ROUTINE MANAGER

ADDING AND MODIFYING ROUTINES

(Release 717)

November 16, 1992

FINAL

Information Systems and Computing
Office of the President
University of California
ROUTINE MANAGER
ADDING AND MODIFYING ROUTINES

Step 1 - Choosing a Module Name

When adding new routines to the system which will be called by the Routine Manager, the first step is to select the proper Module Name. The Routine Module name, which is assigned in the format PPEXNNN, is broken down into three functional components:

- **PPE** - Constant for all Routines called by Routine Manager (excluding indirect calls)
- **X** - Indicates the primary function of the routine
  - C - Consistency Edits
  - I - Implied Maintenance
  - G - Gross-to-Net Edits
  - A - Action Edits
  - M - Monthly Maintenance
  - Q - Quarterly Maintenance
  - Y - Yearly Maintenance
  - F - Fiscal Maintenance
- **NNN** - Routine Number

Reserved for Level of Processing:
- 001-039 - Appointment Level Processing (Functions 'C' and 'I' Only)
- 040-049 - Campus Appointment Level Processing (Functions 'C' and 'I' Only)
- 050 - Appointment Summary Processing (Functions 'C' and 'I' Only)
- 051-089 - Distribution Level Processing (Functions 'C' and 'I' Only)
- 090-099 - Campus Distribution Level Processing (Functions 'C' and 'I' Only)
- 100 - Distribution Summary Processing (Functions 'C' and 'I' Only)
- 101-299 - DET Triggered / Maintenance (Guideline Only)
- 300-700 - Conditional / Unconditional Maintenance (Guideline Only)
- 701-799 - DET Triggered Set Trans Maintenance
ROUTINE MANAGER
ADDING AND MODIFYING ROUTINES

NNN - Routine Number (Cont.)

800-990  Reserved for Campus Use

991-999  Reserved for System Use

These numbering schemes are only used as a guideline for establishing naming standards. With the exception of Appointment and Distribution Level processing (001-100), deviation from these guidelines will not require any Routine Manager program modifications.

Step 2 - Adding the Routine to the Routine Definition Table

The primary purpose of the Routine Definition Table (RTD) is to associate a Routine Identifier (referenced by the Processing Group Table) with an executable load member name (Program Id). The Routine Identifier consists of both Routine Type (i.e. C,I,A,M) and Routine Number. The Routine Type assigned on the Routine Definition Table does NOT have to match the type resident within the Module Name. For example, let's say we are adding a new Consistency Edit Routine which is being executed from both 08 and 12. In this case we may choose to add Routine Definition Table entries which are consistent with each of the calling functions but reference the same module name;

<table>
<thead>
<tr>
<th>Routine Definition</th>
<th>Routine Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>C 150</td>
<td>PPEC150</td>
</tr>
<tr>
<td>I 150</td>
<td>PPEC150</td>
</tr>
</tbody>
</table>

Step 3 - Routine Development

When Developing New Routines it is best to start by selecting a "clone routine" which best fits the functional needs required by the new routine. For example, If we were adding a new Consistency Edit which is to contain edits pertaining to Job Appointment level processing, we would use program PPEC003 as a base model. When adding new Distribution level consistency edit routines we should use an existing Distribution level routine such as PPEC053. When adding a new "Non Appointment" related consistency edit routine, we should use PPEC101 as a basis.

After selecting the routine to "clone", the "clone" routine should be copied to the newly selected routine name. The next step is to delete all of the unwanted logic which will be replaced with new logic for the new routine.
ROUTINE MANAGER
ADDING AND MODIFYING Routines

All routine specific logic will be located within the 1000-MAINLINE-ROUTINE Section. The following is a list of all Sections which will be common to all routines and should NOT be deleted if present in the clone;

0000-DRIVER
0100-INITIALIZATION
9050-AUDITING-RESPONSIBILITIES
9060-APPT-DIST-LOC-CALCULATION
9300-DATE-CONVERSION-DB2

All initialization processes which are related directly to the routine being added should be placed in the Initialization Section of that routine, NOT in the "Application Driver". The first time the routine is called (only the first time), the Initialization Section will be performed.

The following Working Storage copy members should be included in all Routines;

01 DL-FIELD-WORK-AREA COPY CPWSXDLS.
01 DATE-CONVERSION-WORK-AREAS. COPY CPWSXDC2.
01 XWHC-COMPILE-WORK-AREA. COPY CPWSXWHC.
01 COLUMN-88-WORK. COPY CPWS88S.

The following Working Storage EXTERNALS should be included in all Routines;

01 DATA-ELEMENT-CHANGE-INDICATORS EXTERNAL COPY CPWSXDEC.
01 XCSS-COMMON-SYSTEM-SWITCHES EXTERNAL COPY CPWSSWCH.
01 XDTS-COMMON-SYSTEM-DATES EXTERNAL COPY CPWSXDT.
01 ECES-CON-EDIT-SWITCH EXTERNAL COPY CPWSEC.
01 EGES-GTN-EDIT-SWITCH EXTERNAL COPY CPWSEGES.
01 EIMS-IMPL-MAINT-SWITCH EXTERNAL COPY CPWEIM.
01 EAXD-APPT-ARRAY-SWITCH EXTERNAL COPY CPWSEX.
01 XDER-DATA-ELMT-RTN-POINTERS EXTERNAL COPY CPWSXDER.
01 XDRT-DETRoutine-TRIGGERS EXTERNAL COPY CPWSEX.
01 EDB-MAINTENANCE-PASS-AREA EXTERNAL COPY CPWSEMP.

When referencing Data Element Numbers within the application logic, Working Storage defined variables (Ennnn) should be used. All elements should be defined to the "ELEMENTS-USED" "01" level within Working Storage. This group data name should only contain element definitions which are referenced within the source. Any unreferenced elements (remnant from the clone) should be deleted. The same rules will be applied for referenced messages.
The "MESSAGES-USED" "01" level will be used to define all referenced system messages. Only referenced system messages should be resident, unreferenced elements should be deleted.

Various system switches, flags, and values which are referenced in multiple routines (require passing through an EXTERNAL) have been defined within the copy member CPWSSWCH. When adding new variables to this copy member, the procedure division equivalent CPPDSWCH will also require modification. This member initializes all variables defined to copy member CPWSSWCH each time a new employee record is processed.

The following system level dates are defined within copy member CPWSXDTS. This copy member is passed as an EXTERNAL to all routines;

- ISO-ZERO-DATE
- DATE-TODAY-WITH-CC
- DELETE-CUTOFF-ISO-DATE
- FIRST-OF-MONTH-DATE
- END-OF-MONTH-DATE
- 1-MO-PRIOR-DATE
- 2-MO-PRIOR-DATE
- 13-MO-PRIOR-DATE
- APPT-BEGIN-YRMO
- APPT-End-YRMO
- WSP-FISCAL-YR-CCYY

The member CPWSXDTS is populated during the initialization process within the "Application Driver" through copy members CPPDFDTS and CPPDXTS.

If there is a need for System Parameter Values within the new routine, the copy member CPWSXSP2 should be included as an EXTERNAL within the External Section of the routine. All System Parameter Values are loaded during the Initialization processes of PPP080, PPP120, and PPP130. The Procedure Division copy member CPPDXSP2 has been developed to load the System Parameter Table defined within copy member CPWSXSP2.

All posting of XDEC-ARRAY entries for the purpose of recording element value changes will be performed by moving the Working Storage defined element number (Ennn) to the 'DL-FIELD' variable and performing the 9050-AUDITING-RESPONSIBILITIES paragraph. Explicit setting of the XDEC-ARRAY (i.e. MOVE 1 TO DATA-ELEMENT-CHANGE-INDICATORS (0346)) must NOT be done as the Auditing Responsibilities function will insure that the proper implicit trigger switch setting occurs for both implicit and explicit maintenance.
Routine Manager
Adding and Modifying Routines

Step 4 - Update the Data Element / GTN Tables

When adding routines which will be triggered for execution based on explicit or implicit maintenance, the Data Element and/or GTN tables will need to be updated. On the Data Element table there are three types of trigger settings which can be established; Consistency Edits, GTN Edits, and Implied Maintenance. For the purpose of self-documentation, the chosen trigger number should, whenever possible, match the last three digits of the new routine name. Any trigger type can be used to trigger any type routine, but whatever trigger type and number combination is chosen it should match the execution triggers identified on the Processing Group Table entry for that routine.

Note: The establishment of triggers alone on the DET will NOT cause a routine to be executed. The DET routine triggers only identify data element relationships for the purpose of interrogation by Routine Manager. Each unique trigger (type and number) defines a unique relationship between sets of data elements. It is the responsibility of the Processing Group Table (PGT) to define which routine trigger(s) will cause the routine to execute. Without the execution triggers being established on the PGT, the Routine Manager will NOT interrogate the switch arrays to determine if any one of the triggers (defined to the DET) have been set ON. In simple terms, we define the element relationships to the DET in the form of trigger type and number. On the PGT we define which of these relationships (triggers) will be checked to determine routine execution.

When establishing element relationships for GTN Edits where both GTN and Data Elements are related, both the GTN and Data Element Tables need to be updated. In this case the trigger number should match (for documentation purposes) the GTN number being edited. This number should match the trigger defined to the related Data Element(s) on the DET. In some cases, such as Bonds, there are multiple GTN elements defined for a single function. In these cases there is generally one GTN Edit routine which will edit all bond deductions. In this case, even though the GTN numbers are different, all BOND GTN entries will have the same Edit Routine Number assigned on the GTN Table. If a given Consistency Edit routine is to be executed from both USER08 and USER12 and both are to be triggered on the same Data Elements, only one trigger type should be established: Conedit.
ROUTINE MANAGER
ADDING AND MODIFYING ROUTINES

Step 5 - Update the Processing Group Table

For each application (08,12,13) a series of "Application Checkpoints" have been defined (See EDB Re-Engineering Design). For each defined "Application Checkpoint" the previously existing logic has been replaced with calls to Routine Manager (PRTNMGR). Each call to Routine Manager has been assigned a unique Processing Group Number. Each time Routine Manager is called from the "Application Driver", the Processing Group for that call is passed through the Routine Manager Interface (CPLNKRM1) to identify to Routine Manager which Processing Group to execute.

For each Processing Group a series of routines are defined for execution. These Routines are identified by Routine Type and Routine Number (used to access the Routine Definition Table), and processing sequence. For each Routine defined within the Processing Group, execution triggers must be established. The following is a list of available execution triggers:

**U999 - Unconditional Execution**

**Cnnn - Execute only if Consistency Edit trigger 'nnn' is set ON (1) - CPWSECES**

**Innn - Execute only if Implied Maintenance trigger 'nnn' is set ON (1) - CPWSEIMS**

**Gnnn - Execute only if GTN Consistency Edit trigger 'nnn' is set ON (1) - CPWSEGES**

**A0nn - Execute only if Personnel Action 'nn' has been set ON (Y) - CPWSXACN**

When modifying existing routines, there is no need to update the Processing Group Table, since the routine has already been defined to an existing processing group. When adding new routines, the current PGT should be analyzed to determine the proper location to insert the new routine. Each Processing Group will contain definitions for routine execution. Each routine entry will be selected by Sequence Number. It is very important that the proper sequence number be assigned. Since the routines will be executed in "sequence", an improper sequence number may cause illogical execution of the intended routine.
NOTE: When Adding NEW Data Elements

A new feature has been added to maintain the XDEC-ARRAY for Data Elements being updated during Periodic Maintenance. For each Data Base Row, a subroutine has been developed to check the before and after values of each data element defined to the row. Those elements which have been changed are updated on the XDEC-ARRAY to indicate that maintenance has occurred. When adding new elements to existing rows which are maintained by Periodic Maintenance, the corresponding XDEC Maintenance subroutine should be updated to reflect any new elements added. The following is a list of XDEC Maintenance Subroutines;

PPAXDCHG
PPBELCHG
PPBENCHG
PPCMPCHG
PPEARCHG
PPFADCHG
PPLOFCHG
PPPAYCHG
PPPCCCHG
PPPCMCCHG
PPPERCHG

It should be noted that these subroutines are only called during the Periodic Maintenance Process (PPP130).