Service Request 13202
Release 1138
Generic Full Accounting Unit
Overview
What is the Generic FAU project?

Campuses are moving towards a more diverse mix of financial applications. The key financial reference point within the Payroll Payroll/Personnel System (PPS) is the Full Accounting Unit (FAU). In order to interact with the diverse financial applications, it was necessary to “genericize” the FAU and its related application logic within the PPS.

Generic FAU

Generic Functions

The current Base FAU is twenty-three characters long, broken into six fields. These individual fields are used singly, partially in different combinations, and wholly, to perform various functions within the PPS. Functions using parts of the FAU include sorts, report breaks and accumulations, display/data entry of the FAU, retrieval of related data, e.g. department code, etc.

Some of the functions seem straightforward in essence. Displaying an unformatted FAU is reasonably easy to make generic. But what if only 23 spaces exist on the report? Displaying a formatted FAU, e.g. with separating hyphens, becomes slightly more difficult. Since the FAU is generic, then hyphenation could range from one 30 character field with no hyphens, to a 59 character field with each of the single FAU characters separated by its own hyphen. The latter is not probable, but shows that “generic” solutions sometimes involve concrete compromises that seem “reasonable”.

Some functions are less straightforward. Many places in the PPS accessed copymember fund numbers and fund ranges to determine data definition or data grouping. One data definition example is whether an FAU is Federal or State funds. In this instance, the concrete Base fund number itself, by application definition, implies an attribute of funding source. In another financial application, fund number per se might not exist and the attribute might be defined in a completely different manner. Or, it is possible that another application might decide that the attribute is completely unnecessary to that system. In this example, making the PPS generic did not simply mean providing for a different way to move around work fields. It meant identifying the purpose of the question being asked, and then designing a generic answer to that question that serves all “probable” financial systems.

Generic FAU

What is meant by “generic FAU”?

In the Base PPS, the FAU consists of Location, Account, Cost Center, Fund, Project Code and Sub. One campus might not use the Cost Center and Project Code and thus could get by using only thirteen characters. Another campus might need a new field, and thus need a larger FAU. Or the length of any of the FAU subunits might require a character more or less. Whatever the specific need, the FAU exists concretely in various objects such as DB2 tables, EDB update or pay transactions, program work fields and reports. It is not feasible to attempt a completely flexible solution, i.e. one in which a campus could choose any length FAU from zero to infinity. A choice was made to define a fixed length generic FAU of 30 characters, and to provide functionality for this 30-character field throughout the PPS.
**Above the Line and Below the Line**

Base code was rewritten to process an FAU structured in the current Location, Account, Cost Center, Fund, Project Code, Sub structure. However, Base code has been divided into “above-the-line” and “below-the-line” objects.

Above-the-line objects can contain references to a 30 character FAU, i.e. the generic FAU. They cannot refer to any FAU subunit.

Below-the-line objects can refer to FAU subunit fields, and can perform functions based on the subunits.

When an above-the-line object requires FAU subunit functionality, it must access a below-the-line object, e.g. via a subroutine call, to perform that function. The above-the-line object can perform functions based on the returned information, but that returned information must be generic.

An above-the-line and below-the-line example can be seen in program PPINAFP which is called in the Base system to access the Account/Fund Profile for various purposes, including obtaining a department code based on a passed Location and Account. Common code in many Base programs moved a distribution location and account into the PPINAFP linkage, and performed a direct call to PPINAFP. Upon return, data in the linkage was queried to determine the success of the search for a department code and action was taken accordingly. Obviously the Base Account/Fund Profile itself was FAU structure specific in that it contains a six-character account number on the Account Record. Furthermore, the source code moving a specific FAU subunit, the account number, to a linkage field was also obviously FAU structure specific. The design issue was how retain as much above the line as possible, and rewrite as little code as possible.

The entire calling program and PPINAFP could have been defined as below-the-line.

PPINAFP and its linkage could have been changed to accept a generic FAU and the calling program changed to move the generic FAU to the linkage. The calling program would be above-the-line; PPINAFP would be below the line.

The calling program could have been changed to pass a generic FAU to a new below-the-line module, which in turn would call an unchanged PPINAFP. The calling program would be above the line; PPINAFP and the new module would be below the line.

The calling program could have been changed to perform a paragraph contained in a new below-the-line procedure copymember which would be copied into the calling program. The paragraph could perform either of the two previous examples. The calling program would be above the line; PPINAFP and the new copymember would be below the line.

COBOL source, Copymembers, DB2 Tables, Indexes and Views, and Includes, Maps and the related Maplib members and BMS code can all be either above-the-line or below-the-line. Any campus using the Base FAU structure will be able to use both Base above-the-line and below-the-line objects, with the usual local modifications as needed for other issues. Campuses not using the Base FAU structure will need to modify the below-the-line objects. The amount of modification, of course, depends on how different the local FAU structure and functional definition is.
It is intended that campuses not using the Base FAU structure should be able to use, as is, all above-the-line objects (again, with local modifications for other issues). However, it is probable that in some complex situations in which a generic solution was difficult, that some local modification of an above-the-line object might be the best solution for a local need.

**Regression**

For campuses using the Base FAU structure, and Base over and below-the-line objects, the results of PPS processing should be the same after installing the generic FAU changes as before. Thus, regression testing using the pre and post-release code should produce the same results, with the caveat of unavoidable report format changes.

However, just because it should work as before does not mean that there is not a great deal of work to install it. The sheer number of changed objects is daunting. Unchanged COBOL members requiring recompilation add to the task. There are also the simple but tedious and time consuming tasks such as changing forms and local transaction keying procedures/instructions.

It will be possible to isolate parts for testing, especially if you have separate batch and online test environments. It will be difficult to install partially in Production. If certain online subsystems such as History or Rush Checks are not currently installed, then some installation can be deferred, or at least the testing can be deferred. But, for example, once the EDB Distribution table PPPDIS has been changed, a serious amount of work must be done in coordination.

**Local planning for regression testing cannot start too soon.**

**Guidelines**

Some guidelines were used to affect decision making; none have been adhered to completely in all cases.

The primary guideline was to remove all FAU subunit specific code from all Base above-the-line objects.

The second guideline was to define and isolate the below-the-line objects in such a way that campuses moving away from the Base FAU structure could easily replace Base below-the-line objects with their own without having to change above-the-line objects.

The third guideline was to minimize the impact on the campuses. This may seem, in light of the number of objects that were changed, to have been a futile goal. However, in most cases, the actual changes within above-the-line objects were minimal. And initially, the new below-the-line objects could be installed as is.

In some cases, minimizing the change for the user maximized the changes for the technical staff. An example can be seen in online programs that allow data entry of subunits of the FAU in separate screen fields, such as PPWEAPP. The simplest technical solution was to treat the FAU as a single field, but this would have affected data entry users. The decision was made to retain, where it existed, separate
input fields. This means the user will see the same product, but the technical solution required much more coding.

An example of where this guideline to minimize the change was put in abeyance is with the establishment of a single FAU field on transactions, tables and files. It would have had less impact on the Base PPS to add the additional seven bytes for the generic FAU as separate fields and columns on records and tables, and simply leave everything else unaltered. A below-the-line process could have been designed to construct the FAU from the isolated parts where necessary. This was not deemed good design, nor did it seem to meet the intent of the project. Therefore many objects, such as transaction structures, have been changed significantly.

There were some more concrete guidelines.

Screens containing FAU’s are all considered below-the-line. The source code for the screen processing program was changed to exclude FAU subunit references. A copycode include provides the method for local changes to FAU manipulation and display. But no attempt was made to redesign SDF panel code.

Reports containing FAU’s are primarily considered below-the-line. In cases where the unformatted FAU is displayed, and provision for 30 characters could be easily made within the existing report format, then such changes were made and the program remained above-the-line. In cases where a formatted FAU is displayed, and provision for 40 characters could be easily made within the existing report format, then such changes were made and the program remained above-the-line. In cases where the FAU, formatted or unformatted, could not be introduced without significant report redesign, no change was made. There is some ambiguity here as to whether the resulting program should be considered above or below-the-line. From a Base standpoint it will be considered above-the-line, with local changes as required.

**Objects**

This section identifies the types of objects that were changed. Basically, a very large part of the PPS was affected.

COBOL source: **many** were change. There were also many one-time table/file conversion programs, and new permanent below-the-line programs.

Copymembers: all types of copymembers were changed. Many table and record working storage copymembers (CPWS*) were changed, and a few were added to define the Base FAU. A few file definition copymembers (CPFD*) needed to be changed for record length. Some procedure copycode (CPPD*) were changed, and many were added to handle FAU specific processing for screen processing programs. Several new linkage copymembers (CPLNF*) were added for new below-the-line called modules.

Recompiles: every unchanged program containing a changed linkage copymember, External copymember or Include, or SELECT * INTO logic for a changed table needed to be recompiled. There were many.
Binds: as appropriate, every recompiled program had to be rebound either as a package and/or for a plan.

VSAM table structures: several VSAM CTL records were changed. One example is the Work Study table. In most cases this required a one-time conversion program, a Copymember change, and a change to PPP010 and PPP851.

DB2 table structures: this required changes to Table and, in some cases multiple, Views and Include members. In a few cases the Index was also changed. And for a few there were Copymember changes as well. For EDB tables the fetch complex programs had to be changed. For most other tables, utility programs had to be changed. One-time conversion programs were needed for each table. For example, the EDB Distribution table row was redefined to contain a 30 character generic FAU. This involved changes to DDL members TBDIS00C, PPPVDIS1, PPPVDIS2 and PPPVZDIS; Include members PPPVDIS1, PPPVDIS2 and PPPVZDIS; Copymembers CPLNKAP2, CPLNKDS2 and CPWSRDIS; and fetch complex programs PPAXDCHG, PPAXDFET, PPAXDUPD, PPAXDUTL and PPDISUTW.

Transactions: this required changes to Copymembers which define the transaction, either in its original input form or which carry the data forward in transaction form. In the case of EDB File Maintenance transactions, the Data Element Table had to be updated to reflect the new single FAU data element, its transaction location and the changed location of any other transaction fields. For the example of the EDB Distribution transaction, this required 864 transactions just to delete or redefine the current FAU subunit data elements.

Forms: for each transaction there is at least one form, and often more, which needed to reflect the FAU data, its location, and the changed location of any other transaction fields. Released forms only reflect the Base FAU fields.

Table data: some tables needed to be updated, specifically any that contained FAU data element specific entries. The most important was the Data Element Table (PPPDET) as mentioned above. But the PPPDES and PPPHDE also required changes. The PPPRDT positions the FAU on the Personnel Action Form and required changes. Depending on local needs, the PPPBUH might require changes. And System Messages will also be changed and added.

Maps: some maps contain data element specific names. These were changed to refer to the single FAU data element. Maplib and BMS code needed to be regenerated, and the maps assembled. Released maps only reflect the Base FAU fields.

CICS Helptext: both Data Dictionary based Helptext and the generic field and screen level Helptext will be changed. Some reanchoring will be required. This did not occur in Phase I.

JCL: file lengths will change, so appropriate JCL needed to be changed. For example, the Pay Transactions increased in length, so anything that creates such a file needed to be changed. This is primarily a campus task. JCL will also have to be created for the one-time processes.

RDO: Resource Definition had to be performed for any new modules used online.
Identifying Below-the-Line Objects

Some naming standards have been established, but a complete methodology for identifying below-the-line objects has not yet been established. The desire to minimize changes in some cases leads to retaining current names, even though the name does not identify the object as below-the-line. Not all objects are equally easy to “mark.”

New below-the-line subroutines were all named PPFAUxxx, and their linkage copymember were called CPLNFxxx. However, some existing routines, such PPINAFP, retained their current name but are below-the-line.

Procedure copycode members in screen processor programs that handle FAU subunit logic were named CPPDxxxx, where xxxx is the function. For example, program PPWEAPP uses a copymember called CPPDEAPP. This name, therefore, does not contain an explicit FAU relationship.

Other copymembers were created for the Base FAU. An example is CPWSFAUR which is a working storage FAU structure definition. In such cases, “FAU” will most likely appear in the name, but this may not always be the case.

SDF panel members are below-the-line but retained their current names. For example, PPWEAPP still uses map PPEAPP0.

Whole tables might be considered below-the-line but retained their current names, as did their related objects. An example is the PPPTIM table.

There are Base messages that refer to FAU subunits. The first attempt was to make the text generic, where that still served to clearly state the message intent. Where the specific subunit reference needed to be retained, a message with the naming format of AUxxx was established.