank, APCS5960 uses APCS5960 for MR-ADD-USER-ID.

Batch Move Request Processing

Program APCS5970 lets you run a batch job to add, change, delete, copy, verify, run a verification procedure, or close a Move Request. This program processes one request at a time.

The batch interface duplicates online ISPF CA-PanAPT functions. The online functions are documented in the "Move Requests" chapter in the *CA-PanAPT Reference Guide*.

Activity

This job reads records from a user-defined batch input file. The record format for the input file, APCCIREQ, is documented in Appendix C. Output consists of a message indicating whether the activity was successful or not. Several activities involve Move Requests. These are as follows:

Adding a Move Request

An authorized CA-PanAPT user adds the initial Move Request, by creating batch input records with all information required for a new Move Request. Information required for adding a Move Request is described under the topic Move Request Maintenance, in the "Move Requests" chapter in the *CA-PanAPT Reference Guide*. The initial status of a new Move Request is **Being Created**.

Changing a Move Request

An authorized user modifies a current Move Request, reviewing and modifying current data.

Normally, Move Requests are Updated while they are in the Being Created (CRE) status. Only authorized users (through the CHGAWAPP activity record) or the CA-PanAPT system administrator are able to do an update against a Move Request that is awaiting approval.

If the Move Request is ready to be moved (no approvals or verifications are outstanding), then no one can change the Move Request. Before you change the Move Request data, change the status of the Move Request to **Being Created**.

Doing a CA-PanAPT Update (CHG function) against a Move Request results in CA-PanAPT revoking all Approvals that have already been granted. In addition, all Verifications are reset.

Closing a Move Request

An authorized user closes the Move Request by using the CA-PanAPT CLOSE function. The Move Request data and contents are now considered to be complete, and the Move Request is now ready to be approved.

At close time, CA-PanAPT verifies that the requirements, specified (during Library Code maintenance processing) in the Library Code definitions, are met. If the Library Code requires it, CA-PanAPT verifies that members are properly assigned, that approved Inventory Records exist, and that the members are acceptable to the Member Existence Exit.

Any Verifications required by the Library Code at the Test level must have already been run successfully for the Move Request. If any Test level Verification requirements remain outstanding, the Move Request is not be closed. A warning message is issued to request that verification be submitted.

Verifying a Move Request

When the Move Request has Verification Procedure requirements, you must run the appropriate procedures to successful completion. The actual Verification Procedure posts the necessary Verification Category indicating success or failure.

Deleting a Move Request

An authorized user can use the CA-PanAPT DEL function to flag a Move Request as DELETED. The Move Request remains on the Pending file, but is no longer included in pending reports or in other current processes.

Move Requests are physically removed from the Pending file by the batch purge job, APJJ5950. See the topic File Maintenance Jobs in the "Batch Component" chapter in the *CA-PanAPT Reference Guide*, for more information on purging Move Requests from the Pending file.

Copying a Move Request

The Move Request Copy function creates a new Move Request based on an existing Move Request.

As part of final processing, CA-PanAPT attempts to Assign, Retrieve, and Verify the existence of the members as specified in the Library Code definitions.

Input

The input to this job consists of a user-defined batch input file. This file is a variable length spanned sequential data set with a record length of 20000. This file is typically created by a user-written program. Copybook APCCIREQ, documented in Appendix C describes the input record layout. Validation is performed in CA-PanAPT as it is by the corresponding ISPF function. If an error is detected, the Move Request is not updated, and error messages are sent to the output file.

APJJ5970 below is sample JCL that executes program APCS5970:

```
//APJJ5970 JOB (),'BATCH UPDATE',
 //                    TYPRUN=HOLD,
 //                    CLASS=P,
 //                    MSGCLASS=J
//*
//JOBLIB  DD  DSN=<SPFX>,LOADLIB,DISP=SHR
//*
//*
//APJP5970 EXEC APJP5970,
 //             VSAMPFX='<VPFX>',    <====  VSAM data set prefix
//             SOFTPFX='<SPFX>',    <====  APTMODEL prefix
//             APTIN='<SYSIN>',     <====  Input data set name
//             APTOUT='<SYSOUT>'    <====  Output data set name
```

Note: The data set names (APTIN='<SYSIN>') and (APTOUT='<SYSOUT>') must be overridden in APJJ5970.

The value specified in the APCSIREQ-PROCESS field determines what type of request is run against the Move Request. The table shown next documents required values:

Request Type	APCSIREQ-PROCESS value
Add	'A'
Change	'G'
Copy	'O'
Delete	'X'

Request Type	APCSIREQ-PROCESS value
Close	'C'
Verify	'V'
Run verification procedure	'R'

Note: All other APCSIREQ-PROCESS functions are for CA internal processing only. If you try to use them for batch functions, unpredictable results occur.

The fields in the following table must be modified as indicated in APCSIREQ:

Field	Request Type	Value
APCSIREQ-PROCESS	all	'A', 'G', 'O', 'X', 'C', 'V' or 'R'
APCSIREQ-RECORD-LEN	all	The length of the record.
APCSIREQ-RECORD-TYPE	all	'00'
APCSIREQ-USER-ID	all	8 bytes
APCSIREQ-VERSION-STAMP	all	03.1
APCSIREQ-BATCH-INPUT	A, G	Y
APCSIREQ-CKOT-JOB1	A, G, O	The Retrieve job card image (at least one is required, maximum allowed is four).
APCSIREQ-CKOT-JOB2	A, G, O	The Retrieve job card image (at least one is required, maximum allowed is four).
APCSIREQ-CKOT-JOB3	A, G, O	The Retrieve job card image (at least one is required, maximum allowed is four).
APCSIREQ-CKOT-JOB4	A, G, O	The Retrieve job card image (at least one is required, maximum allowed is four).

Field	Request Type	Value
APCSIREQ-NUMBER	G, O, X, C, V	The move request number.
APCSIREQ-VP-NUM	R	Multiples can be selected.
APCSIREQ-VP-JOB1	R	The verification job card (at least one is required, maximum allowed is four).
APCSIREQ-VP-JOB2	R	The verification job card (at least one is required, maximum allowed is four).
APCSIREQ-VP-JOB3	R	The verification job card (at least one is required, maximum allowed is four).
APCSIREQ-VP-JOB4	R	The verification job card (at least one is required, maximum allowed is four).

In addition to the fields documented in the previous two tables for the APCCIREQ copybook, the description record (record type 01) and member data record (record type 02) are required for the add and the change request. See Description Record and Member Data Record under section APCS5960, Add Move Requests to the Pending File.

Valid MR-PROCESS field values of the Member Data Record for the change request are documented in the following table.

Value	Description
'A'	Add the member.
'G'	Change the member
'X'	Delete the member.
'S'	Assign the member.

Value	Description
'K'	Retrieve the member.

For a change request, Retrieve is performed when the auto-Retrieve flag is on. To Retrieve a member, MR-PROCESS of the member data record must be set to 'K'. Assign is performed if the member is not already assigned. Each of the actions in the table above is described in more detail under the topics Entering Member Data for Any Library Codes and Line Command Processing in the "Move Requests" chapter in the *CA-PanAPT Reference Guide*.

Output

The output file is a variable length spanned sequential data set with a record length of 20000.

Output from the Batch Interface is sent to the information message record. The messages are sent to the APT-MESSAGE-REC field in copybook APCCMMSG (see Appendix C) The APTMSG-REC-TYPE is equal to: '07'.

Program

APCS5970

PROC APJP5970

Parameters	Description
APTIN=*x*	*x* must be the name of the input file that contains records used to update a Move Request. Default in PROC: none.
APTOUT=*x*	*x* must be the name of the output file that contains message records. Default in PROC: none.
VSAMPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT VSAM files. Change this to your installation's high-level VSAM index. Default in PROC: none.

Parameters	Description
SOFTPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT files. Change this to your site's high-level software library index.
DUMPS=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Any SYSUDUMP output is routed to this destination. Default in PROC: *.
SYSDA=*x*	*x* indicates a disk type to be used for the temporary work file. Default in PROC: SYSDA.
TRK=*x*	*x* indicates the space allocation unit. Default in PROC: TRK.
TRK1=*x*	*x* indicates the first space allocation. Default in PROC: 10.
TRK2=*x*	*x* indicates the second space allocation. Default in PROC: 10.

LIBCODE Extract Facility

Program APCS5970 provides a means to write LIBCODE records from the CA-PanAPT database to a sequential file.

The record layout of the sequential file produced is described in APCCLIB2 found in the CA-PanAPT SRCELIB. See Appendix C for a listing of this member.

Activity

An authorized CA-PanAPT user can dump all LIBCODE records of the database by supplying one batch input record.

Input

Copybook APCCIREQ, documented in Appendix C describes the input record layout.

Initialize the APCCIREQ record to spaces and insert the following values:

Field	Value
APCSIREQ-RECORD-LEN	Length of this record
APCSIREQ-RECORD-TYPE	'00'
APCSIREQ-NUMBER	'0'
APCSIREQ-VERSION-STAMP	'03.1'
APCSIREQ-PROCESS	'B'
APCSIREQ-BATCH-INPUT	'Y'

Output

When the LIBCODE extract is successful, the extract records are mapped by the APCCLIB2 copybook found in Appendix C and sent to the file allocated with the 'OUTPUT' DDNAME. If the LIBCODE extract has been unsuccessful, then informational and error messages are found in that same file. These messages are mapped by the APCCMMSG copybook, also found in Appendix C.

Program

APCS5970

Validation is performed as with any ISPF, CA-PanAPT function. If an error is detected in the input stream, then the extract is not generated and error messages are produced in its place.

APJJ5970 below is sample JCL that executes program APCS5970.

```
//APJJ5970 JOB (),'BATCH UPDATE',
 //          TYPRUN=HOLD,
 //          CLASS=P,
 //          MSGCLASS=J
//*
//JOBLIB  DD  DSN=<SPFX>,LOADLIB,DISP=SHR
//*
//*
//APJP5970 EXEC APJP5970,
 //             VSAMPFX='<VPFX>',    <====  VSAM data set prefix
//             SOFTPFX='<SPFX>',    <====  APTMODEL prefix
//             APTIN='<SYSIN>',     <====  Input data set name
//             APTOUT='<SYSOUT>'    <====  Output data set name
```

The input to this job consists of the user-defined batch input file. This job is typically created by a user-written program. The data set name (APTIN='<SYSIN>') must be overridden in APJJ5970. Records on the input file consist of the APCSIREQ request record, the record type is 00 for the extract function. Only one input record is needed, and its format is described earlier in the section Input.

PROC APJP5970

Parameters	Description
APTIN=*x*	*x* must be the name of the input file that contains records used to update a Move Request. Default in PROC: none.
APTOUT=*x*	*x* must be the name of the output file that contains message records. Default in PROC: none.

Parameters	Description
VSAMPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT VSAM files. Change this to your installation's high-level VSAM index. Default in PROC: none.
SOFTPFX=*x*	*x* is concatenated ahead of the standard data set name for CA-PanAPT files. Change this to your site's high-level software library index.
DUMPS=*x*	*x* is a SYSOUT class and any other SYSOUT parameters. Any SYSUDUMP output is routed to this destination. Default in PROC: *.
SYSDA=*x*	*x* indicates a disk type to be used for the temporary work file. Default in PROC: SYSDA.
TRK=*x*	*x* indicates the space allocation unit. Default in PROC: TRK.
TRK1=*x*	*x* indicates the first space allocation. Default in PROC: 10.
TRK2=*x*	*x* indicates the second space allocation. Default in PROC: 10.

This job reads records from the input batch data set. It should have a request record (record type 00) with APCSIREQ-PROCESS set to 'B'.

Appendix A: Forms

This section contains the following topics:

CA-PanAPT Forms

This appendix contains forms to help you make decisions when implementing CA-PanAPT. These forms are referenced throughout this guide.

Approval Category Set-up Form

Item Type (JCL/source/...)	Description of Approval	Approvals Required

APPROVAL CATEGORY SET-UP FORM

Page 2 - Approval Category Authority

View the information contained on this form using report APCS5104-01.

CA-PanAPT Approval Number	Approval Category Description	TSO User IDs of Approvers			
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______
_____	______________________________	_______	______	_______	_______

Control File Security Form

CONTROL FILE SECURITY FORM (Page 1 of 4)

This form helps you to determine which types of users will be allowed to perform each CA-PanAPT activity. View the information on this form using the APCS5103-01 report.

Description of Activity	Internal Activity Code	Allow to: Any User	Sharing Group	Owner	Group Admin	Oper.	These Groups/ User IDs		Security Exit Status
CTL FILE - ACTIVITIES									
Change	(CTLACT /CHG)								N
Inquire	(CTLACT /INQ)								N
CTL FILE - USERIDS									
Add	(CTLUSER /ADD)								N
Change	(CTLUSER /CHG)								N
Delete	(CTLUSER /DEL)								N
Inquire	(CTLUSER /INQ)								N
CTL FILE - SYSTEM INFO									
Add	(CTLSYS /ADD)								N
Change	(CTLSYS /CHG)								N
Delete	(CTLSYS /DEL)								N
Inquire	(CTLSYS /INQ)								N
INVENTORY FILE									
Retrieve	(INVENTRY/ACK)								N
Add	(INVENTRY/ADD)								N
Approve	(INVENTRY/APP)								N
Assign	(INVENTRY/ASN)								N
Change	(INVENTRY/CHG)								N
Delete	(INVENTRY/DEL)								N
Inquire	(INVENTRY/INQ)								N
Release	(INVENTRY/REL)								N
Transfer	(INVENTRY/TRN)								N

CONTROL FILE SECURITY FORM (Page 3 of 4)

Description of Activity	Internal Activity Code	Allow to: Any User	Sharing Group	Owner	Group Admin	Oper.	These Groups/ User IDs		Security Exit Status
Approve ___ Move	(MOVEREQ / APM - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____
Approve ___ Backout	(MOVEREQ/ APB - ___)	_____	_____	_____	_____	_____	_____	_____	_____

User ID Set-up Form

USER ID SET-UP FORM

This form helps you to determine the characteristics to give each CA-PanAPT user. View the information on this form using report APCS5103-01.

TSO User ID	Approval Capabilities	CA-PanAPT Admin (Y/N)	Oper. (Y/N)	Group Admin (Y/N)	Group(s) (Mark Group Admins with '*')			

Group Set-up Form

GROUP SET-UP FORM

This form helps you to determine which groups to assign users in various departments or areas.

Name of Department or Area	CA-PanAPT Group Name

Library Code Set-up Form

LIBRARY CODE SET-UP FORM (Page 1 of 7)

This form helps you to determine which characteristics to give each Library Code. Make one copy of all 7 pages of this form for each Library Code and an additional copy of pages 5 and 6 for each Move level defined to your system beyond the starting point test level. View the information on this form using reports APCS5102-01 and APCS5102-02.

Note: There are additional Library Code Set-up forms in the documentation for other CA-PanAPT options, such as the DB2 Option.

See the "Library Codes" chapter in the *CA-PanAPT Reference Guide* for more information about the fields on this form.

Library Code: __ __ __ __ / __ __ __

Description: _____

From/To names *may* / *must* / *may not* be equal. (Circle one.)

Lengths allowed (1-10): Minimum: ____ Maximum: ____

User-data lengths (0-8): From data Minimum: ____ Maximum: ____

To data Minimum: ____ Maximum: ____

LIBRARY CODE SET-UP FORM (Page 2 of 7)

Inventory Processing Options:

```
        Inventory enabled?  . . . . . . . . . . . . . . . . . . . . . . . . yes   no

        Inventory Qualifier:  ____________

        Auto create at assignment time: . . . . . . . . . . . . . . . . . . yes   no

        Auto approve at creation? . . . . . . . . . . . . . . . . . . . . . yes   no

        Require approved inventory records? . . . . . . . . . . . . . . . . yes   no

        Edit exit program: _________________________  parameter: ______________________


Assignment Processing Options:

        Assignment enabled? . . . . . . . . . . . . . . . . . . . . . . . . yes   no

        Auto Assignment with Move Request?  . . . . . . . . . . . . . . . . yes   no

        Auto Release at Prod move time? . . . . . . . . . . . . . . . . . . yes   no

        Retrieve enabled? . . . . . . . . . . . . . . . . . . . . . . . . . yes   no

        Auto Retrieve at assignment . . . . . . . . . . . . . . . . . . . . yes   no

LIBRARY CODE SET-UP FORM (Page 3 of 7)

Inventory Defaults:

        Owned by:_____________________________________________
```

```
        Description:_____________________________________________________________

        Comments:________________________________________________________________

        Environment:_____________________________________

        Application:_____________________________________

        Language:________________________________________

        Compiler Options:________________________________________________________

        Link Edit Options:_______________________________________________________

        User0001:_____________   User0006:_____________

        User0002:_____________   User0007:_____________

        User0003:_____________   User0008:_____________

        User0004:_____________   User0009:_____________

        User0005:_____________   User0010:_____________

        User0011:________________________________________________________________
_

        User0012:________________________________________________________________
_

        User0013:________________________________________________________________
_

        User0014:________________________________________________________________
_

        User0015:________________________________________________________________
_

        User0016:________________________________________________________________
_
```

```
        User0017:_____________________________________________________________
_

        User0018:_____________________________________________________________
_

        User0019:_____________________________________________________________
_

        User0020:_____________________________________________________________
_

LIBRARY CODE SET-UP FORM (Page 4 of 7)

Starting Test Level Details
```

```
Dataset Name:_____________________________________________________________________

DD Name:___________________

Security:_______________

Access Method (Circle one)          PV (CA-Panvalet)            L
(CA-Librarian)

                      PX (CA-Panexec)             PO (PDS)

                      TL (CA-Telon TDF)

                      Other ____ (1 to 2 characters)

Member existence exit: (Circle one)

       =   (Default if Access Method is PV, L, PX, or PO)

       APAS0222 (CA-Panvalet)       APCS0221 (CA-Librarian)

       APAS0223 (CA-Panexec)        APAS0226 (CA-Telon)          APAS0200 (PDS)

       _____________ (Other)

Member existence exit parm: _____________________________________________________
       (Of the exits listed above, only APAS0226 (CA-Telon) uses a parm.)

Member Select List exit: (Circle one)

       =   (Default if Access Method is PV, L, or PO)

       APAS0600 (CA-Panvalet) APCS0620 (CA-Librarian)

       APAS0610 (PDS)               _____________ (Other)
```

```
Member Select List exit parm: _____________________________________________________
       (None of the exits listed above use a parm.)

Verifications required before Close (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
LIBRARY CODE SET-UP FORM (Page 5 of 7)

Make as many copies of this page as you have target move levels in your system.


Destination _____ Level Details


Target:
    Dataset Name:_____________________________________________________________

    DD Name:_________________   Security:_________________

    Access Method (Circle one).      PV  L  PX  PO  TL  Other ___ (1 to 2 characters)


Backup:
    Dataset Name:_____________________________________________________________

    DD Name:_________________   Security:_________________

    Access Method (Circle one).      PV  L  PX  PO  TL  Other ___ (1 to 2 characters)


Backout:
    Dataset Name:_____________________________________________________________

    DD Name:_________________   Security:_________________

    Access Method (Circle one).      PV  L  PX  PO  TL  Other ___ (1 to 2 characters)


Member existence exit: (Circle one)

       =   (Default if Access Method is PV, L, PX, or PO)
```

```
        APAS0222 (CA-Panvalet)        APCS0221 (CA-Librarian)

        APAS0223 (CA-Panexec)         APAS0226 (CA-Telon)          APAS0200 (PDS)

        _________________ (Other)

Move Control (how are members moved into the target dataset, circle one)

        Copy   /   Move and delete   /   Delete without moving (use with caution)

Backup Enabled (Circle one if a Backup Dataset is filled in):   Yes   /   No

Backout Control (Circle one if a Backup Dataset is filled in, controls how members are
backed out)

        Backout and restore   /   Restore without backout   /   Prohibited

LIBRARY CODE SET-UP FORM (Page 6 of 7)

Make as many copies of this page as you have target move levels in your system.

Destination _____ Level Details (continued)

Member existence exit parm: ____________________________________________________
        (Of the exits listed above, only APAS0226 (CA-Telon) uses a parm.)

Member Select List exit: (Circle one)

        =   (Default if Access Method is PV, L, or PO)

        APAS0600 (CA-Panvalet)        APCS0620 (CA-Librarian)

        APAS0610 (PDS)                _________________ (Other)

Member Select List exit parm: ____________________________________________________
```

```
        (None of the exits listed above use a parm.)


Verifications required before Move (Circle as many as desired):

        1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20


Approvals required before Move (Circle as many as desired):

        1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20


Approvals required before Backout (Circle as many as desired):

        1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
LIBRARY CODE SET-UP FORM (Page 7 of 7)

Modeling

Model Base (1 to 4 characters):  ________
(The Model Base controls the order in which Library Codes are presented to Move Modeling.
If this is not important, leave it blank and it will default).

Move Model Specification(s):

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
```

```
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

Retrieve Model Specification(s):

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
```

____

_____

_____

_____

_____

_____

CA-PanAPT DB2 Option Forms

DB2 Library Code Set-up Form

DB2 LIBRARY CODE SET-UP FORM (Page 1 of 7)

This form helps you to determine which characteristics to give each Library Code. Make one copy of all seven pages of this form for each Library Code and an additional copy of pages 5 and 6 for each Move level defined to your system beyond the starting point test level. View the information on this form using reports APCS5102-01 and APCS5102-02.

```
Library Code:  __ __ __ __ / __ __ __

Description:  ____________________________________________________________

DB2 Type (circle one):     Source     DBRM     Source/DBRM

From/To names   may  /  must  /  may not  be equal. (Circle one.)

Lengths allowed (1-10):             Minimum: ____        Maximum: ____

User-data lengths (0-8):  From data Minimum: ____        Maximum: ____

                            To data Minimum: ____        Maximum: ____
```

DB2 LIBRARY CODE SET-UP FORM (Page 2 of 7)

```
Inventory Processing Options:

       Inventory enabled?  . . . . . . . . . . . . . . . . . . . . . . . yes no
       Inventory Qualifier:  ____________
       Auto create at assignment time: . . . . . . . . . . . . . . . . . yes no
       Auto approve at creation? . . . . . . . . . . . . . . . . . . . . yes no
       Require approved inventory records? . . . . . . . . . . . . . . . yes no
       Edit exit program:  ________________________  parameter:  ____________________

Assignment Processing Options:

       Assignment enabled? . . . . . . . . . . . . . . . . . . . . . . . yes no
```

```
        Auto Assignment with Move Request?  . . . . . . . . . . . . . . . . . yes no
        Auto Release at Prod move time? . . . . . . . . . . . . . . . . . . . yes no
        Retrieve enabled? . . . . . . . . . . . . . . . . . . . . . . . . . . yes no
        Auto Retrieve at assignment . . . . . . . . . . . . . . . . . . . . . yes no

DB2 LIBRARY CODE SET-UP FORM (Page 3 of 7)

Inventory Defaults:

        Owned by:_____________________________________________

        Description:__________________________________________________________

        Comments:_____________________________________________________________

        Environment:__________________________________________

        Application:__________________________________________

        Language:_____________________________________________

        Compiler Options:_____________________________________________________

        Link Edit Options:____________________________________________________

        User0001:_______________   User0006:_______________

        User0002:_______________   User0007:_______________

        User0003:_______________   User0008:_______________

        User0004:_______________   User0009:_______________

        User0005:_______________   User0010:_______________

        User0011:_____________________________________________________________
_

        User0012:_____________________________________________________________
_

        User0013:_____________________________________________________________
```

```
_

        User0014:_____________________________________________________________
_

        User0015:_____________________________________________________________
_

        User0016:_____________________________________________________________
_

        User0017:_____________________________________________________________
_

        User0018:_____________________________________________________________
_

        User0019:_____________________________________________________________
_

        User0020:_____________________________________________________________
_

DB2 LIBRARY CODE SET-UP FORM (Page 4 of 7)

Starting Test Level Details


Dataset Name:_________________________________________________________________


DD Name:___________________


Security:__________________


Access Method (Circle one)           PV (CA-Panvalet)             L
(CA-Librarian)

                      PX (CA-Panexec)              PO (PDS)
```

```
                    TL (CA-Telon TDF)

                    Other ____ (1 to 2 characters)

Member existence exit: (Circle one)

       =   (Default if Access Method is PV, L, PX, or PO)

       APAS0222 (CA-Panvalet)       APCS0221 (CA-Librarian)

       APAS0223 (CA-Panexec)        APAS0226 (CA-Telon)          APAS0200 (PDS)

       _____________ (Other)


Member existence exit parm: _____________________________________________________
       (Of the exits listed above, only APAS0226 (CA-Telon) uses a parm.)


Member Select List exit: (Circle one)

       =   (Default if Access Method is PV, L, or PO)

       APAS0600 (CA-Panvalet) APCS0620 (CA-Librarian)

       APAS0610 (PDS)               _____________ (Other)

Member Select List exit parm: ____________________________________________________
       (None of the exits listed above use a parm.)

Verifications required before Close (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
DB2 LIBRARY CODE SET-UP FORM (Page 5 of 7)

Make as many copies of this page as you have target move levels in your system.


Destination _____ Level Details
```

```
Target:
    Dataset Name:_____________________________________________________________

    DD Name:_________________   Security:_________________

    Access Method (Circle one).      PV  L  PX  PO  TL  Other ___ (1 to 2 characters)


Backup:
    Dataset Name:_____________________________________________________________

    DD Name:_________________   Security:_________________

    Access Method (Circle one).      PV  L  PX  PO  TL  Other ___ (1 to 2 characters)


Backout:
    Dataset Name:_____________________________________________________________

    DD Name:_________________   Security:_________________

    Access Method (Circle one).      PV  L  PX  PO  TL  Other ___ (1 to 2 characters)


Member existence exit: (Circle one)

       =   (Default if Access Method is PV, L, PX, or PO)

       APAS0222 (CA-Panvalet)        APCS0221 (CA-Librarian)

       APAS0223 (CA-Panexec)         APAS0226 (CA-Telon)          APAS0200 (PDS)

       _________________ (Other)


Move Control (how are members moved into the target dataset, circle one)

       Copy   /   Move and delete   /   Delete without moving (use with caution)


Backup Enabled (Circle one if a Backup Dataset is filled in):   Yes   /   No
```

Backout Control (Circle one if a Backup Dataset is filled in, controls how members are backed out)

Backout and restore / Restore without backout / Prohibited

DB2 LIBRARY CODE SET-UP FORM (Page 6 of 7)

Make as many copies of this page as you have target move levels in your system.

Destination _____ Level Details (continued)

Member existence exit parm: _____
(Of the exits listed above, only APAS0226 (CA-Telon) uses a parm.)

Member Select List exit: (Circle one)

= (Default if Access Method is PV, L, or PO)

APAS0600 (CA-Panvalet) APCS0620 (CA-Librarian)

APAS0610 (PDS) _____ (Other)

Member Select List exit parm: _____
(None of the exits listed above use a parm.)

Verifications required before Move (Circle as many as desired):

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Approvals required before Move (Circle as many as desired):

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

```
Approvals required before Backout (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20

DB2 LIBRARY CODE SET-UP FORM (Page 7 of 7)

Modeling

Model Base (1 to 4 characters):  ________
(The Model Base controls the order in which Library Codes are presented to Move Modeling.
If this is not important, leave it blank and it will default).

Move Model Specification(s):

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____
```

```
--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

Retrieve Model Specification(s):

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----
```

_____

_____

DB2 Package Library Code Set-up Form

DB2 PACKAGE LIBRARY CODE SET-UP FORM (Page 1 of 6)

This form helps you to determine which characteristics to give each Library Code. Make one copy of all six pages of this form for each Library Code and an additional copy of page 5 for each Move level defined to your system beyond the starting point test level. View the information on this form using reports APCS5102-01 and APCS5102-02.

Library Code: __ __ __ __ / __ __ __

Description: _____

DB2 Type: PACKAGE

From/To names *may* / *must* / *may not* be equal. (Circle one.)

Lengths allowed (1-8): Minimum: ____ Maximum: ____

User-data lengths (0-8): From data Minimum: ____ Maximum: ____

```
                          To data Minimum: ____        Maximum: ____

DB2 PACKAGE LIBRARY CODE SET-UP FORM (Page 2 of 6)

Inventory Processing Options:

        Inventory enabled?  . . . . . . . . . . . . . . . . . . . . . . . . yes   no

        Inventory Qualifier:  ____________

        Auto create at assignment time: . . . . . . . . . . . . . . . . . . yes   no

        Auto approve at creation? . . . . . . . . . . . . . . . . . . . . . yes   no

        Require approved inventory records? . . . . . . . . . . . . . . . . yes   no

        Edit exit program: _____________________  parameter: _____________________


Assignment Processing Options:

        Assignment enabled? . . . . . . . . . . . . . . . . . . . . . . . . yes   no

        Auto Assignment with Move Request?  . . . . . . . . . . . . . . . . yes   no

        Auto Release at Prod move time? . . . . . . . . . . . . . . . . . . yes   no

        Retrieve enabled? . . . . . . . . . . . . . . . . . . . . . . . . . yes   no
```

```
        Auto Retrieve at assignment . . . . . . . . . . . . . . . . . . . . yes   no

DB2 PACKAGE LIBRARY CODE SET-UP FORM (Page 3 of 6)

Inventory Defaults:

        Owned by:_____________________________________________

        Description:_____________________________________________________________

        Comments:________________________________________________________________

        Environment:_____________________________________________

        Application:_____________________________________________

        Language:________________________________________________

        Compiler Options:________________________________________________________

        Link Edit Options:_______________________________________________________

        User0001:_______________   User0006:_______________

        User0002:_______________   User0007:_______________

        User0003:_______________   User0008:_______________

        User0004:_______________   User0009:_______________

        User0005:_______________   User0010:_______________

        User0011:________________________________________________________________
_

        User0012:________________________________________________________________
_

        User0013:________________________________________________________________
_
```

```
        User0014:_____________________________________________________________
_
        User0015:_____________________________________________________________
_
        User0016:_____________________________________________________________
_
        User0017:_____________________________________________________________
_
        User0018:_____________________________________________________________
_
        User0019:_____________________________________________________________
_
        User0020:_____________________________________________________________
_
DB2 PACKAGE LIBRARY CODE SET-UP FORM (Page 4 of 6)

Starting Test Level Details

DB2 Subsystem ID (1-4 characters):________

DB2 Owner (1-8 characters):________________

DB2 Qualifier (1-8 characters):________________

DB2 Collection ID (1-18 characters):___________________________________

Security (typically blank):________________
```

```
Access Method (0-2 characters, typically blank):____

Member existence exit: (Circle one)  APCS0227   or   None

Member existence exit parm: (Circle one)     APCS0227   or   _________________

Member Select List exit: (typically blank):_________________

Member Select List exit parm (typically blank):_____________________________________

Verifications required before Close (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
DB2 PACKAGE LIBRARY CODE SET-UP FORM (Page 5 of 6)

Make as many copies of this page as you have target move levels in your system.

Destination _____ Level Details

Target:
    Subsystem ID (1-4 characters):________

    Owner (1-8 characters):_________________

    Qualifier (1-8 characters):_________________

    Collection ID (1-18 characters):_____________________________________

    Security (typically blank):_________________

    Access Method (0-2 characters, typically blank):  ____

Move Control:  Copy
```

```
Backout Control (Circle one): Restore without backout   /   Prohibited

Member existence exit: (Circle one)  APCS0227   or   None

Member existence exit parm: (Circle one)     APCS0227   or   _________________

Member Select List exit: (typically blank):_________________

Member Select List exit parm (typically blank):_____________________________________

Verifications required before Move (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20

Approvals required before Move (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20

Approvals required before Backout (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20

DB2 PACKAGE LIBRARY CODE SET-UP FORM (Page 6 of 6)

Modeling

Model Base (1 to 4 characters):  ________
(The Model Base controls the order in which Library Codes are presented to Move Modeling.
If this is not important, leave it blank and it will default).

Move Model Specification(s):

_____________________________________________________________________________
```

```
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

Retrieve Model Specification(s):

_____________________________________________________________________________
____

_____________________________________________________________________________
```

____

_____

_____

_____

_____

_____

_____

_____

_____

_____

DB2 Plan Library Code Set-up Form

DB2 PLAN LIBRARY CODE SET-UP FORM (Page 1 of 6)

This form helps you to determine which characteristics to give each Library Code. Make one copy of all six pages of this form for each Library Code and an additional copy of page 5 for each Move level defined to your system beyond the starting point test level. View the information on this form using reports APCS5102-01 and APCS5102-02

```
Library Code:  __ __ __ __ / __ __ __

Description:   _____________________


DB2 Type:   PLAN

From/To names  may / must / may not be equal. (Circle one.)

Lengths allowed (1-8):              Minimum: ____        Maximum: ____

User-data lengths (0-8):  From data Minimum: ____        Maximum: ____

                            To data Minimum: ____        Maximum: ____


DB2 PLAN LIBRARY CODE SET-UP FORM (Page 2 of 6)

Inventory Processing Options:

        Inventory enabled?  . . . . . . . . . . . . . . . . . . . . . . . . yes   no

        Inventory Qualifier:  ____________

        Auto create at assignment time: . . . . . . . . . . . . . . . . . . yes   no

        Auto approve at creation? . . . . . . . . . . . . . . . . . . . . . yes   no

        Require approved inventory records? . . . . . . . . . . . . . . . . yes   no
```

```
        Edit exit program: _________________________  parameter: ____________________


Assignment Processing Options:

        Assignment enabled? . . . . . . . . . . . . . . . . . . . . . . . . yes   no

        Auto Assignment with Move Request?  . . . . . . . . . . . . . . . . yes   no

        Auto Release at Prod move time? . . . . . . . . . . . . . . . . . . yes   no

        Retrieve enabled? . . . . . . . . . . . . . . . . . . . . . . . . . yes   no

        Auto Retrieve at assignment . . . . . . . . . . . . . . . . . . . . yes   no
DB2 PLAN LIBRARY CODE SET-UP FORM (Page 3 of 6)
Inventory Defaults:

        Owned by:_____________________________________________

        Description:_____________________________________________________________

        Comments:________________________________________________________________

        Environment:_____________________________________

        Application:_____________________________________

        Language:________________________________________

        Compiler Options:________________________________________________________

        Link Edit Options:_______________________________________________________
```

```
        User0001:_________________   User0006:_________________

        User0002:_________________   User0007:_________________

        User0003:_________________   User0008:_________________

        User0004:_________________   User0009:_________________

        User0005:_________________   User0010:_________________

        User0011:_____________________________________________________________
_

        User0012:_____________________________________________________________
_

        User0013:_____________________________________________________________
_

        User0014:_____________________________________________________________
_

        User0015:_____________________________________________________________
_

        User0016:_____________________________________________________________
_

        User0017:_____________________________________________________________
_

        User0018:_____________________________________________________________
_

        User0019:_____________________________________________________________
_

        User0020:_____________________________________________________________
_
```

```
DB2 PLAN LIBRARY CODE SET-UP FORM (Page 4 of 6)

Starting Test Level Details


DB2 Subsystem ID (1-4 characters):________


DB2 Owner (1-8 characters):_________________


DB2 Qualifier (1-8 characters):_________________


Security (typically blank):_________________


Access Method (0-2 characters, typically blank):____


Member existence exit: (Circle one)  APCS0225   or   None


Member existence exit parm: (Circle one)     APCS0225   or   _________________


Member Select List exit: (typically blank):_________________


Member Select List exit parm (typically blank):_____________________________________


Verifications required before Close (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
DB2 PLAN LIBRARY CODE SET-UP FORM (Page 5 of 6)

Make as many copies of this page as you have target move levels in your system.

Destination _____ Level Details
```

```
Target:
    Subsystem ID (1-4 characters):________

    Owner (1-8 characters):_________________

    Qualifier (1-8 characters):_________________

    Security (typically blank):_________________

    Access Method (0-2 characters, typically blank):  ____


Move Control:  Copy

Backout Control (Circle one): Restore without backout   /   Prohibited

Member existence exit: (Circle one)  APCS0225   or   None

Member existence exit parm: (Circle one)     APCS0225   or   _________________

Member Select List exit: (typically blank):_________________

Member Select List exit parm (typically blank):_____________________________________

Verifications required before Move (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20

Approvals required before Move (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
```

```
Approvals required before Backout (Circle as many as desired):

       1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20

DB2 PLAN LIBRARY CODE SET-UP FORM (Page 6 of 6)


Modeling

Model Base (1 to 4 characters):  ________
(The Model Base controls the order in which Library Codes are presented to Move Modeling.
If this is not important, leave it blank and it will default).

Move Model Specification(s):

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____

_____________________________________________________________________________
____
```

```
--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

Retrieve Model Specification(s):

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----

--------------------------------------------------------------------------------
----
```

_____

_____

Appendix B: User Exits

This section contains the following topics:

About User Exits

CA-PanAPT provides these types of user exits:

- Member Existence Exit
- Inventory Edit Exit
- MSL Exit programs
- Security Exit

You specify all four exits. Exits one through three are specified for each Library Code on the panels in Library Code maintenance. The fourth exit is the Security Exit. It is specified on the Control file.

This appendix describes the parameters passed to each type of exit and the return codes the exits use to communicate with CA-PanAPT. This is the information you need to write your own exits.

The sample exits provided with CA-PanAPT are provided in source form on the APTEXSRC data set. The ones with member name prefixes of APAS are written in Assembler language. The ones with prefixes of APCS are written in LE/390 COBOL.

The LE/390 COBOL compiler options, DYNAM,DATA (24), should be used when compiling LE/390 COBOL exits for CA-PanAPT.

Member Existence Exits

Member Existence Exits can be called when members are added to a Move Request, when the Move Request is changed or closed, when a Move Request is being set up for a Back Out request, and when a Move Request is selected by APCS5310 for move processing. These exits provide an installation-defined check on members of a Move Request. They exist primarily to ensure that a member exists before CA-PanAPT attempts to move it. The exits are as follows:

- APAS0200 for members of PDS libraries
- APAS0221 for members of CA-Librarian masters
- APAS0222 for members of CA-Panvalet libraries
- APAS0223 for members of CA-Panexec libraries
- APAS0226 for members of CA-Telon TDFs

Inventory Edit Exit

The Inventory Edit Exit (APCS0304) inspects Inventory Records as they are maintained using the CA-PanAPT Inventory Maintenance panels. This sample exit is provided in both source and executable form.

MSL Exit

The MSL Exit can be called when MSL is requested doing Move Request maintenance. The purpose of these exits is to return the names of members that exist in a library. The exits provided with CA-PanAPT are:

- APAS0600 for CA-Panvalet
- APAS0610 for PDS libraries
- APAS0620 for CA-Librarian

Security Exit

The Security Exit (ASCS0401) permits you to control the use of certain CA-PanAPT operations.

Member Existence Exit Interface

The Member Existence Exit serves two purposes. The primary purpose is to verify that a member to be moved exists on the sending library. If the daily move job cannot find the member when it is time to move it, part of the processing fails and some Move Requests are not completed in a timely manner.

The Verify flag for each member of a Move Request displays the result of the most recent execution of the exit. You can view these flags on the Member Moves panel during Move Request Add, Change, or Inquire activities.

In addition to verifying member existence, the Member Existence Exit is also used to remove a member from the test library when a purge of the member is requested. Of the exits distributed with CA-PanAPT, only the PDS Member Existence Exit supports purging members.

CA-PanAPT provides several sample Member Existence Exits. You can use the executable forms of these exits, modify the source, and compile them to make new exits, or you can create your own exits. If you do modify the sample exits, select a new name for your new exit and leave the original for future comparisons. Users can then recognize immediately that the new exit is not identical to the sample described in CA-PanAPT guides because it has a different name. The sample CA-PanAPT member existence exits are in the following table.

Exit name	Library Type
APAS0200	PDS
APCS0221	CA-Librarian
APAS0222	CA-Panvalet

Exit name	Library Type
APAS0223	CA-Panexec
APAS0226	CA-Telon

Note: For CA-Librarian, CA-Panvalet, and CA-Panexec member existence exits, their load libraries must be available both to CA-PanAPT online and to job APJJ5310. When using the member existence exit for CA-Panexec members, CA-Panexec PTF 1005604 is required.

Exit Calls

You specify in the Library Code definition which member existence exit CA-PanAPT calls for members of that Library Code. If the Library Code does not specify a member existence exit, CA-PanAPT does not call an exit for members of that Library Code.

CA-PanAPT calls member existence exits at several points during online and batch Move Request processing. Each point in the Move Process from which an exit is called is detailed below.

A field in the Exit Parameter Structure contains a code that indicates which of the following processes were active when the exit was called. An exit can use the value of this code to determine processing requirements.

Note: The exit is called only when the associated Library Code has specified an EXIT for the appropriate level.

Exit Codes

The following chart summarizes the codes passed to the Member Existence Exits:

Code	Process	Description
1	MAKECOPY	Copy a Move Request

Code	Process	Description
2	PRECHANGE	Prior to displaying the Member Moves panel for a change
3	POSTCOPY	After all changes have been completed to a Move Request for a copy
4	POSTCHANGE	After all changes have been completed to a Move Request for a change
5	ADD,COP,CHG	When you add or change a member to a Move Request during an add, copy or change
6	CLO,VER	When you close or verify a Move Request
7	BACKOUT	During Back Out setup
8	SELECTION	During the move processing selection process
9		During ASN or ACK actions
A	PURGE	When you purge a member from a Move Request

Code=1

When you copy a Move Request (MAKECOPY, Code=1), CA-PanAPT calls the exit for eligible members before the Description of Move Request panel is displayed. Note that the verify flag is always initialized to V during this process.

Code=2

When you specify an action of **CHG** for a Move Request (PRECHANGE, Code=2), CA-PanAPT calls the exit for each member of the Move Request after the Description of Move Request panel is displayed and before the Member Moves panel is displayed.

Code=3

When you have completed all changes to a Move Request (POSTCOPY, Code=3) and entered END, CA-PanAPT calls the exit for all members of the Move Request after all assignments have been done and the Inventory Record updated (before the Move Request Maintenance or Browse Move Requests List panel is displayed).

Code=4

When you have completed all changes to a Move Request (POSTCHANGE, Code=4) and entered END, CA-PanAPT calls the exit for each member (before the Move Request Maintenance or Browse Move Requests List panel is displayed).

Code=5

When you add or change a member during ADD, COP, or CHG processing, CA-PanAPT calls the exit for the new member (Code=5). At the time of the call, all assignments have been attempted and the Inventory Record is updated, if possible. The verify flag is always initialized to V before the exit is called.

When you **ASN** or **ACK** during ADD, COP, or CHG processing; Code=9.

Code=6

When you close (**CLO**) or verify (**VER**) a Move Request, CA-PanAPT calls the exit for each member in the Move Request (Code=6).

Code=7

When you specify an action of **BAK** for a Move Request (Back Out, Code=7), CA-PanAPT calls the exit for each member before completing the Back Out action. The verify flag is always blank during this process.

The $MEMBEREXIST modeling keyword is set according to the return value from the member existence exit:

Y

The member passed the checks in the exit.

N

The exit was invoked, but the member did not pass the checks in the exit.

blank

The exit was not invoked.

Code=8

When APCS5310 selects a Move Request for move processing (MOVE, Code=8), CA-PanAPT calls the exit for each member during the selection process. The verify flag is always blank during this process.

The exit specified for the Test level is called with Codes 1-6 during preparation of the Move Request. It is also called with Code=8 during Move Request selection for Move to QA or Move to Prod if there is no QA level library.

The exit specified for the QA level is called:

- By Online (ISPF) processing with Codes 1-6 when you change the Move Request while it is awaiting Prod approvals
- By Back Out processing with Code=7 when preparing a Move Request for Back Out processing
- By Move Request selection with Code=8 for Move to Prod when there is a QA level library or for Back Out requests.

Code=9

When you perform the ASN or ACK action during Move Request maintenance, the exit is called with Code=9.

Code=A

When you perform the PURGE action during Move Request maintenance, the exit is called with Code=A. In this case, ACTION field passed to the exit (described in the APCC02XX and APAM02XX copy members) also contains the value MEMPURGE as opposed to MEMBER.

Parameters Passed

When the Member Existence Exit point is taken, five parameter records, described in Appendix C are passed to the exit program:

- The first parameter record supplies data required for physical access to the library and member. It is mapped by COBOL member APCC02XX and Assembler member APAM02XX.
- The second supplies inventory data for the member. It is mapped by COBOL member APCCDIB2 and Assembler member APAMDIB2.
- The third contains information about the Library Code to which this member belongs. It is mapped by COBOL member APCCLIB2.
- The fourth supplies data from the Move Request for this member. It is mapped by COBOL member APCCMMBR.
- The fifth supplies data from the Move Request for the Move Request as a whole, not member-related data. It is mapped by COBOL member APCCMDES.

Exit Point

Using these parameters, the exit point must set one of the return Codes in register 15 as shown below.

Return Code	Purge Request?	Interpretation	Verify Flag Setting
0	No	Member found (member validated by the exit)	blank
4	No	Member not found (member rejected by the exit)	V
8	No	Nothing processed	unchanged

Return Code	Purge Request?	Interpretation	Verify Flag Setting
12	No	Error encountered	V
0	Yes	Member purged or didn't exist	N/A
4	Yes	Member doesn't exist, exit has message to be reported	N/A
8	Yes	Exit doesn't support purge	N/A
12	Yes	Error encountered	N/A

Notes:

- In case of an error, the exit sets the error code and error message and returns to the caller.
- The verify flag is not used for exit Codes 7, 8, and A.
- If the return code is not 0 or 8, CA-PanAPT displays the first error message generated and contained in the field APCS02XX-MSG. CA-PanAPT ignores all other fields.
- If the return code is not 0 or 8 for Codes 1 through 6, or if the return code is not 0 for Code A, messages are written to the ISPF Log file. The Log file can be viewed online using ISPF Dialog Test, LOG facility (=7.5). For Code 7, messages are viewed online during the BAK action. For Code 8, messages are written to the APCS5310-01 report.
- If the return code is 8, verify flags remain the same and there is no error message or Log message.
- Any other return code is treated as an internal error. CA-PanAPT displays the message contained in the field APCS02XX-MSG.

Important! The member existence exit is called from many different places in the online and batch Move process in an effort to give the most current information about each member of the Move Request. Verifying existence is resource intensive; it requires allocation/deallocation as well as open and close processing for each member. If an exit verified each member each time the exit was called, much of the information would be redundant and the online response time would be degraded.

The sample exits provide a compromise between response time and currency of information. If you find that the information is not current enough and want to add the overhead of an additional verification, you can insert an entry into the verify table (APAMXTBL-TABLE or APCCXTBL-TABLE for COBOL) for each exit that allows the extra verify process.

For example, suppose you want to ensure that the verify flag is current for a member in any of the following conditions:

- The member had been previously verified (verify flag is blank)
- After Change processing (Code=4)

- Auto Retrieve is not enabled (N)

Insert the following entry into the verify TABLE of each exit:

```
DC CL3' 4N' VERIFY DONE, POST-CHANGE, NO AUTO RETRIEVE
```

These Codes are explained in the source code for each of the sample exits. Note that the exits can perform additional edits on each member based upon information in the Inventory Record and/or the point in the Move processes from which the exit is being called. You can use this information to call your security system if you desire.

Inventory Exit Interface

CA-PanAPT models can use information stored in the Inventory Record to generate customized processing instructions which vary by member. A model could, for example, use the ENVIRONMENT inventory item to determine whether to apply CICS pre-processing to a member. In this way, members for batch and CICS could be maintained under the same Library Code, but the model could generate different processing for each.

But models can only deal with pre-defined conditions. If a user entered CICSHX as the ENVIRONMENT inventory item for a member, the model might fail. That, in turn, could terminate the entire daily processing job.

The CA-PanAPT Inventory Exit can ensure that data entered for an Inventory Record meets the site's standards. You can also use it to alter information entered on the Inventory File Maintenance panel. CA-PanAPT invokes the exit when a user adds, changes, or approves an Inventory Record.

The Inventory Exit is called after user input is collected and before the data is written to the Inventory file.

When the Inventory Exit point is taken, four parameters are passed to the exit program:

- Inventory information obtained from the Inventory File panel
- The Inventory File record
- The Library Code record
- Fields from the CA-PanAPT system data area.

These parameters give the user an opportunity to approve or disapprove pending activity by comparing panel input data to existing records.

When the user exit finishes processing, data contained in the following fields is used to update the Inventory Record. *Your exit must not place bad data in these fields.*

```
APIP610-LIB-CODE APIP610-USER-TABLE-8
APIP610-LIB-SUBCODE      APIP610-USER-TABLE-16
APIP610-OWNER-ID APIP610-USER-TABLE-50
APIP610-APPROVED-FLAG    APIP610-COMPILER-OPTIONS
APIP610-DESCR    APIP610-LINKAGE-OPTIONS
APIP610-ENVIRONMENT      APIP610-DB-OPTIONS
APIP610-APPLICATION      APIP610-CICS-OPTIONS
APIP610-LANGUAGE APIP610-LINK-STREAM
APIP610-COMMENTS APIP610-LIST-OVERRIDE
APIP610-PANEL-ID APIP610-OBJECT-OVERRIDE
APIP610-DIBSCFLG APIP610-LOAD-OVERRIDE
APIP610-DIBSUFLG
```

CA-PanAPT provides a sample exit program (APCS0304) to edit Inventory Records being created, modified, or deleted.

Notes:

- Field APIP610-PANEL-ID contains either APIP610, APIP611, or APIP612. If you want to edit data on any of these panels, you must first check that APIP610-PANEL-ID currently contains the correct panel name. If it does not, then your exit might validate data that has not been entered yet.
- Field APIP610-DIBSCFLG contains the 1-byte Compile/Link Options Y/N value from panel APIP610. It allows the user to indicate if they want to display or maintain the compiler option fields on panel APIP612.
- Field APIP610-DIBSUFLG contains the 1-byte User data Y/N value from panel APIP610. It allows the user to indicate if they want to display or maintain the user fields on panel APIP611.
- CA-PanAPT provides a sample exit program (APCS0304) to edit Inventory Records being created, modified, or deleted.

Parameter Record Layouts

Appendix C contains the record layouts of the parameters passed to the inventory exit program. These records are as follows:

- The APIP610-RECORD, which is mapped by COBOL member APCCINV1.
- The INV-REC (Inventory Record), which is mapped by COBOL member APCCDIB2 and Assembler language member APAMINV.
- The Library Code Record (01 LIB-REC), which is mapped by COBOL member APCCLIB2 and Assembler member APAMLIB2.
- The ENVIRONMENT-RECORD, which is mapped by COBOL member APCCENVM and Assembler member APAMENVM.

Exit Point

Using the above parameters, the exit point must set one of the return Codes in register 15 as is shown below.

Return Code	Interpretation
0	Record passed edits.
4	Data did not pass review edits. Do not update Inventory file.
8	Invalid action, only valid actions are Add, Change, Autoadd, and Approve.
	Any other value is interpreted as an edit error.

If a non-zero return code is passed back from the exit program, CA-PanAPT displays the contents of a message passed back from the exit point in ENVIRON-MSG. CA-PanAPT re-displays the panel without performing the action.

Move Request MSL Exit Programs

The Member Selection List (MSL) Exit is used to display a list of members from a specific data set specified in the Libcode file. The exit is called from the MSL processing program, which displays the member returned from the exit.

CA-PanAPT provides three sample MSL exit programs:

- APAS0600 - Reads a CA-Panvalet directory
- APAS0610 - Reads a PDS directory
- APAS0620 - Reads a CA-Librarian directory.

You can use the source forms of these exits, found in CA-PanAPT APTEXSRC, to create your own exits. You can also create your own exits.

If you have Library Codes that contain a mixture of CA-Panvalet, CA-Librarian, and PDS libraries, you can consolidate these sample exits into one program. Since the APCCLIB2/APAMLIB2 copybook is passed to the exit, you can interrogate the field ACCMETH, (access method for the associated data set). This field tells you which access method to use. The access method is added and maintained using CA-PanAPT Libcode maintenance. Each data set in the Library Code has separate Access Method specifications.

If you do modify the sample exits, select new names for your exits and leave the originals for future comparisons.

Exit Calls

You specify, in the Library Code definition, the MSL exit program to be invoked when an MSL is requested for a data set in the Library Code. If the Library Code does not specify an MSL exit program, a message to that effect is displayed at the user's terminal, and the MSL request is not processed.

The MSL Exit is called only when requested through a field on the Member Moves panel. The APXXFUNC field in the Exit Parameter Structure contains a code that indicates which of the following functions were being performed when the exit was called. (An exit can use the value of this code to determine processing requirements.)

Exit Call Functions

INIT

The INIT call is performed once per MSL request. The INIT call passes required information to the exit and, in return, expects the exit to return data for the first member from the specified data set. The possible Return Codes from the INIT call are:

Return Code	Interpretation	Action
0	The call was successful.	Data from the first member has been returned.
4	End of File	No matching members were found that passed the criteria.
8	An Open/Allocation error occurred in the exit program.	The exit should return a diagnostic message in the area provided in the Exit Parameter Structure. This message is displayed to the user.
16	Other error (unspecified).	The exit program encountered an error. The exit should provide a diagnostic message in the area provided in the Exit Parameter Structure. The message displays for the user.

PROCESS

The PROCESS call is performed multiple times until a non-zero return code received from the exit program. The possible Return Codes from the PROCESS call are:

Return Code	Interpretation	Action
0	The call was successful.	Member data has been returned.
4	End of File/No more member data.	The last member passing the selection criteria has been returned. The MSL Processing Program, APCS1150 acknowledges this with a TERM call.
8	An error (non-fatal) occurred in the exit program.	Any members returned are displayed on the MSL. The exit should return a diagnostic message in the area provided in the Exit Parameter Structure. This message is displayed to the user with the MSL. Depending on the type of error encountered, it is possible that the member list is incomplete.
16	Fatal error	The exit program encountered an error. The MSL is not to be displayed to the user. The exit should provide a diagnostic message in the area provided in the Exit Parameter Structure. The message displays for the user. This return code assures that an incomplete MSL is not displayed.

TERM

The TERM call is made after the exit has returned a EOF (end of directory--last member has been returned). This is an acknowledgment to the exit program that the MSL request has successfully completed. The possible Return Codes from the TERM call are:

Return Code	Interpretation	Action
0	The call was successful.	Program terminated successfully.

Return Code	Interpretation	Action
8	An error occurred during program termination.	Any members returned are displayed on the MSL. The exit should return a diagnostic message in the area provided in the Exit Parameter Structure. This message is displayed to the user.
16	Fatal error.	The exit program encountered an error. The MSL is not to be displayed to the user. The exit should provide a diagnostic message in the area provided in the Exit Parameter Structure. The message is displayed for the user. This return code assures that the MSL is not displayed.

Security Exit Interface

The Security Exit Interface permits you to control the use of certain CA-PanAPT operations (see the subject Security Control in the chapter "Planning the Implemetation").

The ACTIVITY event is activated by your CA-PanAPT system administrator, but the way in which your CA-PanAPT system responds to the events depends upon how you set up the exit Pass/Fail indicator in your Security Exit Program.

If any security events are activated, the load module that the system administrator specified on the System Security Information Maintenance panel must be available to CA-PanAPT.

If the Exit Program module cannot be loaded, an abort message displays and CA-PanAPT exits. The exit's existence is checked only during the INIT event. To correct the problem, either restore the Control file to its image prior to activating the Security Exit Program for the CA-PanAPT environment or supply the appropriate Security Exit Program.

Pass/Fail Indicator

Information is passed from events to the Security Exit Program through a Parameter Address List described later in this appendix. The exit program then indicates what action CA-PanAPT should take based upon the Pass/Fail indicator.

Upon entry to the Security Exit Program, the value of the Pass/Fail indicator is set to C or F according to CA-PanAPT's built-in authorization for the activity. Based on the indicator, CA-PanAPT can take one of two possible actions:

- Process the request and continue normally.
- Fail the current request but permit another selection.

To cause a particular response, set up the Pass/Fail indicator as follows:

- Process the request and CONTINUE normally. This occurs when you set the Pass/Fail indicator to C. CA-PanAPT processes the user's request and continues normally.
- FAIL the current request, but permit another selection. This occurs when you set the Pass/Fail indicator to F. The user remains on the CA-PanAPT panel but receives a security violation error message.
- FAIL the request if an invalid response is specified in the Pass/Fail indicator and write a message to the ISPF log data set.

Information Blocks

CA-PanAPT passes information to the Security Exit Program through six information blocks:

- Environment Block
- Event Block
- Move Request Block
- Member Block
- Approval Block
- Inventory Block.

The specific CA-PanAPT activity determines which blocks are valid and what data is available.

Environment Block

Contains environment information, such as the name and release of CA-PanAPT. For details, see Appendix C. The Environment Block is always available.

Event Block

Contains:

- Information regarding the event and the exit program
- The Pass/Fail Indicator
- A list of data blocks containing valid information for this event
- A list of flags indicating when each data block is valid.

The exit program uses the Event Block to communicate what action CA-PanAPT should take. For details, see Appendix C. The Event Block is always available.

Move Request Block

Contains information specific to the entire Move Request. For details, see Appendix C.

Member Block

Contains information specific to the member being added or changed. For details, see Appendix C.

Approval Block

Contains information specific to the approvals needed and the approvals being issued for this Move Request. For details, see Appendix C.

Inventory Block

Contains information specific to the inventory data of the member being added or changed. For details, see Appendix C.

Security Events

CA-PanAPT supports three types of security events: initialization, termination, and activity. Initialization and Termination Events are automatically invoked whenever the user enters or exits CA-PanAPT once the CA-PanAPT system administrator has activated the security exit. Activity events are invoked after all the information for a specific activity has been entered by the user.

INIT

The Initialization Event (INIT) is invoked as each user enters the CA-PanAPT environment. At the INIT Event, the Move Request, Member, Approval, and Inventory Blocks do not contain any data. The only blocks containing information are the Event Block and the Environment Block.

Two Pass/Fail indicators are possible: Continue or Fail. The default is Continue. If the response is Continue, CA-PanAPT continues normally. If the response is Fail, CA-PanAPT displays an internal error message and returns to the ISPF Primary Option panel. The TERM event is not processed in this situation.

If the response is an unknown value, CA-PanAPT returns to the ISPF Primary option and displays an internal error message.

TERM

Each time a user exits the CA-PanAPT environment, the Termination Event (TERM) is invoked as a termination call for the exit program. The exit program performs any clean up procedures at this time.

At the TERM Event, the Move Request, Member, Approval, and Inventory Blocks do not contain any data. The only blocks containing information are the Event and Environment Blocks.

Pass/Fail indicators are ignored at this event because you are leaving the CA-PanAPT environment.

ACTIVITY

If the Activity Event (ACTIVITY) is selected, CA-PanAPT invokes the Exit Program after all information for a specific activity is entered by the user. The Activity event receives control after CA-PanAPT user authorization has verified the activity.

For example, this Exit can enforce emergency Move Request approval procedures. The sample Exit allows certain User IDs approval/disapproval capabilities when the move type indicates that this is an emergency Move Request.

At the ACTIVITY Event, the Move Request Block, Member Block, Approval Block, and Inventory Block contain the member name, Move Request number, move type, Move-to-QA-Only flag, move date, first run date, service request, status, number of members, approvals required for current status, approvals-to-date for current status, and approvals-being-issued for current status. The data available varies depending on the activity.

Two Pass/Fail indicators are possible: Continue or Fail. CA-PanAPT's built-in authorization determines the default. If the response is Continue, the activity continues normally. If the response is Fail, the current panel re-displays with a security violation error message supplied by the exit program. The user is not allowed to perform the requested activity, but the user can continue with other activities in CA-PanAPT.

If the response is an unknown value, CA-PanAPT returns to the ISPF Primary Option panel and displays an internal error message. Further information is in the ISPF log file.

Security Exit Linkage Conventions

Standard subroutine linkage conventions pass control to the Exit Program. Six parameter records are passed to the exit program. The first parameter record supplies data to the event block; the second supplies data to the environment block; the third supplies data to the Move Request block; the fourth supplies data to the member block; the fifth supplies data to the approval block; and the sixth supplies data to the Inventory block.

Event Block

The first parameter (see Appendix C) is mapped by COBOL member APCCSXEV and Assembler member APAMSXEV.

Environment Block

The second parameter (see Appendix C) is mapped by COBOL member APCCSXEN and Assembler member APAMSXEN.

Move Request Block

The third parameter (see Appendix C) is mapped by COBOL member APCCMDES and Assembler member APAMMDES.

Member Block

The fourth parameter (see Appendix C) is mapped by COBOL member APCCMMBR and Assembler member APAMMBR.

Approval Block

The fifth parameter (see Appendix C) is mapped by COBOL member APCCSXAP and Assembler member APAMSXAP.

Inventory Block

The sixth parameter (see Appendix C) is mapped by LE/390 COBOL member APCCLIB2 and Assembler member APAMLIB2.

Appendix C: Record Layouts

This section contains the following topics:

Member Existence Exit Parameter, 1

COBOL Member APCC02XX

The following record layout, mapped by COBOL member APCC02XX and Assembler member APAM02XX, describes the data required for physical access to the library and member.

```
****************** < Description Begin      > ********************
*                                                                 *
* Name       : APCC02XX                                           *
* Product    : PANAPT                                             *
* Type       : Cobol Copybook                                     *
*                                                                 *
* Descript.   : APCS02XX (Member Existence Exit) parameter list. *
*                                                                 *
* Notices    : This module is part of the distributed source     *
*               code for PANAPT.                                  *
*                                                                 *
*              Copyright (C) 1992 Computer Associates             *
*              International Inc. All rights reserved.            *
*                                                                 *
*              This software is proprietary information and its   *
*              use by unauthorized persons is prohibited.         *
*                                                                 *
****************** < Description End        > ********************
     SKIP1
****************** < Documentation Begin    > ********************
*                                                                 *
* Function   : To provide a common record description for the    *
*              calling paremeters for the APxS02XX series         *
*              programs, which are the Member Existence exits.    *
*                                                                 *
* Related to : APAM02XX must reflect changes made to this member.*
 *                                                                 *
* Comments   : None.                                              *
*                                                                 *
****************** < Documentation End      > ********************
     SKIP1
     SKIP1
 01  APCS02XX-PARM-AREA.
     05  APCS02XX-ACTION         PIC X(08)      VALUE SPACE.
       88  APCS02XX-ACTION-TEST-MEMBER          VALUE 'MEMBER'.  WOOLWRTH
       88  APCS02XX-ACTION-PURGE-MEMBER         VALUE 'MEMPURGE'. WOOLWRTH
     05  APCS02XX-DDN            PIC X(08)      VALUE SPACE.
     05  APCS02XX-DSN            PIC X(44)      VALUE SPACE.
     05  APCS02XX-SECURITY       PIC X(10)      VALUE SPACE.
     05  APCS02XX-MEMBER         PIC X(10)      VALUE SPACE.
     05  APCS02XX-DATA           PIC X(08)      VALUE SPACE.
                                                                020A*BAD
```

```
*    The APCS02XX-LEVEL-ID, ABBR, SNAME, and SYSTEM-POS field     020A*BAD
*    are all indicators of the CA PanAPT level for the member     020A*BAD
*    being tested. The LEVEL-ID field contains the internal       020A*BAD
*    unique name CA PanAPT uses for the level. ABBR and SNAME     020A*BAD
*    contain the names the users associate with the level, such   020A*BAD
*    as "P" and "PROD" for a Production level. SYSTEM-POS         020A*BAD
*    contains the position of the level in regards to other       020A*BAD
*    levels in the CA PanAPT system (whether they are used by     020A*BAD
*    this Move Request or not), with the starting level's         020A*BAD
*    position being 1. If you add or delete levels from your      020A*BAD
*    system, the SYSTEM-POS field will change. If you change      020A*BAD
*    the names of levels, the ABBR and SNAME field will change,   020A*BAD
*    but the LEVEL-ID field will not.                             020A*BAD
                                                                  020A*BAD
     05  APCS02XX-LEVEL-ID       PIC 9(09)      COMP.            020A*BAD
     05  APCS02XX-LEVEL-ABBR     PIC X(02).                      020A*BAD
     05  APCS02XX-LEVEL-SNAME    PIC X(04).                      020A*BAD
     05  APCS02XX-LEVEL-SYSTEM-POS PIC 9(04)    COMP.            020A*BAD
                                                                  020A*BAD
     05  APCS02XX-PROCESS-OPTIONS.
       10  APCS02XX-VERIFY-FLAG  PIC X(01)      VALUE SPACE.
       10  APCS02XX-SOURCE       PIC X(01)      VALUE SPACE.
         88  APCS02XX-FROM-MAKECOPY             VALUE '1'.
         88  APCS02XX-FROM-PRECHANGE            VALUE '2'.
         88  APCS02XX-FROM-POSTCOPY             VALUE '3'.
         88  APCS02XX-FROM-POSTCHANGE           VALUE '4'.
         88  APCS02XX-FROM-ADD                  VALUE '5'.
         88  APCS02XX-FROM-COP                  VALUE '5'.       1050162
         88  APCS02XX-FROM-CHG                  VALUE '5'.       1050162
         88  APCS02XX-FROM-CLOSE                VALUE '6'.
         88  APCS02XX-FROM-BACKOUT              VALUE '7'.
         88  APCS02XX-FROM-SELECTION            VALUE '8'.       1050162
         88  APCS02XX-FROM-ASN                  VALUE '9'.       1050162
         88  APCS02XX-FROM-ACK                  VALUE '9'.       1050162
         88  APCS02XX-FROM-PURGE                VALUE 'A'.       WOOLWRTH
       10  APCS02XX-AUTO-RETRIEVE PIC X(01)     VALUE SPACE.     030A*BAD
     05  APCS02XX-PARM           PIC X(50)      VALUE SPACE.
     05  APCS02XX-VERSION        PIC X(04)      VALUE SPACE.
                                                                  020A*BAD
*    Note. Field APCS02XX-MR-STATUS from previous versions        020A*BAD
*    of CA PanAPT has been removed. If any of your exits          020A*BAD
*    referenced this field they should be changed to reference    020A*BAD
```

```
*    the MR-STATUS or MR-CURRENT-... fields of the APCCMDES       020A*BAD
*    copy book.                                                   020A*BAD
                                                                  020A*BAD
     05  APCS02XX-MSG            PIC X(79)      VALUE SPACE.
```

Member Existence Exit Parameter, 2

COBOL Member APCCDIB2

The following record layout, mapped by COBOL member APCCDIB2, describes the inventory data for a member.

```
****************** < Description Begin      > ********************
*                                                                 *
* NAME       : APCCDIB2                                           *
* Product    : PANAPT                                             *
* Type       : Cobol Copybook                                     *
*                                                                 *
* DESCRIPT.  : PANAPT Inventory record definition for the         *
*               User Inventory Exit.                              *
*                                                                 *
* Notices    : This module is part of the distributed source      *
*               code for PANAPT.                                  *
*                                                                 *
*              Copyright (C) 1992, 1996 Computer Associates      *030A*BAD
*              International Inc. All rights reserved.            *
*                                                                 *
*              This software is proprietary information and its   *
*              use by unauthorized persons is prohibited.         *
*                                                                 *
****************** < Description End        > ********************
     SKIP2
****************** < Documentation Begin    > ********************
*                                                                 *
* Function   : To provide a common record description for the     *
*              PANAPT User Inventory Exit.                        *
*                                                                 *
* Related to : APAMDIB2 must reflect all changes to this member.  *
*                                                                 *
* Comments   : APAMDIB2 MUST Reflect ALL Changes to this Member.  *
*                                                                 *
****************** < Documentation End      > ********************
     SKIP2
* RECORD LENGTH   1152 BYTES                                       030A*BAD
 01  INV-REC.
     SKIP1
*****************************************************************
* LIBRARY CODE / MEMBER NAME FIELDS
*****************************************************************
     SKIP1
     02  INV-REC-LEN             PIC 9(05) COMP.                 030A*BAD
     02  INV-REC-TYPE            PIC X(02).                      020A*SYW
*        A value of '04' indicates an INV-REC.                   030A*BAD
     02  INV-LIBCODE.
```

```
       03  INV-LIB-CODE          PIC X(04)      VALUE SPACE.
       03  INV-LIB-SUBCODE       PIC X(03)      VALUE SPACE.
     02  INV-MEMBER-QUALIFIER.
       03  INV-MEMBER            PIC X(10)      VALUE SPACE.
       03  INV-QUALIFIER         PIC X(08)      VALUE SPACE.
     SKIP1
*****************************************************************
* DATE/TIME INFORMATION FIELDS
*****************************************************************
     SKIP1
     02  INV-UPDATE-ID           PIC X(08)      VALUE SPACE.
     02  INV-DATE.
       03  DATE-CC               PIC X(02)      VALUE SPACE.
       03  DATE-YY               PIC X(02)      VALUE SPACE.
       03  DATE-MM               PIC X(02)      VALUE SPACE.
       03  DATE-DD               PIC X(02)      VALUE SPACE.
     02  INV-TIME.
       03  TIME-HH               PIC X(02)      VALUE SPACE.
       03  TIME-MM               PIC X(02)      VALUE SPACE.
       03  TIME-SS               PIC X(02)      VALUE SPACE.
     SKIP1
*****************************************************************
* INVENTORY INFORMATION FIELDS
*****************************************************************
     SKIP1
     02  INV-APPLICATION         PIC X(08)      VALUE SPACE.
     02  INV-APPROVED-FLAG       PIC X(01)      VALUE SPACE.
     02  INV-COMMENTS            PIC X(55)      VALUE SPACE.
     02  INV-DESCR               PIC X(55)      VALUE SPACE.
     02  INV-ENVIRONMENT         PIC X(08)      VALUE SPACE.
     02  INV-LANGUAGE            PIC X(08)      VALUE SPACE.
     02  INV-LAST-MOVED-BY-MR    PIC X(06)      VALUE SPACE.
     02  INV-OWNER-ID            PIC X(08)      VALUE SPACE.
     02  FILLER                  PIC X(08)      VALUE SPACE.
     SKIP1
*****************************************************************
* ASSIGNMENT INFORMATION FIELDS
*****************************************************************
     SKIP1
     02  INV-ASSIGNED-FLAG       PIC X(01)      VALUE SPACE.
     02  INV-ASSIGNED-DATE.
       03  ASSIGNED-CC           PIC X(02)      VALUE SPACE.
```

```
       03  ASSIGNED-YY           PIC X(02)      VALUE SPACE.
       03  ASSIGNED-MM           PIC X(02)      VALUE SPACE.
       03  ASSIGNED-DD           PIC X(02)      VALUE SPACE.
     02  INV-ASSIGNED-TIME.
       03  ASSIGNED-HH           PIC X(02)      VALUE SPACE.
       03  ASSIGNED-MM           PIC X(02)      VALUE SPACE.
       03  ASSIGNED-SS           PIC X(02)      VALUE SPACE.
     02  INV-ASSIGNED-TO-MR      PIC X(06)      VALUE SPACE.
     02  INV-ASSIGNED-TO-USER    PIC X(08)      VALUE SPACE.
     02  FILLER                  PIC X(20)      VALUE SPACE.
 ****************************************************************030A*BAD
* Compile options                                                 030A*BAD
****************************************************************030A*BAD
     02  INV-COMPILE-DATA.                                       030A*BAD
       03  INV-COMPILER-OPTIONS  PIC X(60)      VALUE SPACE.      030A*BAD
       03  INV-LINKAGE-OPTIONS   PIC X(60)      VALUE SPACE.      030A*BAD
*          Data Base precompile                                  030A*BAD
       03  INV-DB-OPTIONS        PIC X(60)      VALUE SPACE.      030A*BAD
*          CICS precompile                                       030A*BAD
       03  INV-CICS-OPTIONS      PIC X(60)      VALUE SPACE.      030A*BAD
*          Name of linkage editor control statement member       030A*BAD
       03  INV-LINK-STREAM       PIC X(10)      VALUE SPACE.      030A*BAD
*          For OVERRIDE fields, Y means the LIST, OBJ, or LOAD   030A*BAD
*          must be kept. If the Library Code is no longer        030A*BAD
*          setup to keep the output, a modelling error should    030A*BAD
*          be generated by the model. N means don't keep the     030A*BAD
*          output, even if the Library Code is setup to keep it. 030A*BAD
*          Blank means save it if the Library Code is setup to   030A*BAD
*          save it, otherwise don't save it.                     030A*BAD
       03  INV-LIST-OVERRIDE     PIC X(01)      VALUE SPACE.      030A*BAD
       03  INV-OBJECT-OVERRIDE   PIC X(01)      VALUE SPACE.      030A*BAD
       03  INV-LOAD-OVERRIDE     PIC X(01)      VALUE SPACE.      030A*BAD
       03  FILLER                PIC X(20)      VALUE SPACE.      030A*BAD
****************************************************************
* USER INVENTORY TABLES
****************************************************************
     02  INV-USER-DATA.                                          012D*JLT
       03  INV-USER-TABLE-8.                                     012D*JLT
         05  INV-USER-ENTRY-01   PIC X(8)       VALUE SPACE.    012D*JLT
         05  INV-USER-ENTRY-02   PIC X(8)       VALUE SPACE.    012D*JLT
         05  INV-USER-ENTRY-03   PIC X(8)       VALUE SPACE.    012D*JLT
         05  INV-USER-ENTRY-04   PIC X(8)       VALUE SPACE.    012D*JLT
```

```
  05  INV-USER-ENTRY-05   PIC X(8)       VALUE SPACE.     012D*JLT
03  INV-WORK-TABLE-8  REDEFINES INV-USER-TABLE-8.         012D*JLT
  05  INV-WORK-ENTRY-8    PIC X(8) OCCURS 5 TIMES.       012D*JLT
03  INV-USER-TABLE-16.                                   012D*JLT
  05  INV-USER-ENTRY-06   PIC X(16)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-07   PIC X(16)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-08   PIC X(16)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-09   PIC X(16)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-10   PIC X(16)      VALUE SPACE.     012D*JLT
03  INV-WORK-TABLE-16 REDEFINES INV-USER-TABLE-16.        012D*JLT
  05  INV-WORK-ENTRY-16   PIC X(16) OCCURS 5 TIMES.      012D*JLT
03  INV-USER-TABLE-50.                                   012D*JLT
  05  INV-USER-ENTRY-11   PIC X(50)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-12   PIC X(50)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-13   PIC X(50)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-14   PIC X(50)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-15   PIC X(50)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-16   PIC X(50)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-17   PIC X(50)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-18   PIC X(50)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-19   PIC X(50)      VALUE SPACE.     012D*JLT
  05  INV-USER-ENTRY-20   PIC X(50)      VALUE SPACE.     012D*JLT
03  INV-WORK-TABLE-50 REDEFINES INV-USER-TABLE-50.        012D*JLT
  05  INV-WORK-ENTRY-50   PIC X(50) OCCURS 10 TIMES.     012D*JLT
```

Parameter Mapped by APCCINV1

The following record layout, mapped by COBOL member APCCINV1, describes the data of the parameter passed to the inventory exit program.

```
****************** < Description Begin      > ********************
*                                                                  *
* Name       : APCCINV1                                            *
* Product    : PANAPT                                              *
* Type       : Cobol Copybook                                      *
*                                                                  *
* Descript. : APCCINV1 Inventory exit parameter list.              *
*                                                                  *
*                                                                  *
* Notices    : This module is part of the non-distributed source *
*              code for PANAPT.                                    *
*                                                                  *
*              Copyright (C) 1992, 1996 Computer Associates      *030A*BAD
*              International Inc. All rights reserved.             *
*                                                                  *
*              This software is proprietary information and its    *
*              use by unauthorized persons is prohibited.          *
*                                                                  *
****************** < Description End        > ********************
     SKIP1
****************** < Documentation Begin    > ********************
*                                                                  *
* Function   : To provide a common record description for the     *
*              data obtained from panel APIP610 that is passed     *
*              to the Inventory exit.                              *
*                                                                  *
* Related to : Panel APIP610 - any changes to panel APIP610 must *030A*BAD
*              be reflected in this member. Changes to this     *030A*BAD
*              member must also be made in APAMINV.             *030A*BAD
*                                                                  *
* Comments   : None.                                               *
*                                                                  *
****************** < Documentation End      > ********************
     SKIP1
     SKIP1
 01  APIP610-RECORD.
 *    APIP610 fields...                                            030A*BAD
     02  APIP610-LIB-CODE        PIC X(04)      VALUE SPACE.
     02  APIP610-LIB-SUBCODE     PIC X(03)      VALUE SPACE.
     02  APIP610-QUALIFIER       PIC X(08)      VALUE SPACE.
     02  APIP610-MEMBER          PIC X(10)      VALUE SPACE.
     02  APIP610-OWNER-ID        PIC X(08)      VALUE SPACE.
```

```
     02  APIP610-APPROVED-FLAG   PIC X(01)      VALUE SPACE.
     02  APIP610-LAST-MOVED-BY-MR PIC X(06)     VALUE SPACE.
     02  APIP610-DESCR           PIC X(55)      VALUE SPACE.
     02  APIP610-ENVIRONMENT     PIC X(08)      VALUE SPACE.
     02  APIP610-APPLICATION     PIC X(08)      VALUE SPACE.
     02  APIP610-LANGUAGE        PIC X(08)      VALUE SPACE.
     02  APIP610-COMMENTS        PIC X(55)      VALUE SPACE.
     02  APIP610-PANEL-ID        PIC X(07)      VALUE SPACE.     030A*BAD
     02  APIP610-DIBSCFLG        PIC X(01)      VALUE SPACE.     030A*BAD
     02  APIP610-DIBSUFLG        PIC X(01)      VALUE SPACE.     030A*BAD
     02  APIP610-ASSIGN-FLAG     PIC X(01)      VALUE SPACE.
     02  APIP610-ASSIGNED-TO-USER PIC X(08)     VALUE SPACE.
     02  APIP610-ASSIGNED-DATE.                                  013A*JMM
       03  APIP610-ASSIGNED-DATE-CC PIC X(02)   VALUE SPACE.     013A*JMM
       03  APIP610-ASSIGNED-DATE-YY PIC X(02)   VALUE SPACE.     013A*JMM
       03  APIP610-ASSIGNED-DATE-MM PIC X(02)   VALUE SPACE.     013A*JMM
       03  APIP610-ASSIGNED-DATE-DD PIC X(02)   VALUE SPACE.     013A*JMM
     02  APIP610-ASSIGNED-TIME.                                  013A*JMM
       03  APIP610-ASSIGNED-TIME-HH PIC X(02)   VALUE SPACE.     013A*JMM
       03  APIP610-ASSIGNED-TIME-MM PIC X(02)   VALUE SPACE.     013A*JMM
       03  APIP610-ASSIGNED-TIME-SS PIC X(02)   VALUE SPACE.     013A*JMM
     02  APIP610-ASSIGNED-TO-MR  PIC X(06)      VALUE SPACE.
                                                                 030A*BAD
*    APIP611 fields, user data...                                030A*BAD
     02  APIP610-USER-DATA.                                      012D*JLT
       03  APIP610-USER-TABLE-8.                                 012D*JLT
         05  APIP610-USER-ENTRY-01 PIC X(8)     VALUE SPACE.     012D*JLT
         05  APIP610-USER-ENTRY-02 PIC X(8)     VALUE SPACE.     012D*JLT
         05  APIP610-USER-ENTRY-03 PIC X(8)     VALUE SPACE.     012D*JLT
         05  APIP610-USER-ENTRY-04 PIC X(8)     VALUE SPACE.     012D*JLT
         05  APIP610-USER-ENTRY-05 PIC X(8)     VALUE SPACE.     012D*JLT
       03  APIP610-WORK-TABLE-8 REDEFINES APIP610-USER-TABLE-8.  012D*JLT
         05  APIP610-WORK-ENTRY-8 PIC X(08) OCCURS 5 TIMES.      012D*JLT
       03  APIP610-USER-TABLE-16.                                012D*JLT
         05  APIP610-USER-ENTRY-06 PIC X(16)    VALUE SPACE.     012D*JLT
         05  APIP610-USER-ENTRY-07 PIC X(16)    VALUE SPACE.     012D*JLT
         05  APIP610-USER-ENTRY-08 PIC X(16)    VALUE SPACE.     012D*JLT
         05  APIP610-USER-ENTRY-09 PIC X(16)    VALUE SPACE.     012D*JLT
         05  APIP610-USER-ENTRY-10 PIC X(16)    VALUE SPACE.     012D*JLT
       03  APIP610-WORK-TABLE-16 REDEFINES APIP610-USER-TABLE-16. 012D*JLT
         05  APIP610-WORK-ENTRY-16 PIC X(16) OCCURS 5 TIMES.     012D*JLT
       03  APIP610-USER-TABLE-50.                                012D*JLT
```

```
        05  APIP610-USER-ENTRY-11 PIC X(50)    VALUE SPACE.     012D*JLT
        05  APIP610-USER-ENTRY-12 PIC X(50)    VALUE SPACE.     012D*JLT
        05  APIP610-USER-ENTRY-13 PIC X(50)    VALUE SPACE.     012D*JLT
        05  APIP610-USER-ENTRY-14 PIC X(50)    VALUE SPACE.     012D*JLT
        05  APIP610-USER-ENTRY-15 PIC X(50)    VALUE SPACE.     012D*JLT
        05  APIP610-USER-ENTRY-16 PIC X(50)    VALUE SPACE.     012D*JLT
        05  APIP610-USER-ENTRY-17 PIC X(50)    VALUE SPACE.     012D*JLT
        05  APIP610-USER-ENTRY-18 PIC X(50)    VALUE SPACE.     012D*JLT
        05  APIP610-USER-ENTRY-19 PIC X(50)    VALUE SPACE.     012D*JLT
        05  APIP610-USER-ENTRY-20 PIC X(50)    VALUE SPACE.     012D*JLT
      03  APIP610-WORK-TABLE-50 REDEFINES APIP610-USER-TABLE-50. 012D*JLT
        05  APIP610-WORK-ENTRY-50 PIC X(50) OCCURS 10 TIMES.    012D*JLT
                                                                 030A*BAD
*    APIP612 fields, Compile options...                         030A*BAD
     02  APIP610-COMPILE-FIELD.                                 030A*BAD
       03  APIP610-COMPILER-OPTIONS PIC X(60)   VALUE SPACE.    030A*BAD
       03  APIP610-LINKAGE-OPTIONS  PIC X(60)   VALUE SPACE.    030A*BAD
*          Data Base precompile                                  030A*BAD
       03  APIP610-DB-OPTIONS       PIC X(60)   VALUE SPACE.    030A*BAD
*          CICS precompile                                       030A*BAD
       03  APIP610-CICS-OPTIONS     PIC X(60)   VALUE SPACE.    030A*BAD
*          Name of linkage editor control statement member       030A*BAD
       03  APIP610-LINK-STREAM      PIC X(10)   VALUE SPACE.    030A*BAD
*          For OVERRIDE fields, Y means the LIST, OBJ, or LOAD   030A*BAD
*          must be kept. If the Library Code is no longer        030A*BAD
*          setup to keep the output, a modelling error should    030A*BAD
*          be generated by the model. N means don't keep the     030A*BAD
*          output, even if the Library Code is setup to keep it. 030A*BAD
*          Blank means save it if the Library Code is setup to   030A*BAD
*          save it, otherwise don't save it.                     030A*BAD
       03  APIP610-LIST-OVERRIDE    PIC X(01)   VALUE SPACE.    030A*BAD
       03  APIP610-OBJECT-OVERRIDE  PIC X(01)   VALUE SPACE.    030A*BAD
       03  APIP610-LOAD-OVERRIDE    PIC X(01)   VALUE SPACE.    030A*BAD
     02  APIP610-LIBCODE-SUPPORT.                               030A*BAD
       03  APIP610-LIST-SUPPORT     PIC X(01)   VALUE SPACE.    030A*BAD
       03  APIP610-OBJECT-SUPPORT   PIC X(01)   VALUE SPACE.    030A*BAD
       03  APIP610-LOAD-SUPPORT     PIC X(01)   VALUE SPACE.    030A*BAD
```

Parameter Mapped by APCCLIB2

The following record layout, mapped by COBOL member APCCLIB2, describes the entire Library Code record passed to the inventory exit program.

```
****************** < Description Begin      > ********************
*                                                                 *
* Name       : APCCLIB2                                           *
* Product    : PANAPT                                             *
* Type       : Cobol Copybook                                     *
*                                                                 *
* Descript. : PANAPT Library Code File record definition used     *
*               by the User Inventory Exit.                       *
*                                                                 *
* Notices    : This module is part of the distributed source      *
*               code for CA PanAPT.                              *030A*BAD
*                                                                 *
*               Copyright (C) 1992, 1997 Computer Associates      *030A*BAD
*               International Inc. All rights reserved.           *
*                                                                 *
*               This software is proprietary information and its  *
*               use by unauthorized persons is prohibited.        *
*                                                                 *
****************** < Description End        > ********************
                                                                    020A*BAD
****************** < Documentation Begin    > ********************
*                                                                 *
* Function   : To provide a common record description for the     *
*               PANAPT Library Code File passed the User          *
*               Inventory Exit(this is a rearranged copy of the   *
*               actual Library Code File record layout).          *
*                                                                 *
* Related To : PANAPT Library Code File record. Any change to  . *
*               the Library Code File record layout should be     *
*               reflected in this rearranged record layout.       *
*               This member should match APAMLIB2 byte for byte   *
*               and field for field.                              *
*                                                                 *
* Comments   : None.                                              *
*                                                                 *
****************** < Documentation End      > ********************
                                                                    020A*BAD
                                                                    020A*BAD
 01  LIB-REC.
                                                                    020A*BAD
     02  LIB-REC-LEN             PIC 9(05) COMP.                   030A*BAD
     02  LIB-REC-TYPE            PIC X(02)      VALUE '03'.        020A*BAD
```

```
     02  LIB-CODE-SUBCODE.
       03  LIB-CODE              PIC X(04)      VALUE SPACE.
       03  LIB-SUBCODE           PIC X(03)      VALUE SPACE.
                                                                020A*BAD
     02  LIB-UPD-ID              PIC X(08)      VALUE SPACE.
     02  LIB-DATE.
       03  DATE-CC               PIC X(02)      VALUE SPACE.      013A*BTK
       03  DATE-YY               PIC X(02)      VALUE SPACE.
       03  DATE-MM               PIC X(02)      VALUE SPACE.
       03  DATE-DD               PIC X(02)      VALUE SPACE.
     02  LIB-TIME.
       03  TIME-HH               PIC X(02)      VALUE SPACE.
       03  TIME-MM               PIC X(02)      VALUE SPACE.
       03  TIME-SS               PIC X(02)      VALUE SPACE.
                                                                020A*BAD
*****************************************************************
* GENERAL INFORMATION.                                         *
*****************************************************************
                                                                020A*BAD
     02  LIB-DESCR               PIC X(55)      VALUE SPACE.      012D*JLT
                                                                020A*BAD
*****************************************************************020A*BAD
* TYPE OF MEMBER PROCESSED BY THIS LIBRARY CODES. TYPICAL TYPES   030A*BAD
* ARE SOURCE, OBJECT, LOAD...                                    030A*BAD
*****************************************************************020A*BAD
                                                                020A*BAD
     02  LIB-TYPE                PIC X(08).                     030A*BAD
                                                                020A*BAD
*****************************************************************
* EDIT CRITERIA.                                               *
*****************************************************************
                                                                020A*BAD
     02  LIB-MEM-LEN-MIN         PIC 9(02)      VALUE ZERO.
     02  LIB-MEM-LEN-MAX         PIC 9(02)      VALUE ZERO.
                                                                020A*BAD
     02  LIB-FROM-DATA-LEN-MIN   PIC 9(02)      VALUE ZERO.
     02  LIB-FROM-DATA-LEN-MAX   PIC 9(02)      VALUE ZERO.
     02  LIB-TO-DATA-LEN-MIN     PIC 9(02)      VALUE ZERO.
     02  LIB-TO-DATA-LEN-MAX     PIC 9(02)      VALUE ZERO.
                                                                020A*BAD
     02  LIB-MEM-EQ              PIC X(01)      VALUE SPACE.
       88  LIB-MEM-EQ-MAY                       VALUE '1'.
```

```
       88  LIB-MEM-EQ-MUST                     VALUE '2'.
       88  LIB-MEM-EQ-MAY-NOT                  VALUE '3'.
                                                                      020A*BAD
****************************************************************
* PROCESSING CONTROL.                                           *
****************************************************************
                                                                      020A*BAD
     02  LIB-MODEL-BASE          PIC X(04).                          020A*BAD
     02  LIB-LEADING-MODEL-STMTS.                                    030A*BAD
       03  LIB-LEADING-MODEL-X OCCURS 12 TIMES.                      030A*BAD
         04  LIB-LEADING-CONTROL PIC X(75).                         030A*BAD
     02  LIB-TRAILING-MODEL-STMTS.                                   030A*BAD
       03  LIB-TRAILING-MODEL-X OCCURS 12 TIMES.                     030A*BAD
         04  LIB-TRAILING-CONTROL PIC X(75).                        030A*BAD
     02  LIB-MODEL-CONTROL-STMTS.
       03  LIB-MODEL-CONTROL-X OCCURS 12 TIMES.                      013A*BTK
         04  LIB-MODEL-CONTROL   PIC X(75).
                                                                      020A*BAD
****************************************************************
*  INVENTORY OPTIONS.                                           *
****************************************************************
                                                                      020A*BAD
     02  LIB-INV-AUTO-APPROVE    PIC X(01)      VALUE SPACE.
     02  LIB-INV-AUTO-CREATE     PIC X(01)      VALUE SPACE.
     02  LIB-INV-EXIT-PGM        PIC X(08)      VALUE SPACE.
     02  LIB-INV-EXIT-PARM       PIC X(50)      VALUE SPACE.
     02  LIB-INV-ENABLED         PIC X(01)      VALUE SPACE.
     02  LIB-INV-QUALIFIER       PIC X(08)      VALUE SPACE.
     02  LIB-INV-REQ-APPROVE     PIC X(01)      VALUE SPACE.
                                                                      020A*BAD
****************************************************************
* INV ASSIGNMENT OPTIONS.                                       *
****************************************************************
                                                                      020A*BAD
     02  LIB-INV-AS-ENABLED      PIC X(01)      VALUE SPACE.
     02  LIB-INV-AUTO-ASSIGNMENT PIC X(01)      VALUE SPACE.
     02  LIB-INV-AUTO-RELEASE    PIC X(01)      VALUE SPACE.
                                                                      020A*BAD
****************************************************************
* INV DEFAULT INVENTORY FIELDS.                                 *
****************************************************************
                                                                      020A*BAD
```

```
02  LIB-INV-APPLICATION     PIC X(08)      VALUE SPACE.
02  LIB-INV-COMMENTS        PIC X(55)      VALUE SPACE.
02  LIB-INV-ENVIRONMENT     PIC X(08)      VALUE SPACE.
02  LIB-INV-DESCR           PIC X(55)      VALUE SPACE.
02  LIB-INV-LANGUAGE        PIC X(08)      VALUE SPACE.
02  LIB-INV-OWNER-ID        PIC X(08)      VALUE SPACE.
                                                        030A*BAD
02  LIB-INV-COMPILE-DATA.                              030A*BAD
  03  LIB-INV-COMP-OPTIONS  PIC X(60)      VALUE SPACE.  030A*BAD
  03  LIB-INV-LINK-OPTIONS  PIC X(60)      VALUE SPACE.  030A*BAD
  03  LIB-INV-DB-OPTIONS    PIC X(60)      VALUE SPACE.  030A*BAD
  03  LIB-INV-CICS-OPTIONS  PIC X(60)      VALUE SPACE.  030A*BAD
  03  LIB-INV-LINK-STREAM   PIC X(10)      VALUE SPACE.  030A*BAD
  03  LIB-INV-LIST-OVERRIDE PIC X(01)      VALUE SPACE.  030A*BAD
  03  LIB-INV-OBJ-OVERRIDE  PIC X(01)      VALUE SPACE.  030A*BAD
  03  LIB-INV-LOAD-OVERRIDE PIC X(01)      VALUE SPACE.  030A*BAD
                                                        030A*BAD
02  LIB-USER-DATA.                                    012D*JLT
  03  LIB-USER-TABLE-8.                               012D*JLT
    05  LIB-USER-ENTRY-01   PIC X(8)       VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-02   PIC X(8)       VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-03   PIC X(8)       VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-04   PIC X(8)       VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-05   PIC X(8)       VALUE SPACE.  012D*JLT
  03  LIB-USER-TABLE-16.                              012D*JLT
    05  LIB-USER-ENTRY-06   PIC X(16)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-07   PIC X(16)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-08   PIC X(16)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-09   PIC X(16)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-10   PIC X(16)      VALUE SPACE.  012D*JLT
  03  LIB-USER-TABLE-50.                              012D*JLT
    05  LIB-USER-ENTRY-11   PIC X(50)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-12   PIC X(50)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-13   PIC X(50)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-14   PIC X(50)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-15   PIC X(50)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-16   PIC X(50)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-17   PIC X(50)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-18   PIC X(50)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-19   PIC X(50)      VALUE SPACE.  012D*JLT
    05  LIB-USER-ENTRY-20   PIC X(50)      VALUE SPACE.  012D*JLT
                                                         030A*BAD
```

```
*****************************************************************030A*BAD
* DEVELOPMENT OPTIONS.                                          *030A*BAD
*****************************************************************030A*BAD
                                                                 030A*BAD
     02  LIB-DEV-ENABLED         PIC X(01)      VALUE SPACE.     030A*BAD
     02  LIB-DEV-CHECKOUT-ENABLED PIC X(01)     VALUE SPACE.     030A*BAD
     02  LIB-DEV-WORK-ENABLED    PIC X(01)      VALUE SPACE.     030A*BAD
     02  LIB-DEV-LOOKUP-ACCMETH  PIC X(02)      VALUE SPACE.     030A*BAD
     02  LIB-DEV-PROCESS-COMPILE PIC X(01)      VALUE SPACE.     030A*BAD
     02  LIB-DEV-PROCESS-LINK    PIC X(01)      VALUE SPACE.     030A*BAD
     02  LIB-DEV-PROCESS-COMPLNK PIC X(01)      VALUE SPACE.     030A*BAD
     02  LIB-CHECKOUT-MODEL-STMTS.                               030A*BAD
       03  LIB-CHECKOUT-MODEL-X OCCURS 12 TIMES.                 030A*BAD
         04  LIB-CHECKOUT-CONTROL PIC X(75).                     030A*BAD
     02  LIB-CAN-CKOT-MODEL-STMTS.                               030A*BAD
       03  LIB-CAN-CKOT-MODEL-X OCCURS 12 TIMES.                 030A*BAD
         04  LIB-CAN-CKOT-CONTROL PIC X(75).                     030A*BAD
     02  LIB-CHECKIN-MODEL-STMTS.                                030A*BAD
       03  LIB-CHECKIN-MODEL-X OCCURS 12 TIMES.                  030A*BAD
         04  LIB-CHECKIN-CONTROL PIC X(75).                      030A*BAD
     02  LIB-COMPILE-MODEL-STMTS.                                030A*BAD
       03  LIB-COMPILE-MODEL-X OCCURS 12 TIMES.                  030A*BAD
         04  LIB-COMPILE-CONTROL PIC X(75).                      030A*BAD
                                                                 030A*BAD
*****************************************************************030A*BAD
* RELATED LIBRARY CODES                                         *030A*BAD
*****************************************************************030A*BAD
                                                                 030A*BAD
     02  LIB-RELATED-LIBCS.                                      030A*BAD
       03  LIB-RELATED-COMP-INCLUDES.                            030A*BAD
         04  LIB-RELATED-COMP-INCLUDE  PIC X(7)                  030A*BAD
               occurs 4 times indexed by LIB-COMP-INC-IX.        030A*BAD
       03  LIB-RELATED-LKED-INCLUDES.                            030A*BAD
         04  LIB-RELATED-LKED-INCLUDE  PIC X(7)                  030A*BAD
               occurs 4 times indexed by LIB-LKED-INC-IX.        030A*BAD
       03  LIB-RELATED-PRECMP-SYSLIBS.                           030A*BAD
         04  LIB-RELATED-PRECMP-SYSLIB PIC X(7)                  030A*BAD
               occurs 4 times indexed by LIB-PRECMP-SYSLIB-IX.   030A*BAD
       03  LIB-RELATED-COMP-SYSLIBS.                             030A*BAD
         04  LIB-RELATED-COMP-SYSLIB   PIC X(7)                  030A*BAD
               occurs 4 times indexed by LIB-COMP-SYSLIB-IX.     030A*BAD
       03  LIB-RELATED-LKED-SYSLIBS.                             030A*BAD
```

```
         04  LIB-RELATED-LKED-SYSLIB   PIC X(7)                   030A*BAD
               occurs 4 times indexed by LIB-LKED-SYSLIB-IX.      030A*BAD
       03  LIB-RELATED-LKED-SYSLIN     PIC X(7).                  030A*BAD
       03  LIB-RELATED-LISTING         PIC X(7).                  030A*BAD
       03  LIB-RELATED-OBJECT          PIC X(7).                  030A*BAD
       03  LIB-RELATED-LOAD            PIC X(7).                  030A*BAD
       03  LIB-RELATED-SOURCE-OUT      PIC X(7).                  030A*BAD
       03  LIB-RELATED-OTHER-OUT       PIC X(7).                  030A*BAD
                                                                  020A*BAD
*****************************************************************
* RETRIEVE OPTIONS.                                            *030A*BAD
*****************************************************************
                                                                  020A*BAD
     02  LIB-AUTO-RETRIEVE       PIC X(01)      VALUE SPACE.     030A*BAD
     02  LIB-RETRIEVE-ENABLED    PIC X(01)      VALUE SPACE.     030A*BAD
     02  LIB-RETRIEVE-MODEL-STMTS.                               030A*BAD
       03  LIB-RETRIEVE-MODEL-X OCCURS 12 TIMES.                 030A*BAD
         04  LIB-RETRIEVE-CONTROL PIC X(75).                     030A*BAD
                                                                  020A*BAD
*****************************************************************020A*BAD
* LEVEL SPECIFIC DATA.                                         *020A*BAD
*****************************************************************020A*BAD
                                                                  020A*BAD
     02  LIB-LEVEL-COUNT         PIC 9(02) COMP.                 020A*BAD
     02  LIB-LEVEL-DATA                                          020A*BAD
                 occurs 1 to 16 times                             020A*BAD
                 depending on LIB-LEVEL-COUNT                     020A*BAD
                 indexed by LIB-LEVEL-INDEX.                      020A*BAD
*                                                                 020A*BAD
* Note. The first entry is always for the test phase. In the      020A*BAD
*        test phase many of the fields such as approval           020A*BAD
*        requirements, backup libraries, and backout libraries    020A*BAD
*        do not apply. The LIB-LEVEL-TAG for the Test phase is    020A*BAD
*        always 1. A LIB-LEVEL-TAG value of zero marks the end    020A*BAD
*        of the table. LIB-LEVEL-TAG contains the internal        020A*BAD
*        unique ID CA PanAPT uses for a level. LIB-LEVEL-SNAME    020A*BAD
*        contains the name associate with the level, such as      020A*BAD
*        "PROD" for a Production level. LIB-LEVEL-SYSTEM-POS      020A*BAD
*        contains the position of the level in regards to other   020A*BAD
*        levels in the CA PanAPT system (whether they are used    020A*BAD
*        by this library code or not), with the starting level's  020A*BAD
*        position being 1. If you add or delete levels from       020A*BAD
```

```
*        your system, the LIB-LEVEL-SYSTEM-POS field will         020A*BAD
*        change. If you change the names of levels, the           020A*BAD
*        LIB-LEVEL-SNAME field will change, but the               020A*BAD
*        LIB-LEVEL-TAG field will not.                            020A*BAD
*                                                                 020A*BAD
                                                                  020A*BAD
       05  LIB-LEVEL-TAG         PIC 9(09)      COMP.             020A*BAD
       05  LIB-LEVEL-SNAME       PIC X(04).                       020A*BAD
       05  LIB-LEVEL-SYSTEM-POS  PIC 9(04)      COMP.             020A*BAD
*                                                                 020A*BAD
* If a phase is inactive, APT treats it as though it didn't       020A*BAD
* exist. This doesn't serve a lot of purpose, but exists for      020A*BAD
* compatibility with prior release of PANAPT, where you could     020A*BAD
* define a QA level but disable it by setting the "TEST -> QA"    020A*BAD
* flag to N in library code maintenance.                          020A*BAD
*                                                                 020A*BAD
       05  LIB-LEVEL-ACTIVE-STATUS       PIC X(1).               020A*BAD
         88  LIB-LEVEL-IS-ACTIVE                VALUE 'A'.       020A*BAD
         88  LIB-LEVEL-IS-INACTIVE              VALUE 'I'.       020A*BAD
       05  LIB-APPVER-AREA.                                       020A*BAD
         10  LIB-APPROVALS-AREA.                                  020A*BAD
           15  LIB-APPROVAL              PIC X(01)               020A*BAD
               OCCURS 20 TIMES                                    020A*BAD
               INDEXED BY LIB-APPROVALS-INDEX.                    020A*BAD
         10  LIB-VERIFICATIONS-AREA.                              020A*BAD
           15  LIB-VERIFICATION          PIC X(01)               020A*BAD
               OCCURS 20 TIMES                                    020A*BAD
               INDEXED BY LIB-VERIFICATIONS-INDEX.                020A*BAD
         10  LIB-BKOTAPP-AREA.                                    020A*BAD
           15  LIB-BKOTAPP               PIC X(01)               020A*BAD
               OCCURS 20 TIMES                                    020A*BAD
               INDEXED BY LIB-BKOTAPP-INDEX.                      020A*BAD
                                                                  020A*BAD
*                                                                 020A*BAD
* Member existence exit                                          020A*BAD
*                                                                 020A*BAD
       05  LIB-EXISTENCE-EXIT-PGM        PIC X(08).              020A*BAD
       05  LIB-EXISTENCE-EXIT-PARM       PIC X(50).              020A*BAD
*                                                                 020A*BAD
* Member selection list exit                                     020A*BAD
*                                                                 020A*BAD
       05  LIB-MSL-EXIT-PGM              PIC X(08).              020A*BAD
```

```
       05  LIB-MSL-EXIT-PARM            PIC X(50).              020A*BAD
*                                                               030A*BAD
* Member browse exit                                           030A*BAD
*                                                               030A*BAD
       05  LIB-BROWSE-EXIT-PGM          PIC X(08).              030A*BAD
       05  LIB-BROWSE-EXIT-PARM         PIC X(50).              030A*BAD
*                                                               020A*BAD
* What to do when 'MOVING' to this level.                      020A*BAD
*                                                               020A*BAD
       05  LIB-MOVE-CONTROL             PIC X(01).              020A*BAD
           88  LIB-MOVE-COPY                   VALUE 'C'.       020A*BAD
           88  LIB-MOVE-DELETE                 VALUE 'D'.       020A*BAD
           88  LIB-MOVE-MOVE                   VALUE 'M'.       020A*BAD
           88  LIB-MOVE-NONE                   VALUE 'N'.       020A*BAD
*                                                               020A*BAD
* Backup / Backout enabled?                                    020A*BAD
*                                                               020A*BAD
       05  LIB-BKUP-CONTROL             PIC X(01).              020A*BAD
           88  LIB-BKUP-ENABLED                VALUE 'Y'.       020A*BAD
       05  LIB-BKOT-CONTROL             PIC X(01).              020A*BAD
           88  LIB-BKOT-ENABLED                VALUE 'Y'.       020A*BAD
                                                               020A*BAD
       05  LIB-DDNAME            PIC X(08).                   020A*BAD
       05  LIB-DSN               PIC X(44).                   020A*BAD
       05  LIB-ACCMETH           PIC X(02).                   020A*BAD
       05  LIB-SECURITY          PIC X(10).                   020A*BAD
                                                               020A*BAD
       05  LIB-BACKUP-DDNAME     PIC X(08).                   020A*BAD
       05  LIB-BACKUP-DSN        PIC X(44).                   020A*BAD
       05  LIB-BACKUP-ACCMETH    PIC X(02).                   020A*BAD
       05  LIB-BACKUP-SECURITY   PIC X(10).                   020A*BAD
                                                               020A*BAD
       05  LIB-BACKOUT-DDNAME    PIC X(08).                   020A*BAD
       05  LIB-BACKOUT-DSN       PIC X(44).                   020A*BAD
       05  LIB-BACKOUT-ACCMETH   PIC X(02).                   020A*BAD
       05  LIB-BACKOUT-SECURITY  PIC X(10).                   020A*BAD
```

Parameter Mapped by APCCENVM

The following record layout, mapped by COBOL member APCCENVM, describes the field data from the CA-PanAPT system data area passed to the inventory exit program.

```
01  ENVIRONMENT-RECORD.
    02  ENVIRONMENT-ACTION      PIC X(08)      VALUE SPACE.
    02  ENVIRONMENT-MSG         PIC X(79)      VALUE SPACE.
    02  ENVIRONMENT-SYSTEM-ID   PIC X(04)      VALUE SPACE.
    02  ENVIRONMENT-VERSION     PIC X(04)      VALUE SPACE.
 *       ENVIRONMENT-VERSION is in the format VV.R where VV is the
*       version and R is the release, as in 02.0
```

Parameter Mapped by APCCSXEV

The following record layout, mapped by COBOL member APCCSXEV, describes the event data block passed to the security exit program.

```
01  APCCSXEV-EVENT-BLOCK.
    05  APCCSXEV-EVENT          PIC X(08).
    05  APCCSXEV-EXIT-TYPE      PIC X(08).
    05  APCCSXEV-INDICATOR      PIC X(01).
    05  FILLER                  PIC X(01).
    05  APCCSXEV-BLOCKS         PIC X(01)
        OCCURS 10 TIMES
        INDEXED BY APCCSXEV-BLOCKS-INDEX.
    05  APCCSXEV-USER-DATA      PIC X(04).
    05  FILLER                  PIC X(10).
```

Parameter Mapped by APCCSXEN

The following record layout, mapped by COBOL member APCCSXEN, describes the environment data block passed to the security exit program.

```
01  APCCSXEN-ENVIRONMENT-BLOCK.
    05  APCCSXEN-PROD-ID        PIC X(10)      VALUE SPACE.
    05  APCCSXEN-PROD-OPT       PIC X(10)      VALUE SPACE.
    05  APCCSXEN-PROD-VER       PIC X(04)      VALUE SPACE.
 *       APCCSXEN-PROD-VER is in the format VV.R where VV is the
*       version and R is the release, as in 02.0
    05  APCCSXEN-USER-ID        PIC X(08)      VALUE SPACE.
    05  APCCSXEN-ACTIVITY       PIC X(17)      VALUE SPACE.
    05  FILLER                  PIC X(19)      VALUE SPACE.
```

Parameter Mapped by APCCSXAP

The following record layout, mapped by COBOL member APCCSXAP, describes the approval data block passed to the security exit program.

```
01  APCCSXAP-APPROVAL-BLOCK.
    05  APCCSXAP-APPROVAL-STATUS PIC X(03)     VALUE SPACE.
      88  APCCSXAP-APPROVED-PROD               VALUE 'APP'.
      88  APCCSXAP-APPROVED-QA                 VALUE 'APQ'.
      88  APCCSXAP-APPROVED-BKOT               VALUE 'APB'.
    05  APCCSXAP-APPROVAL-NUMBER PIC 9(02)     VALUE ZERO.
    05  APCCSXAP-APPROVAL-ISSUED PIC X(01)     VALUE SPACE.
      88  APCCSXAP-APPROVED                    VALUE 'Y'.
      88  APCCSXAP-DISAPPROVED                 VALUE 'D'.
      88  APCCSXAP-UNAPPROVED                  VALUE 'N'.
    05  FILLER                  PIC X(14)      VALUE SPACE.
```

Move Request Record Mapped by APCCMDES

The following record layout, mapped by COBOL member APCCMDES, describes the Move Request Description record produced by the Create Sequential File of the Move Requests from the History File Job (APJJ5955). This record is input to the Batch Add Move Request Job (APJJ5960) and passed to the security exit program.

```
****************** < Description Begin      > ********************
*                                                                 *
* Name       : APCCMDES                                           *
* Product    : PANAPT                                             *
* Type       : Cobol Copybook                                     *
*                                                                 *
* Descript. : Move Request Description Record layout used by      *
*             Batch Add Move Request, Purge Move Request and      *
*             Security Exits.                                     *
*                                                                 *
* Notices    : This module is part of the distributed source      *
*             code for PANAPT.                                    *
*                                                                 *
*             Copyright (C) 1992, 1996 Computer Associates       *030A*BAD
*             International Inc. All rights reserved.             *
*                                                                 *
*             This software is proprietary information and its    *
*             use by unauthorized persons is prohibited.          *
*                                                                 *
****************** < Description End        > ********************
                                                                  020A*BAD
****************** < Documentation Begin    > ********************
*                                                                 *
* Function   : To provide a common record description for the     *
*             Move Request Description record.                    *
*                                                                 *
* Related to :APAMMDES must reflect all changes to this member.   *
*                                                                 *
* Comments   :APAMMDES MUST Reflect ALL Changes to this Member.   *
*                                                                 *
****************** < Documentation End      > ********************
                                                                  020A*BAD
                                                                  020A*BAD
 01  MOVE-REQUEST-DESCRIPTION.
     02  MR-RECORD-LEN           PIC 9(05)      COMP.             030A*BAD
     02  MR-RECORD-TYPE          PIC X(02)      VALUE SPACE.
       88  MR-RECORD-IS-DESC                   VALUE '01'.       013A*GRS
     02  MR-USER-ID              PIC X(08)      VALUE SPACE.      020A*SYW
     02  MR-NUMBER               PIC 9(06)      VALUE ZERO.       020A*SYW
                                                                  020A*BAD
*
*       MR-VERSION-STAMP is in the format VV.R where VV is the
```

```
*       version and R is the release, as in 02.0                  020A*BAD
*       The version is filled in by the DUMP/PURGE programs and
*       by the exit facilities. You must fill in the version
*       if you are constructing a record for the batch add
*       program.
 *
     02  MR-VERSION-STAMP        PIC X(04)      VALUE SPACE.      020A*BAD
                                                                  020A*BAD
     02  MR-PROCESS              PIC X(01)      VALUE SPACE.
       88  MR-PROCESS-IS-PURGE                  VALUE 'P'.
       88  MR-PROCESS-IS-DUMP                   VALUE 'D'.
       88  MR-PROCESS-IS-NIL                    VALUE 'N'.
     02  MR-DATE-TIME-OF-PROCESS.
       03  MR-DATE-OF-PROCESS.
         04  MR-PROCESS-CC       PIC X(02)      VALUE SPACE.      013A*GRS
         04  MR-PROCESS-YY       PIC X(02)      VALUE SPACE.
         04  MR-PROCESS-MM       PIC X(02)      VALUE SPACE.
         04  MR-PROCESS-DD       PIC X(02)      VALUE SPACE.
       03  MR-TIME-OF-PROCESS.
         04  MR-PROCESS-HH       PIC X(02)      VALUE SPACE.
         04  MR-PROCESS-MM       PIC X(02)      VALUE SPACE.
         04  MR-PROCESS-SS       PIC X(02)      VALUE SPACE.
                                                                  020A*SYW
*                                                                 020A*SYW
*       Counters that control the size of OCCURS DEPENDING ON     020A*SYW
*       area sizes and the locations of fields that follow.       020A*SYW
*                                                                 020A*SYW
     02  MR-EXP-DESC-COUNT       PIC S9(04)     COMP VALUE ZERO.  020A*SYW
     02  MR-COMPLETED-MOVE-COUNT PIC  9(02)     COMP VALUE ZERO.  020A*SYW
                                                                  020A*BAD
     02  MR-STATUS               PIC X(06)      VALUE SPACE.      020A*BAD
*        The MR-CURRENT-... fields compliment the MR-STATUS       020A*BAD
*        field. The MR-CURRENT-LEVEL-ID contains the internal     020A*BAD
*        unique name CA PanAPT uses for a level. The ABBR and     020A*BAD
*        SNAME contain the names the users accociate with the     020A*BAD
*        level, such as "P" and "PROD"  for a Production level.   020A*BAD
*        The POS contains the position of this level in regards   020A*BAD
*        to other levels in the CA PanAPT system, with the        020A*BAD
*        starting level being 1. If you add or delete levels      020A*BAD
*        from your system, the POS field will change. If you      020A*BAD
*        change the names of levels, the ABBR and SNAME fields    020A*BAD
*        will change, but the ID field will not.                  020A*BAD
```

```
02  MR-CURRENT-LEVEL-ID     PIC 9(09) COMP.                 020A*BAD
02  MR-CURRENT-LEVEL-ABBR   PIC X(02).                      020A*BAD
02  MR-CURRENT-LEVEL-SNAME  PIC X(04).                      020A*BAD
02  MR-CURRENT-LEVEL-POS    PIC 9(04) COMP.                 020A*BAD
02  MR-CURRENT-LEVEL-STATUS PIC X(02).                      020A*BAD
  88  STATUS-IS-DELETED                   VALUE '00'.       020A*BAD
  88  STATUS-IS-BEING-CREATED             VALUE '01'.       020A*BAD
  88  STATUS-IS-AWAIT-APPROVALS           VALUE '02'.       020A*BAD
  88  STATUS-IS-APPROVED                  VALUE '03'.       020A*BAD
  88  STATUS-IS-SEL-FOR-MOVE              VALUE '04'.       020A*BAD
  88  STATUS-IS-AWAITING-MOVE             VALUE '05'.       020A*BAD
  88  STATUS-IS-AWAITING-EXTRN            VALUE '06'.       020A*BAD
  88  STATUS-IS-MOVE-COMPLETE             VALUE '07'.       020A*BAD
  88  STATUS-IS-AWAIT-BACK-APP            VALUE '22'.       020A*BAD
  88  STATUS-IS-APP-FOR-BKOT              VALUE '23'.       020A*BAD
  88  STATUS-IS-SEL-FOR-BKOT              VALUE '24'.       020A*BAD
  88  STATUS-IS-AWAIT-BKOT                VALUE '25'.       020A*BAD
  88  STATUS-IS-AWAIT-BKOT-EP             VALUE '26'.       020A*BAD
  88  STATUS-IS-BKOT-COMPLETE             VALUE '27'.       020A*BAD
02  MR-MOVE-TYPE            PIC X(01)      VALUE SPACE.
02  MR-SERVICE-REQUEST      PIC X(16)      VALUE SPACE.
02  MR-NAME                 PIC X(16)      VALUE SPACE.      030A*BAD
02  MR-PROJECT              PIC X(16)      VALUE SPACE.      030A*BAD
02  MR-NUMBER-OF-MEMBERS    PIC S9(5)      COMP VALUE ZERO.
                                                             020A*BAD
02  MR-DATES.
  03  MR-FINAL-MOVE-DATE.                                   020A*BAD
    04  MR-MV-CC            PIC X(02)      VALUE SPACE.      013A*GRS
    04  MR-MV-YY            PIC X(02)      VALUE SPACE.
    04  MR-MV-MM            PIC X(02)      VALUE SPACE.
    04  MR-MV-DD            PIC X(02)      VALUE SPACE.
  03  MR-NEXT-MOVE-DATE.                                    020A*BAD
    04  MR-NXMV-CC          PIC X(02)      VALUE SPACE.      020A*BAD
    04  MR-NXMV-YY          PIC X(02)      VALUE SPACE.      020A*BAD
    04  MR-NXMV-MM          PIC X(02)      VALUE SPACE.      020A*BAD
    04  MR-NXMV-DD          PIC X(02)      VALUE SPACE.      020A*BAD
  03  MR-SCHEDULED-RUN-DATE.                                013A*GRS
    04  MR-SR-CC            PIC X(02)      VALUE SPACE.      013A*GRS
    04  MR-SR-YY            PIC X(02)      VALUE SPACE.
    04  MR-SR-MM            PIC X(02)      VALUE SPACE.
    04  MR-SR-DD            PIC X(02)      VALUE SPACE.
02  MR-MOVE-INFORMATION.
```

```
       03  MR-SPCL-HANDLING      PIC X(01)      VALUE SPACE.
       03  MR-EARLY-STOP-SNAME   PIC X(04)      VALUE SPACE.      020A*MMC
                                                                  020A*BAD
     02  MR-ADD-INFORMATION.
       03  MR-ADD-USER-ID        PIC X(08)      VALUE SPACE.
       03  MR-ADD-DATE.
         04  MR-ADD-CC           PIC X(02)      VALUE SPACE.      013A*GRS
         04  MR-ADD-YY           PIC X(02)      VALUE SPACE.
         04  MR-ADD-MM           PIC X(02)      VALUE SPACE.
         04  MR-ADD-DD           PIC X(02)      VALUE SPACE.
       03  MR-ADD-TIME.
         04  MR-ADD-HH           PIC X(02)      VALUE SPACE.
         04  MR-ADD-MM           PIC X(02)      VALUE SPACE.
         04  MR-ADD-SS           PIC X(02)      VALUE SPACE.
                                                                  020A*BAD
     02  MR-LAST-UPDATED-BY-INFO.
       03  MR-UPD-USER-ID        PIC X(08)      VALUE SPACE.
       03  MR-UPD-DATE.
         04  MR-UPD-CC           PIC X(02)      VALUE SPACE.      013A*GRS
         04  MR-UPD-YY           PIC X(02)      VALUE SPACE.
         04  MR-UPD-MM           PIC X(02)      VALUE SPACE.
         04  MR-UPD-DD           PIC X(02)      VALUE SPACE.
       03  MR-UPD-TIME.
         04  MR-UPD-HH           PIC X(02)      VALUE SPACE.
         04  MR-UPD-MM           PIC X(02)      VALUE SPACE.
         04  MR-UPD-SS           PIC X(02)      VALUE SPACE.
     SKIP2
     02  MR-APPROVALS-AREA.                                      013A*GRS
**      *-------------------------------------------------------*
**      *     A P P R O V A L S   A R E A                       *
**      *-------------------------------------------------------*
                                                                  020A*BAD
       03  MR-APPROVALS-NEEDED-AREA.                            013A*GRS
**        *-----------------------------------------------------*
**        *     A P P R O V A L S   N E E D E D   A R E A     *
**        *-----------------------------------------------------*
         04  MR-CL-APPROVALS-NEEDED-AREA.                        020A*MMC
**         *-----------------------------------------------------*
**         *     Approvals for CL area.                        * 020A*MMC
**         *-----------------------------------------------------*
           05  MR-CL-APPROVALS-NEEDED PIC X(01) OCCURS 20 TIMES   020A*MMC
                                 INDEXED BY MR-CL-NEEDED-INDEX.   020A*MMC
```

```
           88  CL-APPROVALS-NEEDED-IS-YES     VALUE 'Y'.        020A*MMC
           88  CL-APPROVALS-NEEDED-IS-NO      VALUE 'N'.        020A*MMC
                                                                 020A*BAD
        04  MR-BO-APPROVALS-NEEDED-AREA.                        013A*GRS
**         *-----------------------------------------------*
**         *     Approvals for BACKOUT area.               *
**         *-----------------------------------------------*
          05  MR-BO-APPROVALS-NEEDED PIC X(01) OCCURS 20 TIMES   013A*GRS
                              INDEXED BY MR-BO-NEEDED-INDEX.   013A*GRS
           88  BO-APPROVALS-NEEDED-IS-YES     VALUE 'Y'.        013A*GRS
           88  BO-APPROVALS-NEEDED-IS-NO      VALUE 'N'.        013A*GRS
                                                                 020A*BAD
        04  MR-CL-VER-NEEDED-AREA.                              020A*MMC
**         *-----------------------------------------------*
**         *     Verify CL flag area.                      * 020A*MMC
**         *-----------------------------------------------*
          05  MR-CL-VER-APPROVALS-NEEDED PIC X(01)               020A*MMC
        OCCURS 20 TIMES                                          020A*MMC
                            INDEXED BY MR-CL-VER-NEEDED-INDEX.   020A*MMC
           88  CL-VER-APRV-NEEDED-IS-YES      VALUE 'Y'.        020A*MMC
           88  CL-VER-APRV-NEEDED-IS-NO       VALUE 'N'.        020A*MMC
                                                                 020A*BAD
                                                                 020A*BAD
      03  MR-APPROVALS-SO-FAR-AREA.                             013A*GRS
**       *-----------------------------------------------*
**       *     A P P R O V A L S   S O   F A R   A R E A     *
**       *-----------------------------------------------*
                                                                 020A*BAD
        04  MR-CL-APPROVALS-SO-FAR-AREA.                        020A*MMC
**         *-----------------------------------------------* 020A*MMC
**         *     So Far Approvals for CL.                  * 020A*MMC
**         *-----------------------------------------------* 020A*MMC
          05  MR-CL-APPROVALS-SO-FAR PIC X(01) OCCURS 20 TIMES   020A*MMC
                              INDEXED BY MR-CL-SO-FAR-INDEX.   020A*MMC
           88  CL-APPROVALS-SO-FAR-IS-YES     VALUE 'Y'.        020A*MMC
           88  CL-APPROVALS-SO-FAR-IS-NO      VALUE 'N'.        020A*MMC
                                                                 020A*BAD
        04  MR-BO-APPROVALS-SO-FAR-AREA.                        013A*GRS
**         *-----------------------------------------------*
**         *     So Far Approvals for BACKOUT.             *
**         *-----------------------------------------------*
          05  MR-BO-APPROVALS-SO-FAR PIC X(01) OCCURS 20 TIMES   013A*GRS
```

```
                              INDEXED BY MR-BO-SO-FAR-INDEX.   013A*GRS
             88  BO-APPROVALS-SO-FAR-IS-YES     VALUE 'Y'.        013A*GRS
             88  BO-APPROVALS-SO-FAR-IS-NO      VALUE 'N'.        013A*GRS
                                                                  020A*BAD
         04  MR-CL-VER-SO-FAR-AREA.                              020A*MMC
**           *-------------------------------------------------*
**           *     So Far Verification for CL.                 * 020A*MMC
**           *-------------------------------------------------*
           05  MR-CL-VER-APPROVALS-SO-FAR PIC X(01)              020A*MMC
         OCCURS 20 TIMES                                          013A*GRS
                          INDEXED BY MR-CL-VER-SO-FAR-INDEX.     020A*MMC
             88  CL-VER-APRV-SO-FAR-IS-YES      VALUE 'Y'.       020A*MMC
             88  CL-VER-APRV-SO-FAR-IS-NO       VALUE 'N'.       020A*MMC
                                                                  020A*BAD
                                                                  020A*BAD
     02  FILLER              REDEFINES MR-APPROVALS-AREA.        013A*GRS
       03  FILLER OCCURS 2 TIMES                                 013A*GRS
                       INDEXED BY MR-AP-ND-INDEX MR-AP-SF-INDEX. 013A*GRS
**           *-------------------------------------------------*
**           *     A P P R O V A L S   (redefined)             *
**           *-------------------------------------------------*
         04  MR-APPROVALS        PIC X(01) OCCURS 60 TIMES       020A*MMC
                                  INDEXED BY MR-AP-INDEX.        020A*MMC
                                                                  020A*MMC
                                                                  020A*BAD
     02  MR-DESCRIPTION-AREA.
       03  MR-DESCRIPTION        PIC X(55)      VALUE SPACE.
       03  MR-EXP-MAX-DESC-INDEX           INDEX.                020A*SYW
       03  MR-EXPANDED-DESCRIPTION-AREA.                         013A*BAD
         04  MR-EXPANDED-DESCRIPTION       PIC X(70)            020A*SYW
                           OCCURS 0 TO 12 TIMES                  020A*SYW
                           DEPENDING ON MR-EXP-DESC-COUNT        020A*SYW
                           INDEXED BY MR-EXT-DESC-INDEX.         020A*SYW
                                                                 020A*BAD
                                                                 020A*BAD
     02  MR-COMPLETED-MOVE-AREA.                                020A*BAD
       03  MR-COMPLETED-MOVE                                    020A*BAD
                          occurs 0 to 15 times                   020A*BAD
                          depending on MR-COMPLETED-MOVE-COUNT   020A*BAD
                          indexed by MR-COMPLETED-INDEX.         020A*BAD
*            MR-COMPLETED-LEVEL-ID is the internal unique name  020A*BAD
*            CA PanAPT uses for a level. MR-COMPLETED-LEVEL-ABBR 020A*BAD
```

```
*           and MR-COMPLETED-LEVEL-SNAME are the names the users 020A*BAD
*           associate with the level, such as "P" and "PROD"     020A*BAD
*           for a Production level. MR-COMPLETE-LEVEL-POS        020A*BAD
*           contains the position of this level in regards to    020A*BAD
*           other levels in the CA PanAPT system, with the       020A*BAD
*           starting level being 1.                              020A*BAD
        04  MR-COMPLETED-LEVEL-ID         PIC 9(09) COMP.       020A*BAD
        04  MR-COMPLETED-LEVEL-ABBR       PIC X(02).            020A*BAD
        04  MR-COMPLETED-LEVEL-SNAME      PIC X(04).            020A*BAD
        04  MR-COMPLETED-LEVEL-POS        PIC 9(04) COMP.       020A*BAD
*           Date scheduled to be moved:                          020A*BAD
        04  MR-COMPLETED-DATE-SCHED.                            020A*BAD
          05  MR-COMP-DS-CC               PIC X(02).            020A*BAD
          05  MR-COMP-DS-YY               PIC X(02).            020A*BAD
          05  MR-COMP-DS-MM               PIC X(02).            020A*BAD
          05  MR-COMP-DS-DD               PIC X(02).            020A*BAD
*           Date actually moved. Will contain spaces if an       020A*BAD
*           on-line status change bypassed the move.             020A*BAD
        04  MR-COMPLETED-DATE-MOVED.                            020A*BAD
          05  MR-COMP-DM-CC               PIC X(02).            020A*BAD
          05  MR-COMP-DM-YY               PIC X(02).            020A*BAD
          05  MR-COMP-DM-MM               PIC X(02).            020A*BAD
          05  MR-COMP-DM-DD               PIC X(02).            020A*BAD
        04  MR-COMPLETED-TIME-MOVED.                            020A*BAD
          05  MR-COMP-DM-HH               PIC X(02).            020A*BAD
          05  MR-COMP-DM-MM               PIC X(02).            020A*BAD
          05  MR-COMP-DM-SS               PIC X(02).            020A*BAD
```

Move Request Member Record Mapped by APCCMMBR

The following record layout, mapped by COBOL member APCCMMBR, describes the Move Request Member record that is produced by Create Sequential File of Move Requests from the History file. This record is input to the Batch Add Move Request Job (APJJ5960) and passed to the security exit program and the member existence exit.

```
************************** < Description Begin      > ********************
*                                                                        *
* Name       : APCCMMBR                                                  *
* Product    : PANAPT                                                    *
* Type       : Cobol Copybook                                            *
*                                                                        *
* Descript. : Move Request Member record layout used by Member           *
*               Existence Exits, Security Exits, Batch Add Move          *
*               Move Requests and Purge/Dump Move Requests.              *
*                                                                        *
* Notices    : This module is part of the distributed source             *
*               code for CA PanAPT.                                      *030A*BAD
*                                                                        *
*               Copyright (C) 1992, 1996 Computer Associates             *030A*BAD
*               International Inc. All rights reserved.                  *
*                                                                        *
*               This software is proprietary information and its         *
*               use by unauthorized persons is prohibited.               *
*                                                                        *
****************** < Description End        > ********************
     SKIP1
****************** < Documentation Begin    > ********************
*                                                                        *
* Function   : To provide a common record description for the            *
*               Move Request Member record.                              *
*                                                                        *
* Related to : APAMMMBR must reflect changes made to this member.*
 *                                                                        *
* Comments   : None.                                                     *
*                                                                        *
****************** < Documentation End      > ********************
     SKIP1
     SKIP1
 01  MOVE-REQUEST-MEMBER.
     02  MR-RECORD-LEN           PIC 9(05)      COMP.            030A*BAD
     02  MR-RECORD-TYPE          PIC X(02)      VALUE SPACE.
       88  MR-RECORD-IS-MBR                    VALUE '02'.      013A*GRS
     02  MR-USER-ID              PIC X(08)      VALUE SPACE.     020A*SYW
     02  MR-NUMBER               PIC 9(06)      VALUE ZERO.      020A*SYW
     SKIP1                                                       013A*GRS
     02  MR-VERSION-STAMP        PIC X(04)      VALUE SPACE.     012A*JMM
*
```

```
*       MR-VERSION-STAMP is in the format VV.R where VV is the
*       version and R is the release, as in 02.0                020A*BAD
*       The version is filled in by the DUMP/PURGE programs and
*       by the exit facilities. You must fill in the version
*       if you are constructing a record for the batch add
*       program.
 *
     SKIP1                                                      012A*JMM
     02  MR-PROCESS              PIC X(01)      VALUE SPACE.    012A*JMM
       88  MR-PROCESS-IS-PURGE                  VALUE 'P'.
       88  MR-PROCESS-IS-DUMP                   VALUE 'D'.
       88  MR-PROCESS-IS-NIL                    VALUE 'N'.
       88  MR-PROCESS-IS-ADD                    VALUE 'A'.      020A*SYW
       88  MR-PROCESS-IS-CHG                    VALUE 'G'.      020A*SYW
       88  MR-PROCESS-IS-DEL                    VALUE 'X'.      020A*SYW
       88  MR-PROCESS-IS-MPURGE                 VALUE 'M'.      020A*MMC
       88  MR-PROCESS-IS-ASN                    VALUE 'S'.      020A*SYW
       88  MR-PROCESS-IS-RETRIEVE               VALUE 'K'.      030A*BAD
       88  MR-PROCESS-IS-ACK                    VALUE 'K'.      020A*SYW
       88  MR-PROCESS-IS-CDSET                  VALUE '1'.      020A*BAD
       88  MR-PROCESS-IS-CDCLR                  VALUE '2'.      020A*BAD
     02  MR-DATE-TIME-OF-PROCESS.
       03  MR-DATE-OF-PROCESS.
         04  MR-PROCESS-CC       PIC X(02)      VALUE SPACE.    013A*GRS
         04  MR-PROCESS-YY       PIC X(02)      VALUE SPACE.
         04  MR-PROCESS-MM       PIC X(02)      VALUE SPACE.
         04  MR-PROCESS-DD       PIC X(02)      VALUE SPACE.
       03  MR-TIME-OF-PROCESS.
         04  MR-PROCESS-HH       PIC X(02)      VALUE SPACE.
         04  MR-PROCESS-MM       PIC X(02)      VALUE SPACE.
         04  MR-PROCESS-SS       PIC X(02)      VALUE SPACE.
     SKIP1
     02  MR-FROM-MEMBER          PIC X(10)      VALUE SPACE.
     02  MR-FROM-USER-DATA       PIC X(08)      VALUE SPACE.
     02  MR-TO-MEMBER            PIC X(10)      VALUE SPACE.
     02  MR-TO-USER-DATA         PIC X(08)      VALUE SPACE.
     02  MR-LIBRARY-CODE.
       03  MR-LIB-CODE           PIC X(04)      VALUE SPACE.
       03  MR-LIBSUBCODE         PIC X(03)      VALUE SPACE.
     SKIP1
     02  MR-RESTRICTION-FLAGS.                                 1009690
       03  MR-ASSIGN-FLAG        PIC X(01)      VALUE SPACE.    1009690
```

```
          88  MR-AWATING-ASSIGNMENT              VALUE 'A'.       1009690
          88  MR-ASSIGNMENT-COMPLETE             VALUE SPACE.     1009690
       SKIP1                                                       1009690
         03  MR-DIBS-FLAG           PIC X(01)     VALUE SPACE.     1009690
          88  MR-WAITING-DIBS-APPROVAL           VALUE 'I'.       1009690
          88  MR-DIBS-RECORD-APPROVED            VALUE SPACE.     1009690
       SKIP1                                                       1009690
         03  MR-VERIFY-FLAG         PIC X(01)     VALUE SPACE.     1009690
          88  MR-MEMBER-NOT-VERIFIED             VALUE 'V'.       1009690
          88  MR-MEMBER-VERIFIED                 VALUE SPACE.     1009690
       SKIP1                                                       1009690
         03  MR-MOVE-FLAG           PIC X(01)     VALUE SPACE.     1009690
          88  MR-AWAITING-MODELLING              VALUE '.'.       013A*MCV
          88  MR-AWAITING-MOVE                   VALUE 'M'.       1009690
          88  MR-MEMBER-MOVED                    VALUE SPACE.     1009690
       SKIP1                                                       1009690
         03  MR-EXTERNAL-FLAG       PIC X(01)     VALUE SPACE.     1009690
          88  MR-EXTERNAL-REQUIRED               VALUE 'E'.       1009690
          88  MR-EXTERNAL-STARTED                VALUE 'S'.       1009690
          88  MR-EXTERNAL-COMPLETE               VALUE SPACE.     1009690
       SKIP1                                                       1009690
         03  MR-CONCURRENT-FLAG     PIC X(01)     VALUE SPACE.     020A*SYW
         03  FILLER                 PIC X(02)     VALUE SPACE.     020A*SYW
       SKIP1                                                       1009690
*    following fields were added for PANAPT GUI project           020A*SYW
*    internal processing.                                         020A*SYW
       SKIP1                                                       020A*SYW
       02  MR-MBR-KEY.                                           020A*SYW
         03  MR-MBR-KEY-QUALIFIER   PIC X(08)     VALUE SPACE.     020A*SYW
         03  MR-MBR-KEY-PRODNAME    PIC X(10)     VALUE SPACE.     020A*SYW
       02  MR-MBR-OLD-LIBCODE       PIC X(04)     VALUE SPACE.     020A*SYW
       02  MR-MBR-OLD-SUBCODE       PIC X(03)     VALUE SPACE.     020A*SYW
       02  FILLER                   PIC X(10)     VALUE SPACE.     1009690
```

Move Request Approval Record Mapped by APCCMAPP

The following record layout, mapped by COBOL member APCCMAPP, describes the Move Request approval record produced by the Create Sequential File of Move Requests from the History File Job (APJJ5955). This record is input to the Batch Add Move Request Job (APJJ5960) and passed to the security exit program.

```
****************** < Description Begin      > ********************
*                                                                 *
* Name       : APCCMAPP                                           *
* Product    : PANAPT                                             *
* Type       : Cobol Copybook                                     *
*                                                                 *
* Descript. : Move Request Approval record layout used by the     *
*               Security Exits, Batch Add Move Request, and       *
*               Purge/Dump Move Requests.                         *
*                                                                 *
* Notices    : This module is part of the distributed source      *
*               code for PANAPT.                                  *
*                                                                 *
*               Copyright (C) 1992, 1996 Computer Associates      *030A*BAD
*               International Inc. All rights reserved.           *
*                                                                 *
*               This software is proprietary information and its  *
*               use by unauthorized persons is prohibited.        *
*                                                                 *
****************** < Description End        > ********************
                                                                    020A*BAD
****************** < Documentation Begin    > ********************
*                                                                 *
* Function   : To provide a common record description for the     *
*               Move Request Approval record.                     *
*                                                                 *
* Related to : None.                                              *
*                                                                 *
* Comments   : None.                                              *
*                                                                 *
****************** < Documentation End      > ********************
                                                                    020A*BAD
                                                                    020A*BAD
 01  MOVE-REQUEST-APPROVAL.
     02  MR-RECORD-LEN           PIC 9(05)      COMP.             030A*BAD
     02  MR-RECORD-TYPE          PIC X(02).
       88  MR-RECORD-IS-APP                    VALUE '03'.       013A*GRS
     02                          PIC X(08).                       020A*MMC
     02  MR-NUMBER               PIC 9(06).                       020A*MMC
     02  MR-VERSION-STAMP        PIC X(04).                       012A*JMM
*       MR-VERSION-STAMP is in the format VV.R where VV is the
*       version and R is the release, as in 02.0                  020A*BAD
```

```
                                                                020A*BAD
02  MR-PROCESS              PIC X(01).
  88  MR-PROCESS-IS-PURGE                VALUE 'P'.
  88  MR-PROCESS-IS-DUMP                 VALUE 'D'.
  88  MR-PROCESS-IS-NIL                  VALUE 'N'.
02  MR-DATE-TIME-OF-PROCESS.
  03  MR-DATE-OF-PROCESS.
    04  MR-PROCESS-CC       PIC X(02).                     013A*GRS
    04  MR-PROCESS-YY       PIC X(02).
    04  MR-PROCESS-MM       PIC X(02).
    04  MR-PROCESS-DD       PIC X(02).
  03  MR-TIME-OF-PROCESS.
    04  MR-PROCESS-HH       PIC X(02).
    04  MR-PROCESS-MM       PIC X(02).
    04  MR-PROCESS-SS       PIC X(02).
                                                                020A*BAD
02  MR-APPROVAL-LEVEL       PIC X(04).
02  MR-APPROVAL-DIRECTION   PIC 9(01).                      020A*BAD
  88  MR-APPROVAL-DIR-MOVE               VALUE 0.          020A*BAD
  88  MR-APPROVAL-DIR-BKOT               VALUE 1.          020A*BAD
02  MR-APPROVAL-CATEGORY    PIC 9(02).
02  MR-APPROVAL-FLAG        PIC X(01).
  88  MR-IS-APPROVED                     VALUE 'Y'.
  88  MR-IS-DISAPPROVED                  VALUE 'D'.
  88  MR-IS-UNAPPROVED                   VALUE 'N'.
                                                                020A*BAD
02  MR-APPROVAL-USERID      PIC X(08).
02  MR-APPROVAL-DATE.
  04  MR-APPROVAL-CC        PIC X(02).                     013A*GRS
  04  MR-APPROVAL-YY        PIC X(02).
  04  MR-APPROVAL-MM        PIC X(02).
  04  MR-APPROVAL-DD        PIC X(02).
02  MR-APPROVAL-TIME.
  04  MR-APPROVAL-HH        PIC X(02).
  04  MR-APPROVAL-MM        PIC X(02).
  04  MR-APPROVAL-SS        PIC X(02).
                                                                020A*BAD
02  MR-COMMENT-COUNT        PIC S9(04)     COMP.
02  MR-COMMENT-MAX-INDEX                   INDEX.
02  MR-COMMENT-RECORD       PIC X(70) OCCURS 6 TIMES
                                INDEXED BY MR-COMMENT-INDEX.
```

Move Request Verification Record Mapped by APCCMVER

The following record layout, mapped by COBOL member APCCMVER, describes the Move Request verification record produced by the Create Sequential File of Move Requests from the History File Job (APJJ5955).

```
****************** < Description Begin      > ********************
*                                                                 *
* Name       : APCCMVER                                           *
* Product    : PANAPT                                             *
* Type       : Cobol Copybook                                     *
*                                                                 *
* Descript. : Move Request Approval record layout used by the     *
*               Security Exits, Batch Add Move Request, and       *
*               Purge/Dump Move Requests.                         *
*                                                                 *
* Notices    : This module is part of the distributed source      *
*               code for PANAPT.                                  *
*                                                                 *
*               Copyright (C) 1992, 1996 Computer Associates      *030A*BAD
*               International Inc. All rights reserved.           *
*                                                                 *
*               This software is proprietary information and its  *
*               use by unauthorized persons is prohibited.        *
*                                                                 *
****************** < Description End        > ********************
     SKIP1
****************** < Documentation Begin    > ********************
*                                                                 *
* Function   : To provide a common record description for the     *
*               Move Request Verification Record Layout.          *
*                                                                 *
* Related to : None.                                              *
*                                                                 *
* Comments   : None.                                              *
*                                                                 *
****************** < Documentation End      > ********************
     SKIP1
     SKIP1
 01  MOVE-REQUEST-VERIFICATION.
     SKIP1
     02  MR-RECORD-LEN           PIC 9(05)      COMP.            5160597
     02  MRV-RECORD-TYPE         PIC X(02).                      013A*BAD
       88  MRV-RECORD-IS-MRV                    VALUE '04'.      013A*BAD
     02                          PIC X(08).                      020A*MMC
     02  MRV-REQ-NUM             PIC 9(06).                      020A*MMC
     02  MRV-VERSION-STAMP       PIC X(04).                      013A*BAD
*       MRV-VERSION-STAMP is in the format VV.R where VV is the
```

```
*       the version and R is the release, as in 01.3
     SKIP1                                                    013A*BAD
     02  MRV-PROCESS             PIC X(01).                  013A*BAD
       88  MRV-PROCESS-IS-PURGE                VALUE 'P'.    013A*BAD
       88  MRV-PROCESS-IS-DUMP                 VALUE 'D'.    013A*BAD
       88  MRV-PROCESS-IS-NIL                  VALUE 'N'.    013A*BAD
     02  MRV-DATE-TIME-OF-PROCESS.                           013A*BAD
       03  MRV-DATE-OF-PROCESS.                              013A*BAD
         04  MRV-PROCESS-CC      PIC X(02).                  013A*BAD
         04  MRV-PROCESS-YY      PIC X(02).                  013A*BAD
         04  MRV-PROCESS-MM      PIC X(02).                  013A*BAD
         04  MRV-PROCESS-DD      PIC X(02).                  013A*BAD
       03  MRV-TIME-OF-PROCESS.                              013A*BAD
         04  MRV-PROCESS-HH      PIC X(02).                  013A*BAD
         04  MRV-PROCESS-MM      PIC X(02).                  013A*BAD
         04  MRV-PROCESS-SS      PIC X(02).                  013A*BAD
     SKIP1                                                    013A*BAD
     02  MRV-VER-LEVEL           PIC X(04).                  013A*BAD
     02  MRV-VER-DIRECTION       PIC 9(01).                  020A*BAD
       88  MRV-VER-DIR-MOVE                    VALUE 0.      020A*BAD
       88  MRV-VER-DIR-BKOT                    VALUE 1.      020A*BAD
     02  MRV-VER-CATEGORY        PIC 9(02).                  013A*BAD
     02  MRV-VERIFICATION-DATE.                              013A*GRS
       03  MRV-DATE-CC           PIC X(02).
       03  MRV-DATE-YY           PIC X(02).
       03  MRV-DATE-MM           PIC X(02).
       03  MRV-DATE-DD           PIC X(02).
     02  MRV-VERIFICATION-TIME.                              013A*GRS
       03  MRV-TIME-HH           PIC X(02).
       03  MRV-TIME-MM           PIC X(02).
       03  MRV-TIME-SS           PIC X(02).
     02  MRV-VERIFICATION-FLAG   PIC X(01).                  013A*GRS
       88  MRV-VERIFICATION-STARTED            VALUE 'S'.    013A*BAD
       88  MRV-VERIFICATION-COMPLETE           VALUE 'Y'.    013A*BAD
       88  MRV-VERIFICATION-FAILED             VALUE 'U'.    013A*GRS
     02  MRV-VER-FAILURE-REASON  PIC X(04).
     02  MRV-COMMENT-COUNT       PIC S9(04)     COMP.
     02  MRV-MAX-INDEX                          INDEX.
     02  MRV-COMMENT             PIC X(70) OCCURS 6 TIMES
                                 INDEXED BY MRV-INDEX.
```

Batch Interface Input Record Layout Mapped by APCCIREQ

The following record layout, mapped by COBOL copybook member APCCIREQ, describes the input record layout for the Batch Interface.

```
****************** < Description Begin      > ********************
*                                                                 *
* Name       : APCCIREQ                                           *
* Product    : PANAPT                                             *
* Type       : Cobol Copybook                                     *
*                                                                 *
* Descript. : Panapt batch interface input request 00 record.    *020A*SYW
*                                                                 *
* Notices    : This module is part of the distributed source      *
*              code for PANAPT batch interface.                   *020A*SYW
*                                                                 *
*              Copyright (C) 1994, 1996 Computer Associates       *030A*BAD
*              International Inc. All rights reserved.            *
*                                                                 *
*              This software is proprietary information and its   *
*              use by unauthorized persons is prohibited.         *
*                                                                 *
****************** < Description End        > ********************
                                                                    020A*BAD
                                                                    020A*BAD
 01  APCSIREQ-INPUT-REQUEST.
     02  APCSIREQ-RECORD-LEN     PIC 9(05)      COMP.              030A*BAD
     02  APCSIREQ-RECORD-TYPE    PIC X(02)      VALUE SPACE.
       88  APCSIREQ-RECORD-IS-REQUEST           VALUE '00'.
       88  APCSIREQ-RECORD-IS-DESC              VALUE '01'.
       88  APCSIREQ-RECORD-IS-MBR               VALUE '02'.
       88  APCSIREQ-RECORD-IS-LIB3              VALUE '03'.       020A*SYW
       88  APCSIREQ-RECORD-IS-DIB2              VALUE '04'.       020A*SYW
       88  APCSIREQ-RECORD-IS-MSG               VALUE '07'.       020A*SYW
       88  APCSIREQ-RECORD-IS-CTL               VALUE '08'.       020A*SYW
       88  APCSIREQ-RECORD-IS-LMR               VALUE '09'.       020A*SYW
       88  APCSIREQ-RECORD-IS-MSL               VALUE '10'.       020A*SYW
     02  APCSIREQ-USER-ID        PIC X(08)      VALUE SPACE.
     02  APCSIREQ-NUMBER         PIC 9(06)      VALUE ZERO.
                                                                    020A*BAD
     02  APCSIREQ-VERSION-STAMP  PIC X(04)      VALUE SPACE.
 *
*       APCSIREQ-VERSION-STAMP is in the format VV.R where VV is  020A*BAD
*       the version and R is the release, as in 02.0.             020A*BAD
*       You must filled in the version if you are constructing    020A*BAD
*       a record for the batch interface program.                 020A*BAD
*
```

```
                                                                  020A*BAD
02  APCSIREQ-PROCESS        PIC X(01)      VALUE SPACE.
  88  APCSIREQ-PROCESS-IS-ADD              VALUE 'A'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-LLB              VALUE 'B'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-CLO              VALUE 'C'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-ENQ              VALUE 'E'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-CHG              VALUE 'G'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-LIV              VALUE 'I'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-MSLLIB           VALUE 'J'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-MSLINV           VALUE 'K'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-LMR              VALUE 'L'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-MPURGE           VALUE 'M'.        WOOLWRTH
  88  APCSIREQ-PROCESS-IS-COP              VALUE 'O'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-INQ              VALUE 'Q'.
  88  APCSIREQ-PROCESS-IS-RVP              VALUE 'R'.
  88  APCSIREQ-PROCESS-IS-INIT             VALUE 'T'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-DEQ              VALUE 'U'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-VER              VALUE 'V'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-INQENQ           VALUE 'W'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-DEL              VALUE 'X'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-QLB              VALUE 'Y'.        020A*SYW
  88  APCSIREQ-PROCESS-IS-QIV              VALUE 'Z'.        020A*SYW
02  APCSIREQ-BATCH-INPUT                   PIC X(01).
                                                              020A*SYW
02  APCSIREQ-MR-AREA.
    05  APCSIREQ-START-MR-NUM              PIC X(06).
    05  APCSIREQ-END-MR-NUM                PIC X(06).
    05  APCSIREQ-START-FINAL-DATE          PIC X(08).
    05  APCSIREQ-END-FINAL-DATE            PIC X(08).
    05  APCSIREQ-START-NEXT-DATE           PIC X(08).
    05  APCSIREQ-END-NEXT-DATE             PIC X(08).
    05  APCSIREQ-LIBCODE                   PIC X(04).
    05  APCSIREQ-SUBCODE                   PIC X(03).
    05  APCSIREQ-MEMBER-NAME               PIC X(10).
    05  APCSIREQ-ORIGINATOR                PIC X(08).
    05  APCSIREQ-SERVICE-REQ               PIC X(16).
    05  APCSIREQ-MOVE-TYPE                 PIC X(01).
    05  APCSIREQ-HELD-MR                   PIC X(01).       020A*SYW
    05  APCSIREQ-STATUS-AREA.
        15  APCSIREQ-STATUS-CREATE PIC X.
        15  APCSIREQ-STATUS-DELETE PIC X.
        15  APCSIREQ-STATUS-LEVEL OCCURS 16 TIMES
```

```
                        INDEXED BY APCSIREQ-STL-X.
              20 APCSIREQ-STATUS-LEVELID  PIC 9(09) COMP.     020A*SYW
              20 APCSIREQ-STATUS-SNAME    PIC X(04).          020A*SYW
              20 APCSIREQ-STATUS-TABLE.                       020A*SYW
                 25 APCSIREQ-STAT-AWAITING-APP  PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-APPROVED      PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-SELECTED      PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-AWAITING-MOV  PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-AWAITING-EP   PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-MOVED         PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-AWAITBKT-APP  PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-APPROVED-BKT  PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-SELECTED-BKT  PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-AWAITING-BKT  PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-AWAITBKT-EP   PIC X(01).    020A*SYW
                 25 APCSIREQ-STAT-BACKOUT       PIC X(01).    020A*SYW
              20 APCSIREQ-STATUS     REDEFINES                020A*SYW
                 APCSIREQ-STATUS-TABLE    PIC X(01)          020A*SYW
                 OCCURS 12 TIMES.                            020A*SYW
02  APCSIREQ-VP-AREA REDEFINES APCSIREQ-MR-AREA.
    05 APCSIREQ-VP-NUM    OCCURS 20 TIMES  PIC 9(02).
    05 APCSIREQ-VP-JOB1                    PIC X(80).       020A*SYW
    05 APCSIREQ-VP-JOB2                    PIC X(80).       020A*SYW
    05 APCSIREQ-VP-JOB3                    PIC X(80).       020A*SYW
    05 APCSIREQ-VP-JOB4                    PIC X(80).       020A*SYW
                                                           020A*SYW
02  APCSIREQ-LIBINV-LIST-AREA REDEFINES APCSIREQ-MR-AREA.   020A*SYW
    05  APCSIREQ-START-LIST-KEY.                           020A*SYW
        10 FILLER                          PIC X(01).       020A*SYW
        10 APCSIREQ-START-LIBSUB           PIC X(07).       020A*SYW
        10 APCSIREQ-START-MEMBER           PIC X(10).       020A*SYW
        10 FILLER                          PIC X(11).       020A*SYW
    05  APCSIREQ-END-LIST-KEY.                             020A*SYW
        10 FILLER                          PIC X(01).       020A*SYW
        10 APCSIREQ-END-LIBSUB             PIC X(07).       020A*SYW
        10 APCSIREQ-END-MEMBER             PIC X(10).       020A*SYW
        10 FILLER                          PIC X(11).       020A*SYW
02  APCSIREQ-MSL-LIB-AREA REDEFINES APCSIREQ-MR-AREA.       020A*SYW
    05  APCSIREQ-MSLLIB-LIBSUB          PIC X(07).         020A*SYW
    05  APCSIREQ-MSLLIB-MEMBER          PIC X(10).         020A*SYW
    05  APCSIREQ-MSLLIB-LEVEL           PIC X(04).         020A*SYW
02  APCSIREQ-MSL-INV-AREA REDEFINES APCSIREQ-MR-AREA.       020A*SYW
```

```
    05  APCSIREQ-MSLINV-LIBSUB          PIC X(07).          020A*SYW
    05  APCSIREQ-MSLINV-MEMBER          PIC X(10).          020A*SYW
    05  APCSIREQ-MSLINV-ASSIGN-USER     PIC X(08).          020A*SYW
    05  APCSIREQ-MSLINV-ASSIGN-MR       PIC X(06).          020A*SYW
    05  APCSIREQ-MSLINV-OWNER           PIC X(08).          020A*SYW
    05  APCSIREQ-MSLINV-ENVIRONMENT     PIC X(08).          020A*SYW
    05  APCSIREQ-MSLINV-APPLICATION     PIC X(08).          020A*SYW
    05  APCSIREQ-MSLINV-LANGUAGE        PIC X(08).          020A*SYW
    05  APCSIREQ-MSLINV-LAST-MOVED-MR   PIC X(06).          020A*SYW
    05  APCSIREQ-MSLINV-ASSIGNED        PIC X(01).          020A*SYW
    05  APCSIREQ-MSLINV-APPROVED        PIC X(01).          020A*SYW
02  APCSIREQ-JOB-AREA REDEFINES APCSIREQ-MR-AREA.          020A*SYW
    05  APCSIREQ-CKOT-JOB1              PIC X(80).          020A*SYW
    05  APCSIREQ-CKOT-JOB2              PIC X(80).          020A*SYW
    05  APCSIREQ-CKOT-JOB3              PIC X(80).          020A*SYW
    05  APCSIREQ-CKOT-JOB4              PIC X(80).          020A*SYW
```

Batch Interface Output Record Mapped by APCCMMSG

The following record layout, mapped by COBOL member APCCMMSG, describes the Batch Interface output message record layout.

```
****************** < DESCRIPTION BEGIN      > ********************
*                                                                  *
* NAME       : APCCMMSG                                            *
* Product    : PANAPT                                              *
* Type       : Cobol Copybook                                      *
*                                                                  *
* Descript. : PANAPT batch interface output records              *020A*SYW
*                                                                  *
*                                                                  *
* Notices    : This module is part of distributed source         *020A*SYW
*              code for PANAPT batch interface.                  *020A*SYW
*                                                                  *
*              Copyright (C) 1994, 1996 Computer Associates      *030A*BAD
*              International Inc. All rights reserved.           *
*                                                                  *
*              This software is proprietary information and its  *
*              use by unauthorized persons is prohibited.        *
*                                                                  *
****************** < Description End        > ********************
     SKIP1
****************** < Documentation Begin    > ********************
*                                                                  *
* Function   : To provide a common record description for the    *
*              PANAPT batch output message, list records.        *020A*SYW
*                                                                  *
*                                                                  *
* Comments   : This commom area is the output msg rec to almost  *
*         all batch modules. Any change to this layout may cause *
*         the recompilation of most of the PANAPT batch modules. *
*                                                                  *
*             (Put most recent change at front of list)          *020A*SYW
*                                                                  *
* Date     PTF/Vers Programmer  Description                      *020A*SYW
* -------- -------- ----------- -------------------------------- *020A*SYW
* 04/11/96 030A*BAD DeFrang    -Increased output record length   *030A*BAD
*                                from halfword to fullword.       *030A*BAD
* 07/10/94 APT2.0A  S. Wang     delete libcode list and inv list *020A*SYW
*                                records                          *020A*SYW
*                                                                 *020A*SYW
* 05/12/94 APT2.0A  S. Wang     created                          *020A*SYW
****************** < Documentation End      > ********************
     SKIP1
```

```
 01  APTB-MOVREQ-LST-REC.                                        020A*SYW
     05  APTBMR-REC-LEN                PIC 9(05) COMP.          030A*BAD
     05  APTBMR-REC-TYPE               PIC X(02).               020A*SYW
*        A value of '09' indicates this record type.            030A*BAD
     05  APTBMR-KEY                    PIC X(29).               020A*SYW
     05  APTBMR-MOVE-TYPE              PIC X(01).               020A*SYW
     05  APTBMR-STATUS                 PIC X(06).               020A*SYW
     05  APTBMR-FINAL-DATE             PIC X(08).               020A*SYW
     05  APTBMR-ORIGINATOR             PIC X(08).               020A*SYW
     05  APTBMR-DESCRIPTION            PIC X(55).               020A*SYW
     SKIP2
 01  APTB-MSL-REC.                                               020A*SYW
     05  APTBMSL-REC-LEN               PIC 9(05) COMP.          030A*BAD
     05  APTBMSL-REC-TYPE              PIC X(02).               020A*SYW
*        A value of '10' indicates this record type.            030A*BAD
     05  APTBMSL-FROM-NAME             PIC X(10).               020A*SYW
     05  APTBMSL-FROM-DATA             PIC X(08).               020A*SYW
     05  APTBMSL-TO-NAME               PIC X(10).               020A*SYW
     05  APTBMSL-TO-DATA               PIC X(08).               020A*SYW
     05  APTBMSL-LIBSUB                PIC X(07).               020A*SYW
     SKIP2                                                       020A*SYW
 01  APT-MESSAGE-REC.
     05  APTMSG-REC-LEN                PIC 9(05) COMP.          030A*BAD
     05  APTMSG-REC-TYPE               PIC X(02).
 *       A value of '07' indicates this record type.             030A*BAD
     05  APTMSG-MSGS-AREA              PIC X(80).
     SKIP2
```

Appendix D: Utilities

This section contains the following topics:

APTALLOC (Dynamic Allocation Command)

APTALLOC is a Dynamic Allocation Command that is similar to the TSO allocate command. The major difference is APTALLOC has the ability to allocate GDG data sets, which is the reason for its existence. Without the ability to dynamically allocate GDG data sets, REXX Move Procedures would be severely limited. The CA-Panvalet Move Procedure APIR5423 uses APTALLOC.

APTALLOC is a TSO command. It is invoked in the same manner as any other TSO command, such as the ALLOCATE and FREE commands.

Parameters

APTALLOC uses a subset of the TSO ALLOCATE command's parameters. With the exception of the DATASET and VOLREF parameters, the APTALLOC parameters use the same syntax and provide the same function as the parameters of the ALLOCATE command.

The following list shows the parameters supported by the APTALLOC command:

```
Avblock Old
Blksize Position
Catalog Recfm
Cylinders       Release
Dataset Retpd
Delete  Reuse
Expdt   Shr
File    Space
Keep    Tracks
Label   Uncatalog
Like    Unit
Lrecl   Using
Mod     Volume
New     Vseq
```

Parameters File, Dataset, and one of Old, Shr, New, or Mod are required; all others are optional.

The Dataset parameter is slightly different in syntax from the ALLOCATE command. Any Dataset parameter that can be specified by ALLOCATE can also be specified to APTALLOC, except an * (indicating allocation to a terminal) is not supported, and passwords are not used. Like the ALLOCATE command you can also concatenate data sets. In addition, a relative GDG generation can be specified in parentheses following the data set name. A minus generation is specified by entering **M#** preceding the generation, a positive generation is specified by entering a **P#** preceding the generation. For example, to allocate the -1 generation of GDG PAYROLL.BACKUP you would specify the parameter **DATASET('PAYROLL.BACKUP(M#1)')**.

The VOLREF parameter is unique to APTALLOC. It can be used to specify a volume reference to an existing cataloged data set, and would typically be used when you want different data sets to reside on the same tape. It provides the same function as VOL=REF in JCL. The argument for the VOLREF parameter is a data set name.

Return Codes

APTALLOC exits with one of the following return codes:

Return Code	Interpretation	Action
0	Successful completion	File allocated
4	Parameter invalid	
8	Parsing error	A message from the TSO IKJPARSE service should have been issued to indicate what is wrong.
12	Dynamic allocation failure	
16	Internal error	Contact your CA-PanAPT administrator or CA at http://ca.com/support.

Check for Member Existence/Set Condition Code

The purpose of this program is to check if a member of a PDS exists and set a condition code. It can also check the first record of the member to check if it matches a string passed to the program as a parameter.

Activity

There is one input file, a PDS, ddname APTINPUT. The file is opened and checked to see if the requested member exists. The condition code is set based on whether the member exists or whether the first record of the member contains the string passed.

Output/Exit Condition

The return code is set.

Program

APAS5902

Parameters for Program APAS5902

```
'MMMMMMMM,CCNF,CCOK,CCMM,STRING'
```

These parameters are positional and are separated by commas. All condition codes are 1 to 4 positions and must be numeric values.

M

(Required) The name of the member requested. The member name must be 1 to 8 characters.

CCNF

(Optional) The condition code if the member is not found or the file is empty. The default is 4.

CCOK

(Optional) The condition code if all conditions are satisfied (member found and string match). The default is 0.

CCMM

(Optional) The condition code if the member is found and there is a string passed and the string does not match the first record (MisMatch). The default is 8.

STRING

(Optional, but required if CCMM is supplied) The string to be used to compare to the first record of the member if the member is found.

Check for Member Existence/Call Another Program

The purpose of this program is to check if a member of a PDS exists and to call another program.

Activity

There is one input file, a PDS, ddname APTINPUT. The file is opened and checked to see if the requested member exists. If the member exists, it is then checked. If the member is empty or does not exist, a CCNF code is returned; otherwise, the return code from PGMNAME is returned.

Output/Exit Condition

The return code is set.

Program

APAS5903

Parameters for Program APAS5903

```
'MMMMMMMM,CCNF,PGMNAME,pgmparm'
```

These parameters are positional. The first three are required, the last is optional. If any required parameter is omitted, APAS5903 abends with an abend code of 1000. All condition codes are 1 to 4 positions and must be numeric values.

M

(Required) The name of the member requested. The member name must be 1 to 8 characters.

CCNF

(Required) The condition code if the member is not found.

PGMNAME

(Required) The name of the program to be called if the member is found.

pgmparm

(Optional) A variable length parameter string to be passed to the called program. The program name must be 1 to 8 characters in length.

Check for Empty File/Set Condition Code

The purpose of this program is to check if a file is empty and to set the condition code. It can also check the first record of the file to check if it matches a string passed to the program as a parameter.

Activity

There is one input file, ddname APTINPUT. The file is opened and checked to see if it is empty. The condition code is set based on whether the member exists or whether the first record of the member contains the string passed.

Output/Exit Condition

The return code is set.

Program

APCS5902

Parameters for Program APCS5902

```
'CC01,CC02,CC03,STRING'
```

These parameters are positional. If a parameter is specified, all previous parameters must also be specified. If a parameter error occurs, APAS5902 abends with an abend code of 1000. All condition codes are 1 to 4 positions and must be numeric values.

CC01

(Optional) The condition code if the file is empty. The default is 0004.

CC02

(Optional) The condition code if the file is not empty and there is a string and it does not match the first record. The default is 0000.

CC03

(Optional) The condition code if the file is not empty and there is a string and the string does match the first record. (Default: none).

STRING

(Optional, but required if CC03 is supplied) The string to be used to compare to the first record of the file if the file is found.

Check for Empty File/Call Another Program

The purpose of this program is to check if a file is empty and to call another program.

Activity

There is one input file, ddname APTINPUT. The file is opened and checked to see if it is empty. If the file is not empty, the specified program is called.

Output/Exit Condition

The return code is set.

Program

APCS5903

Parameters for Program APCS5903

```
'CCCC,PGMNAME:pgmparms'
```

These parameters are positional. The first two are required. If there are any parameters for the program, they must be preceded by a colon (:). The colon is not part of the parameters to be passed to the program.

If a parameter error occurs, APCS5903 abends with an abend code of 1000.

CCCC

(Required) The condition code if the file is empty. This is a four position numeric field (for example, 0012).

PGMNAME

(Required) The name of the program to be called if the file is not empty.

pgmparm

(Optional) A variable length parameter string to be passed to the called program.

Development Status Posting

The purpose of this program is to post the completion of one of the following development activities:

- Checkout
- Checkin
- Checkin for Migration
- Cancel Checkout
- Cancel Development

It can also be used to verify that the development activity is still outstanding before performing the activity. This prevents an accident if a development job is inadvertently run twice. In concurrent development situations, this program also sends messages to concerned users when concurrent development begins and ends.

Activity

The JCL generated by the models distributed by CA-PanAPT invokes the status posting program with the TEST parameter before starting any work, which ensures that the activity is not restarted when it is already complete. Then upon successful completion, the status posting program is run again to post the activity as complete. This second execution should be run under batch TSO because it might use the TSO SEND command to notify concerned users during concurrent development situations.

Input

Input control statements are read from ddname APTSYSIN, which contains fixed-length, 80-byte records. The following describes the record layout:

Position	Length	Description
1	8	Action being posted or tested. Actions are CHECKOUT, CHECKIN, CHECKINM (Checkin for Migration), CANCKOUT (Cancel Checkout), and CANDEV (Cancel Development).
10	6	Move Request number for member being acted upon.
17	4	Library Code for member
21	3	Library Subcode for member
25	10	Member name
36	8	User ID performing the activity.

Output/Exit Condition

If the conditions are valid, the pending development status is posted complete. If the program fails, a message is reported and a non-zero condition code is returned.

Program

APCS5400

Verification Posting

The purpose of this program is to post the results of a Verification Procedure for a Move Request.

Activity

Verification posting is run to post the success or failure of a Verification Procedure. Comments about the results can also be posted.

A Verification Procedure must include two posts of the Move Request.

1. You must post that the Verification process has started (before doing the verification).
2. You must post the results of the verification.

Input

Input to Verification Posting can be from one of three sources:

- If Verification Posting is called from another program, the input is supplied in a control block passed as a standard OS parameter. The control block is described by the supplied copybook APCCMVRX.
- If Verification Posting is called from JCL, the input can be supplied in the PARM= field of the EXEC statement, or in the VRFYUPD file. There are three required parameters and seven optional parameters, separated by commas.
- If supplied in the PARM= field the sequence and values are:

Description	Status	Length
Verification Category Number	Required	1 to 2
Move Request Number	Required	1 to 6
Verification Status	Required	1
Reason Code	Optional	1 to 4
Comment	Optional	1 to 70
Comment	Optional	1 to 70
Comment	Optional	1 to 70

Description	Status	Length
Comment	Optional	1 to 70
Comment	Optional	1 to 70
Comment	Optional	1 to 70

If input is supplied in the VRFYUPD file, the PARM= field of the EXEC statement must be omitted. If there is any parameter in the PARM= field of the EXEC statement, the VRFYUPD file is ignored. You must use one main record and zero to six comment records. Comments can be anywhere in columns 1-70 of the comment records.

The format of the main record is:

Description	Status	Length
Verification Category Number	Required	2
Move Request Number	Required	6
Verification Status	Required	1
Reason Code	Optional	4

Output/Exit Condition

The Pending file is updated and if all approvals and verifications for this level have been met, the status of the Move Request is updated to Approved for this level.

Program

APCS5330

Pending File Extract

The purpose of this program is to extract Move Request information from the Pending file and to create a flat file that can be used as input to On-Demand modeling.

Activity

The extract program uses the Move Request number supplied in the MOVEREQ= parameter, extracts the Move Request information from the Pending file, and writes it to a flat file that is in the format usable by On-Demand modeling.

Input

Input to the extract program is a Move Request number, indicating where the information is to be extracted from the Pending file. The input can be supplied either in the PARM= field of the EXEC statement of the JCL or in the APTSYSIN file. The format is as follows:

MOVEREQ=*nnnnnn*

nnnn

Specifies a 1- to 6-digit Move Request number where information is to be extracted.

Output/Exit Condition

A flat file of extracted data from the Pending file in a format suitable for input to On-Demand modeling.

Program

APCS5940

Parameters for Program APCS5940

MOVEREQ=*nnnnnn*

nnnn

Specifies a 1- to 6-digit Move Request number where information is to be extracted.

Enable the CA-PanAPT DB2 Option

The purpose of this program is to enable the CA-PanAPT DB2 option.

Activity

The input file is ddname APTDB. It is the CA-PanAPT database.

Output/Exit Condition

The return code is set.

Program

APCS0590

Parameters for Program APCS0590

None.

Disable the CA-PanAPT DB2 Option

The purpose of this program is to disable the CA-PanAPT DB2 option.

Activity

The input file is ddname APTDB. It is the CA-PanAPT database.

Output/Exit Condition

The return code is set.

Program

APCS0591

Parameters for Program APCS0591

None.

Appendix E: Release 1.3 Compatibility Keywords

This section contains the following topics:

About Release 1.3 Compatibility Keywords

Many keywords that existed through version 1.3 of CA-PanAPT became obsolete when system definable Move Levels were added. These keywords were replaced with ones that are generic and not tied to a specific level.

If your system has exactly three move levels, these obsolete keywords are still available for any Models you have that have not been modified to use the new keywords. Your first level is assumed to be the Test level, the second one the QA level, and the last one the Production level. If you have more or less than three levels, then these old keywords cannot be used.

System Keywords

All of the compatibility keywords are non-resettable. A description of each keyword follows:

$BKOTAM

The 1 to 8-character Access Method code for the Library Code Production level Back Out data set.

$BKOTDDN

The 1 to 8-character ddname for the Library Code Production level Back Out data set.

$BKOTDSN

The 1 to 44-character data set name for the Library Code Production level Back Out data set.

$BKOTSEC

The 1 to 10-character security data for the Library Code Production level Back Out data set.

$BKUPAM

The 1 to 2-character Access Method code for the Library Code Production level Backup data set.

$BKUPDDN

The 1 to 8-character ddname for the Library Code Production level Backup data set.

$BKUPDSN

The 1 to 44-character data set name for the Library Code Production level Backup data set.

$BKUPSEC

The 1 to 10-character security data for the Library Code Production level Backup data set.

$MOVETIME

A one-character code indicating the type of move being processed. The values for this keyword are:

Q

Move Modeling move to QA

P

Move Modeling move to Production

B

Move Modeling Production Back Out move

T

Retrieve Modeling copy to Test

O

On-demand Modeling, no movement

V

Verification Modeling, no movement

$OPTBP

The one-character BKUP->PROD Option value from the Library Code indicating whether Back Out processing is supported, Y, or not supported, N. It is now emulated, based on the Library Code Production level Backout Control value. If the Backout Control is B or R, then this keyword is set to Y, otherwise it is set to N.

$OPTMVDEL

The one-character Delete from Test at QA Move Time Option value from the Library Code indicating whether the member in the Test library should be deleted, Y, or should not be deleted, N, after the move to QA is complete.

It is now emulated, based on the Library Code QA level Move Control value. If the Move Control value is M or D, then this keyword is set to Y, otherwise it is set to N.

$OPTPB

The one-character PROD->BKUP Option value from the Library Code indicating whether to copy Production members to Backup at production move time, Y, or not, N.

It is now emulated, based on the Library Code Production level Backup Enabled value. If the Backup Enabled value is Y, so is this keyword, otherwise it is set to N.

$OPTPO

The one-character PROD->BKOT Option value from the Library Code indicating whether to copy Production members to Backout at Back Out move time.

It is now emulated, based on the Library Code Production level Backout Control value. If the Backout Control value is R, this keyword is set to Y, otherwise it is set to N.

$OPTQADEL

The one-character Delete from TEST/QA at PROD Move Time Option value from the Library Code indicating whether the member in the Test or QA library should be deleted, Y, or should not be deleted, N, after the move to Production is complete.

It is now emulated, based on the Library Code Production level Move Control value. If the Move Control value is M (Move) or D (Delete), this keyword is set to Y, otherwise it is set to N.

$OPTTP

The one-character TEST/QA->PROD Option value from the Library Code indicating whether to move members into Production, Y, or not, N.

It is now emulated, based on the Library Code Production level Move Control value. If the Move Control value is M (Move) or C (Copy), this keyword is set to Y, otherwise it is set to N.

$OPTTQ

The one-character TEST->QA Option value from the Library Code indicating whether this Library Code supports moves from Test to QA, Y, or not, N.

It is now emulated, based on the Library Code Quality Assurance level Move Control value. If the Move Control value is M (Move) or C (Copy), this keyword is set to Y, otherwise it is set to N.

$PRODAM

The 1 to 2-character Access Method code for the Library Code Production level Target data set.

$PRODDDN

The 1 to 8-character ddname for the Library Code Production level Target data set.

$PRODDSN

The 1 to 44-character data set name for the Library Code Production level Target data set.

$PRODSEC

The 1 to 10-character security data for the Library Code Production level Target data set.

$QAAM

The 1 to 2-character Access Method code for the Library Code Quality Assurance level Target data set.

$QADDN

The 1 to 8-character ddname for the Library Code Quality Assurance level Target data set.

$QADSN

The 1 to 44-character data set name for the Library Code Quality Assurance level Target data set.

$QAONLY

The one-character QA-Only field value from the Move Request identified by $MR. The value indicates whether this is a QA-Only Move Request, Y, or not, N.

It is now emulated based upon the Early Stop Level for the Move Request. If the Early Stop Level specifies your Quality Assurance level this keyword is set to Y, otherwise it is set to N.

$QASEC

The 1 to 10-character security data for the Library Code Quality Assurance level Target data set.

$TESTAM

The 1 to 2-character Access Method code for the Library Code Test level data set.

$TESTDDN

The 1 to 8-character ddname for the Library Code Test level data set.

$TESTDSN

The 1 to 44-character data set name for the Library Code Test level data set.

$TESTSEC

The 1 to 10-character security data for the Library Code Test level data set.

$VERLEVEL

A one-character value indicating the level for the Verification being modeled. The value can be T for Test verification, Q for QA verification, and P for Production verification. The new keyword to determine this is $VERSHORTNAME, which gives to the short level name at which the Verification is being modeled.

$VERMOVETIME

A one-character value indicating what the $MOVETIME variable will contain on the next tentative move. Note that $MOVETIME is also obsolete. The new keyword with the closest meaning is $DEST1SHORTNAME, which gives you the short level name for the next tentative move.

System Keywords Availability

The table below provides information showing the times or processing phases when the Compatibility system keywords have been initialized and are available. The column keyword entries are:

S

Keyword is available during the Start phase

I

Keyword is available during the Init phase

M

Keyword is available during the Move phase

T

Keyword is available during the Term phase

E

Keyword is available during the End phase

N/A

Keyword does not apply to the Modeling facility

******

(Retrieve) The value comes from the Move Request where the member is assigned, or is blank.

A modeling error results if a model attempts to use a keyword when it is not available.

Keywords	Move	Retrieve	On Demand	Verification
$BKOTAM	IMT	IMT	IMT	IMT
$BKOTDDN	IMT	IMT	IMT	IMT
$BKOTDSN	IMT	IMT	IMT	IMT
$BKOTSEC	IMT	IMT	IMT	IMT
$BKUPAM	IMT	IMT	IMT	IMT

Keywords	Move	Retrieve	On Demand	Verification
$BKUPDDN	IMT	IMT	IMT	IMT
$BKUPDSN	IMT	IMT	IMT	IMT
$BKUPSEC	IMT	IMT	IMT	IMT
$MOVETIME	IMT	IMT	SIMTE	SIMTE
$OPTBP	IMT	IMT	IMT	IMT
$OPTMVDEL	IMT	N/A	IMT	IMT
$OPTPB	IMT	IMT	IMT	IMT
$OPTPO	IMT	IMT	IMT	IMT
$OPTQADEL	IMT	IMT	IMT	IMT
$OPTTP	IMT	IMT	IMT	IMT
$OPTTQ	IMT	IMT	IMT	IMT
$PRODAM	IMT	IMT	IMT	IMT
$PRODDDN	IMT	IMT	IMT	IMT
$PRODDSN	IMT	IMT	IMT	IMT
$PRODSEC	IMT	IMT	IMT	IMT
$QAAM	IMT	IMT	IMT	IMT
$QADDN	IMT	IMT	IMT	IMT
$QADSN	IMT	IMT	IMT	IMT
$QAONLY	M	M **	SIMTE	SIMTE
$QASEC	IMT	IMT	IMT	IMT
$TESTAM	IMT	IMT	IMT	IMT
$TESTDDN	IMT	IMT	IMT	IMT
$TESTDSN	IMT	IMT	IMT	IMT
$TESTSEC	IMT	IMT	IMT	IMT
$VERLEVEL	N/A	N/A	N/A	SIMTE
$VERMOVETIME	N/A	N/A	N/A	SIMTE

Appendix F: Enqueues

This section contains the following topics:

Enqueues Performed on CA-PanAPT VSAM Data Sets

Enqueues on CA-PanAPT VSAM data sets are performed as follows:

- The scope of all enqueues is SYSTEMS.
- The QNAME for all enqueues is APTSYS.
- The RNAME for all enqueues is an 84-byte field, formatted as follows:
  - The first 44 characters of the RNAME field is the data set name of the CA-PanAPT VSAM file that is being accessed.
  - When enqueuing VSAM updates, the 40 characters following the data set name always contain "APTDB" padded with spaces.
  - When enqueuing certain records, Move Requests, or other entities, the 40 characters following the data set name begin with 8 characters describing the record type being updated.

When enqueuing various batch programs, the 40 characters following the data set name contain the batch program name padded with spaces. At this time APCS5391 is the only batch program that is single threaded in this manner.See the table that follows:

Records	First 8 characters	Last 32 characters
Control file	APTCTL	Contain position 2 through 29 of the VSAM key of the Control record, padded with spaces.

Records	First 8 characters	Last 32 characters
Pending file	APTPEND	Contain the 6 character Move Request number padded with spaces. This enqueues on a specific Move Request. Also, at times, there are enqueues with the remaining 32 characters containing 000000 followed by spaces. This enqueues the Pending file control record.
Library Code	APTLIBC	Contain the 4 character Library Code and the 3 character subcode padded with spaces, if a Library Code is being enqueued. Contain "$MODEL BASE: " and the 4 character model base padded with spaces, if a Model Base is being enqueued.
Inventory	APTDIBS	Contain the full VSAM key of the Inventory record to be enqueued, padded with spaces.

Glossary

"Active" Move Request

A *Move Request* that is available for Move Processing. That is, it has not been deleted, and its status is not Moved-QA Only, Moved to Production, or any Backout status.

Activity

A CA PanAPT online function. You can designate which users can perform CA-PanAPT activities using the CA PanAPT security panels.

Activity Key

Describes the CA PanAPT activities such as MOVEASGN/ADD. Users must be authorized to perform the activity. See the section Planning Activity Authorization" in the "Planning the Implementation" chapter for a list of CA-PanAPT activities.

ACTIVITY (activity) Event

The *security event* that is called when a user attempts to process data (presses ENTER) after a function has been accessed, but before the database is updated.

Administrator

(See *System Administrator* or *Group Administrator.)*

Age

The age of a *Move Request*, in terms of days, based on the *Move Request*'s Move Date.

Alternate Index

The VSAM path to the Alternate Index for the CA-PanAPT *Move Request File*, usually referenced through ddname *APTPAIX*.

Approval

Granting formal authorization to process a *Move Request* or acknowledging that notification of the move has been received. Author\-ized users grant approvals online using an ISPF panel, eliminating delays caused by signing printed forms.

APTCTL

This is the last node of the data set name of the *Control File* and is usually the ddname used to reference this file.

APTDIBS

The last node of the data set name of the *Inventory File* and is usually the ddname used to reference this file.

APTHIST

The last node of the data set name of the *History File* and is usually the ddname used to reference this file.

APTLIBC

The last node of the data set name of the *Library Code File* and is usually the ddname used to reference this file.

APTMDLO

The last node of the data set name of the *Modeling Output File* and is usually the ddname used to reference this file.

APTMODEL

The last node of the data set name of the PDS that contains *models* and is usually the ddname used to reference this file.

APTPAIX

The last node of the data set name of the VSAM path to the *Alternate Index*. It is usually the ddname used to reference the Alternate Index.

APTPEND

The last node of the data set name of the *Move Request File* and is usually the ddname used to reference this file.

Assignment

Temporary *ownership* of a Production member while it is being changed as part of a *Move Request.* Only one user can hold assignment for a member.

Assignment, Proper

Proper Assignment refers to the relationship tested when a *Move Request* is closed. The MOVEREQ/CLOASSGN activity defines the relationship and the Close Assignment Option (CLO) defines the user IDs to be tested.

Automatic Assignment

A method of assignment for unassigned members. Under Automatic Assignment, when a user adds an unassigned member to a *Move Request,* the member is *assigned* to the user as part of the operation.

Back Out

Physical movement of a member from a *Backup Library* to a *Production Library* to replace a member that is causing problems in the Production environment. The problem member is moved to a *Back Out Library* as part of this process.

Back Out Library

A library that contains the version of a Production software entity that was *backed out* from a *Production Library*.

Backup Library

A library that contains the most recently replaced version of a Production software *entity.* Entities stored here can be recalled to the Production Library in case the new version causes problems.

CLISTLIB

The CA PanAPT TSO CLIST/REXX Procedure library.

Control File

The file, *APTCTL*, that retains CA PanAPT site-defined data, System *Activity* information and user information.

***DEFAULT**

The user ID that assigns CA-PanAPT activity authorization to CA-PanAPT users who are not explicitly defined to CA-PanAPT.

Development Facility

The CA-PanAPT Development Facility (DF) is a functional extension to CA-PanAPT that provides change management during the development portion of the life cycle before any migration is started. The Development Facility provides an environment to perform functions such as Checkout/Checkin, member text modifications, and member compile/link processing for members that are associated with a defined *Project*.

Disapproval

Revoking formal authorization to process a *Move Request* or an acknowledging that notification of the move has been reviewed. When a Move Request is disapproved, only the user ID who disapproved an approval category or the CA-PanAPT System Administrator can *approve* or *unapproved* it. Authorized users revoke approvals online using an ISPF panel.

Dual Maintenance

While converting non-critical parts of CA-PanAPT, you may continue to use the old CA-PanAPT version. However, the data that is updated on your old system must also be updated on the new CA-PanAPT system to ensure that the new system is consistent with the old system.

Enable

Activating a CA PanAPT procedure or method. For example, *Auto Assign\-ment is enabled* means that CA-PanAPT is using an Automatic procedure for *assigning* members to users.

Entity

Any logical association of information that can be moved. This can include programs, JCL, files, or documentation.

Group

A logical association of user IDs used to specify authorizations more easily. Groups are identified when user IDs are defined.

Group Administrator

A specially designated CA PanAPT user who may be allowed special author\-izations to perform CA-PanAPT activities.

History File

The *APTHIST File*, which contains *Move Requests* that have been purged from the *Pending File*.

INIT (Initialization) Event

The *security event* activated when a user enters CA-PanAPT.

Inventory Approval

A formal acknowledgment that an *Inventory Record* meets local site stan\-dards. CA PanAPT provides many fields for optional, *site-specific,* informa\-tion about a member. CA PanAPT does not validate these fields when it creates the record or later. The Inventory Approval assures that the indi\-vidual *Inventory Record* has been reviewed and meets site standards.

Inventory File

The file, which retains an inventory of any or all members of *Production Libraries.* Also referred to as the *APTDIBS File*.

Inventory Qualifier

An eight character code used to link multiple *Library Codes* that use the same *Production Library.* You specify an Inventory Qualifier as part of the Library Code definition when several Library Codes share the same Produc\-tion libraries.

Inventory Record

A record on the *Inventory File* that contains member-specific information about a Production member. Inventory Assignment uses this information to control member assignment. *Model* processing uses this information to control generation of JCL or control statements.

ISPMLIB

The CA PanAPT ISPF message library.

ISPPLIB

The CA PanAPT ISPF panel library.

ISPSLIB

The CA PanAPT ISPF skeleton library.

ISPTLIB

The CA PanAPT ISPF table input library.

JCLLIB

The CA PanAPT JCL library.

Keywords

Representation of data element values used in *Modeling.* Keyword values can be used to control logic in a *Model.*

Levels

(See Library Levels)

Library Code

A collection of information that defines member types and the data set names of the libraries where they reside. You must define at least one Library Code for each set of *Test, QA, Production, Backup*, and *Back Out Libraries* you use. You may choose to have more than one Library Code for a set of libraries (see Inventory Qualifier). The Library Code is most often used to describe the complete xxxx/yyy pattern used to distinguish data sets and processing. It is also used to differentiate the xxxx part of the pattern xxxx/yyy from the Library Subcode yyy.

Library Code File

The file which contains a record for each *Library Code* set up under CA PanAPT. Also referred to as *APTLIBC*.

Library Levels

Different classifications of libraries in the CA-PanAPT system. CA-PanAPT has five library levels: *Test, Quality Assurance* (optional), *Production, Backup* (optional), and *Back Out* (optional).

Library Subcode

An optional qualifier that adds a further description to a *Library Code.* The subcode is the yyy part of the pattern xxxx/yyy.

LOADLIB

The CA PanAPT load module library.

Model

A set of control and data statements that controls generation of JCL and/or control statements required to move a member. Models are stored as members of a CA-PanAPT-supplied PDS. They are prepared and changed with a standard ISPF editor.

Model File

The standard PDS file which retains all CA-PanAPT *models.* Also referred to as *APTMODEL*.

Modeling

A CA PanAPT feature that generates 80-byte images, usually JCL or utility control cards. Modeling provides the flexibility necessary to move members of different library types according to site standards.

Modeling Output File

Contains the output from the modeling process. The members of this PDS are recreated each time the modeling operation is executed.

Move Processing Cycle

The batch portion of the *Move Request Cycle* that entails actually proces\-sing the *Move Requests*.

Move Request

A unit of work to CA PanAPT. A Move Request specifies the members that CA-PanAPT moves in one logical operation. It also specifies details about the move, such as the Move Date.

Move Request Cycle

A combination of batch and online procedures that takes a *Move Request* from creation to completion.

Move Request Status

The current state of a *Move Request* at any point during the entire CA-PanAPT *Move Request Cycle.*

Operations

An attribute that denotes users responsible for the daily operation of CA-PanAPT activities. It may be specified for any user ID.

Ownership

The concept that a CA PanAPT *entity* (a *Move Request* or an *Inventory Record)* explicitly belongs to a CA-PanAPT user.

PARMLIB

The library that contains control statements (parameters) used by other parts of the system. Also referred to as the PARM library.

Pending File

The file that contains *Move Request* information. Also referred to as *APTPEND*.

Pending File Alternate Index (APTPAIX)

A (VSAM) alternate path to the *Pending File* that contains *Move Request* information.

Permanent Owner

A CA PanAPT user ID who is responsible for a Production member when it is not assigned to a *Move Request.*

PROCLIB

The CA PanAPT cataloged procedure library.

Production Library

A library that contains the current version of a software *entity* used in a company's day-to-day operations.

Production Turnover

The movement of software *entities* into the *Production Libraries*. The entities can be executable programs, documentation for reference, source code, or any other entity a site considers current.

Project

A Project definition creates a CA-PanAPT system record that associates a special group of libraries, called "Development Libraries" with a group of members defined in one or more Move (Change) Requests.

Quality Assurance Library

A library where pre-Production testing and staging of a software *entity* takes place.

Reassign/Release Processing

Processing that determines whether a member should be released or assigned to a different Move Request.

Release

Giving up *assignment* of a Production member.

Retrieve

Physical movement of a member from a *Production*, *Quality Assurance*, or *Back Out Library* to a *Test Library* prior to modification.

Security Events

Logical points in the CA PanAPT environment where a user-written *security exit program* can perform tests. There are three types of security events (see *INIT Event*, *ACTIVITY Event*, and *TERM Event*).

Security Exit Program

A user-written program that controls security processing. It performs tests that permit or restrict a user from performing a CA PanAPT *activity*. If active, it complements CA PanAPT authorization.

Site-specific

Refers to any standard, methodology, or convention not recognized by every other CA-PanAPT (MVS) data processing site. Job and data set naming conventions are some of the more obvious *site-specific* standards. CA-PanAPT sites can also use the optional fields in inventory and *Library Code* definitions for their own purposes. These are also site-specific standards.

Special Handling Move Request

A specially designated *Move Request* in which CA-PanAPT does not prepare a batch job or move any *entities.* Special Handling Move Requests never have members and *approvals* are not allowed. They are outside the normal *Move Request Cycle.* This is used to document an event external to CA PanAPT.

Status

(See Move Request Status.)

System Administrator

A specially designated CA PanAPT user who can always use any CA-PanAPT *activity.* There can be more than one CA PanAPT System Administrator.

System Keywords

A type of *keyword* used in *modeling* that starts with a $ and is initialized by CA PanAPT.

TERM (Termination) Event

The *security event* in which the user exits CA-PanAPT. The user-written *security exit program* can be modified to perform your own termination routine.

Test Library

A library where a new version of a software *entity* is developed.

Transfer

Reassignment of the *assignment* of a member to another CA PanAPT user ID.

Turnover

(See Production Turnover.)

Unapproval

The removal of formal authorization to process a *Move Request*. This indicates that notification of the move has not been reviewed by an author\-ized user. When a Move Request is unapproved for an *approval* category, it is as if the Move Request has never had approval granted. Authorized users can remove approvals or *disapprovals* online using an ISPF panel.

User Keywords

A type of *keyword* used in *modeling* that is defined on the first *model* statement on which it is encountered.

Verification

Confirmation of the completeness, readiness, or validity of a *Move Request* and its contents given by an analysis procedure or program.

Verification Procedure

A program or set of programs designed to analyze the attributes of a Move Request and post a pass or fail indication.

Verification Procedure Category

A number from 01 to 20 identifying a specific *Verification Procedure* as defined in the *Control File*. Each category has a description, and associated model control statements.

Verification Requirements

A set of *Verification Procedures* which must be satisfied for a particular level (Test, QA, or Production) for any Move Request with a member from a Library Code which requires the *Verification*.

Index

$

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V