LOUISIANA DEPT. OF HEALTH & HOSPITALS

Medicaid Eligibility Data System

External Design Part 5: Common Facilities

Table of Contents

Document Information

Created for:	Louisiana Dept. of Health & Hospitals
Project Name:	Medicaid Eligibility Data System
Project Abbreviation:	MEDS
Document Title:	External Design Part 5: Common Facilities
Document Subject:	
Revision Number:	73 (incremented each time the document is saved)
Status:	Final
Last Saved:	Dec 16, 2011 03:06:00 PM
Printed:	Dec 16, 2011 03:06:00 PM
Comments:	
Document Template:	RedManeSpec_W2000.dot

Revision Summary

The following revisions have been made to the document since it was first published:

Date	Description of Change	By
4/29/2003	Revised with RedMane template	M. Smutko
07/23/2003	Misdirected Workflow SIR779	D. DeLaurentis
4/21/2004	Add globals – SIR1017	Lizette Nel
6/01/2004	Add default priority to reason code.	David von Bargen
8/29/2005	Add Workflow Reason Code PF1 Help – SIR 1116	P. Crowley
10/05/2009	Add JCL substitute variables – SIR 1161	K. Powers
3/31/2011	Documentation cleanup – SIR1701	Jacobus Badenhorst

Common Facilities Overview

The Common Facilities modules are the components of the system which are available for use to all processes. The following is a list of common facilities described in this document:

- Security Subsystem
- Workflow Tracking Subsystem
- Batch Submission Subsystem
- Snapshot Number Processing
- Run Control Processing
- Run Control/Snapshot Processing
- Next Number Processing
- Random Number Generator
- The Error Handler
- System Code Translators

Security Subsystem

Overview

The security subsystem has the following objectives:

- To restrict users from performing transactions that they are not authorized to perform.
- To provide each transaction with information necessary to implement “security by action” and “field level security”.

In order to accomplish these objectives, the following functionality is provided. For further information refer to the individual transactions within this document.

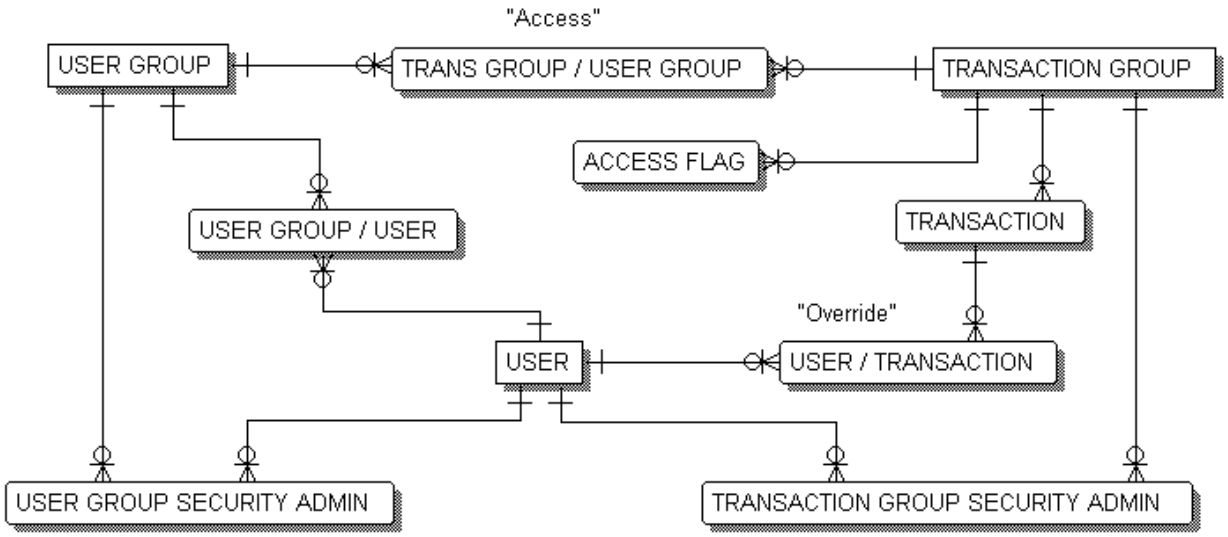
- All users are defined within the security subsystem. They are identified uniquely by their user id.
- Users are associated with logical categories called user groups. All users within a user group perform similar functions and therefore can have their access controlled at a group level. For example, users that are workers could be grouped into a “MEDS worker” user group. A user can be part of many user groups. For example, if a user performs both a worker function and a supervisor function, the user can be in the “MEDS worker” and the “MEDS supervisor” user groups.
- The user group is maintained by security administrators. A user group security administrator can be responsible for one or more user groups and each user group can have as many user group security administrators as required.
- A transaction can be an on-line or a batch process. Since a number of transactions are normally associated with a particular function, the security subsystem allows these transactions to be related through a transaction group. For example, the transactions used to maintain the various reference table can be grouped into a “table maintenance” transaction group.
- The transaction groups are maintained by security administrators. The transaction group security administrator can be responsible for one or more transaction groups. Each transaction group can have as many transaction group security administrators as required.
- The definition of what a user is or is not allowed to do is accomplished by linking user groups with transaction groups. When linking a transaction group with a user group the “allowable actions” are defined. Typical actions are add, modify, display and purge. This allows a “LAMI worker” user group to be linked to the “application maintenance” transaction group with an “allowable action” of display, while the “MEDS worker” user group is linked with “allowable actions” of add, modify, display and purge. Remember that the MEDS worker users can be further segregated into several user groups to allow a finer level of security definition.
- Further refinement of security can be done through the use of the “override function”. This allows a specific user to be linked with a specific transaction. The linking process can either further restrict the user or allow additional functions to be added.
- In general, if a user group has access to a transaction group the users can access all the fields in the transaction group's transactions. However, there may be a need to further secure an individual field. This can be accomplished by defining an access flag for the transaction group. It can be used to control the displaying of information or to control the values that a user is allowed to enter.

All additions, changes and deletions in the security subsystem are written to an audit trail file. The audit trail file can be viewed in entry sequence or in date and time sequence.

The security subsystem has two parts:

- The maintenance portion of the subsystem provides functions to update and inquire on related information. (Described above).
- The control portion of the security subsystem accesses this security information whenever a user wishes to perform a transaction, and determines whether or not the user may execute the transaction. (Incorporated into each program).

Entity Diagram



Description Of The Entity Diagram

Transactions are categorized into **transaction groups**. For example, category code file maintenance and type case file maintenance could be categorized into a “Table File maintenance” transaction group.

Users belong to **user groups**. For example, all workers may be in one user group, and all users who use the system for inquiry purposes may be in another user group. Clearly, one user can belong to more than one user group. The classification of users into user groups is depicted in the entity model as **user group / user**.

One or more users are appointed as **user group security administrators** for each user group. The user group security administrators specify which users belong to the user group.

One or more users are appointed as **transaction group security administrators** for each transaction group. The transaction group security administrators specify which user groups may **access** the transaction groups. Each access record contains a user group id, a transaction group id, a list of actions which the user group may perform when executing transactions in the transaction group, and access flags that allow each transaction in the group to further limit the access capability of users in the user group.

When a user group is granted access to a transaction group, all the users in the group are permitted to perform all the transactions in the transaction group. If there are any exceptions to this rule, the transaction group security administrator can **override** the rule. The override record contains the identification of the user for whom the exception is being made, the transaction that is being given to (or taken away from) the user, a list of actions that the user may perform when executing the transaction, and access flags that allows the transaction to further limit the user's access capability.

Whenever a user wishes to perform a transaction, the security control routine is invoked. This routine checks that the user has permission to execute the transaction.

The user may execute the transaction if:

- any of the user groups to which the user belongs has access to the transaction's group and the user does not have an override that prevents access to the transaction
- or the user has an override that specifies that he may execute the transaction
- or use of the transaction is unrestricted.

The security control routine also determines the actions that the user may perform when he executes the transaction, and the transaction specific parameters (if any) that are applicable.

If the user is not allowed to execute the transaction, an error message is displayed, and the user is returned to the transaction from which the control routine was invoked.

If the user is allowed to execute the transaction, the control routine passes control to that transaction. When the transaction is invoked, the actions that the user may perform, and the parameters from the access or override record are made available to the program.

Menus

Main Menu

```
     TRAN                        SECURITY SUBSYSTEM              *DATE
     PROGRAM                          MAIN MENU                  *TIME

                Id Tran     Description
                -- -------- --------------------------------
                 1 SCMSTM   MASTER DATA MENU
                 2 SCINQM   INQUIRY MENU
                 3 SCATEM   AUDIT TRAIL BY ENTRY MENU
                 4 SCATTM   AUDIT TRAIL BY TIME MENU









ID/*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12
      Help  Main  Retn  Quit
```

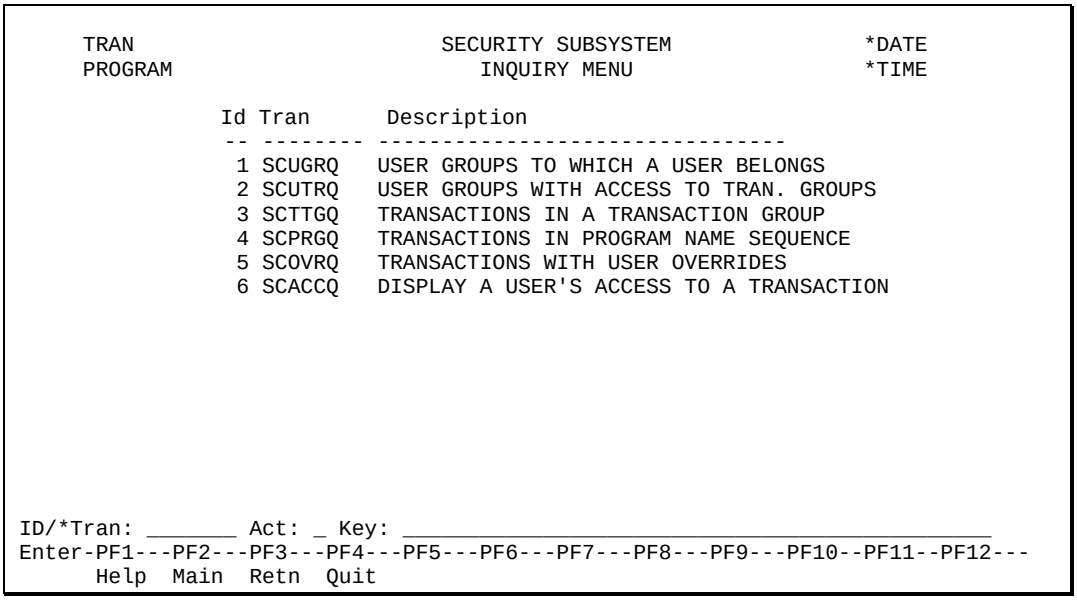
There are four sub-menus available in the security system.

- Master Data Menu. The master data menu is used to invoke transactions that maintain the entities in the system.
- Inquiry Menu. The inquiry menu is used to invoke inquiries that display information about security relationships.
- Audit Trail By Entry Menu. The audit trail by entry menu is used to invoke inquiries into the audit trail information. All of the inquiries display information about when and who added, changed and deleted information. This information is output in time within key sequence. One inquiry exists for every entity in the system.
- Audit Trail By Time Menu. The audit trail by time menu is used to invoke inquiries into the audit trail information. All of the inquiries display information about when and who added, changed and deleted information. This information is output in time sequence. One inquiry exists for every entity in the system.

Master Data Menu

```
     TRAN                          SECURITY SUBSYSTEM                *DATE
     PROGRAM                        MASTER DATA MENU                 *TIME

                ID TRAN     DESCRIPTION
                -- -------- --------------------------------
                 1  SCUSRF  USER
                 2  SCUSGF  USER GROUP
                 3  SCUGAF  USER GROUP SECURITY ADMINISTRATOR
                 4  SCUSUF  USERS IN A USER GROUP
                 5  SCTRGF  TRANSACTION GROUP
                 6  SCTRAF  TRANSACTION GROUP SECURITY ADMINISTRATOR
                 7  SCTRNF  TRANSACTION
                 8  SCFLGF  TRANSACTION GROUP ACCESS FLAG
                 9  SCACCF  USER GROUP ACCESS TO TRANSACTION GROUPS
                10  SCOVRF  USER OVERRIDES FOR A TRANSACTION




ID/*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit
```

Inquiry Menu



Audit Trail Menus

```
     TRAN                          SECURITY SUBSYSTEM                *DATE
     PROGRAM                    AUDIT TRAIL BY ENTRY MENU            *TIME

           Id Tran     Description
           -- -------- --------------------------------
            1 SCUSRQE  USER
            2 SCUSGQE  USER GROUP
            3 SCUGAQE  USER GROUP SECURITY ADMINISTRATOR
            4 SCTRNQE  TRANSACTION
            5 SCTRGQE  TRANSACTION GROUP
            6 SCTRAQE  TRANSACTION GROUP SECURITY ADMINISTRATOR
            7 SCFLGQE  ACCESS FLAG DEFINITION
            8 SCUSUQE  USERS IN A USER GROUP
            9 SCACCQE  USER GROUP ACCESS TO TRANSACTION GROUPS
           10 SCOVRQE  USER OVERRIDES FOR A TRANSACTION




ID/*Tran: _______ Act: _ Key: ____________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit
```

```
     TRAN                          SECURITY SUBSYSTEM                *DATE
     PROGRAM                    AUDIT TRAIL BY TIME MENU             *TIME

           Id Tran     Description
           -- -------- --------------------------------
            1 SCUSRQT  USER
            2 SCUSGQT  USER GROUP
            3 SCUGAQT  USER GROUP SECURITY ADMINISTRATOR
            4 SCTRNQT  TRANSACTION
            5 SCTRGQT  TRANSACTION GROUP
            6 SCTRAQT  TRANSACTION GROUP SECURITY ADMINISTRATOR
            7 SCFLGQT  ACCESS FLAG DEFINITION
            8 SCUSUQT  USERS IN A USER GROUP
            9 SCACCQT  USER GROUP ACCESS TO TRANSACTION GROUPS
           10 SCOVRQT  USER OVERRIDES FOR A TRANSACTION




ID/*Tran: _______ Act: _ Key: ____________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit
```

Maintenance Transactions

User

```
     TRAN                          SECURITY SUBSYSTEM              *DATE
     PROGRAM                        USER MAINTENANCE               *TIME

*Action           : _ (A,B,C,D,M,N,P)
*USER ID          : ________
 Last Name        : ____________________
 First Name       : ________________
*Main Menu        : _______
*Printer          : ________









*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  UGRP  OVRD              Wrkru
```

Every user who will execute transactions must be identified to the security subsystem on this screen. The user's CICS sign on id will be the unique identifier.

The Main Menu is the transaction id of the menu that will be displayed to the user when she/he enters the MEDS system. In addition, this is the menu that will be invoked when the user presses PF4 (main) under certain situations.

The fields on the maintenance screen are described below:

Action (A1)
The action codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

USER ID (A8)
The unique code that is used to identify the user.

Last Name (A20)
The last name of the user.

First Name (A15)
The first name of the user.

Main Menu (A7)
The transaction name.

Printer (A8)
Default printer. The user's reports will be printed at this printer for all report requests. Usually the user's local printer address.

Additional PF Keys

PF Key 5 - UGRP
Invokes the transaction that maintains the users in a user group.

PF Key 6 - OVRD
Invokes the users that have override access to a transaction maintenance.

PF Key 9 – Wrkru
Invokes the Worker Browse by User Id function.

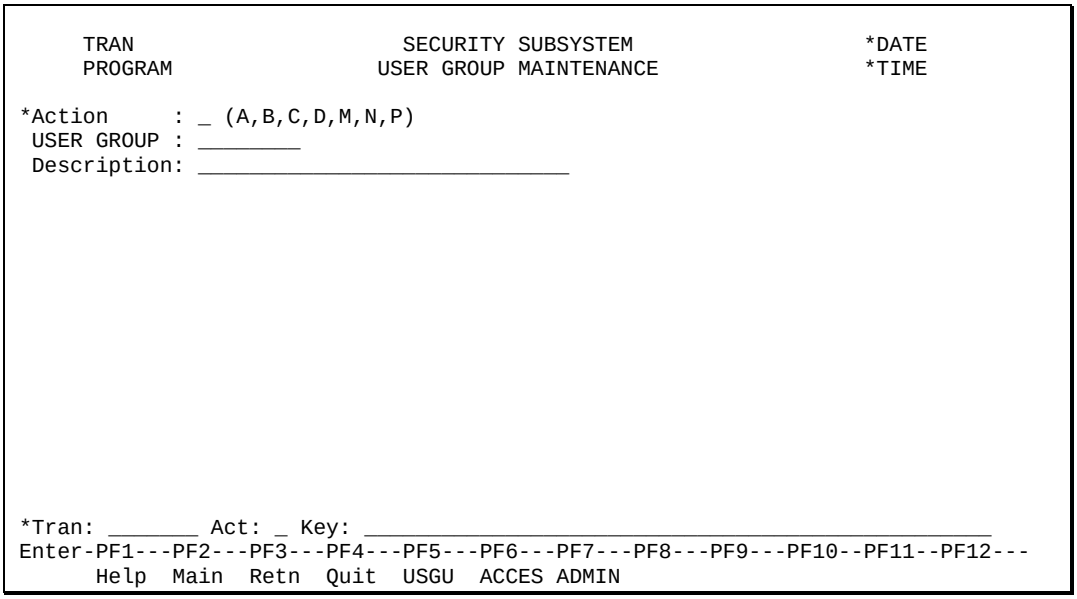
Processing

Whenever an update is performed against your own user record, your security buffer will be refreshed.

The following will be deleted when the user is deleted:

- All transaction override records for the user.
- All transaction group security administrator records for the user.
- All user group security administrator records for the user.
- All user group/user records for the user.

User Group



A user group is a generic id under which users may be grouped. This transaction is used to identify and maintain the group id's. Another transaction (invoked by pressing PF5) is used to actually link users to the group.

The fields on the maintenance screen are described below:

Action (A1)
The action codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

USER GROUP (A8)
The unique code which is used to identify the user group.

Description (A30)
The description of the user group.

Additional PF Keys

PF Key 5 - USGU
Invokes the transaction that maintains the users in a user group.

PF Key 6 - ACCES
Invokes the user group access to transaction group transaction.

PF Key 7 - ADMIN
Invokes the user group security administrator transaction.

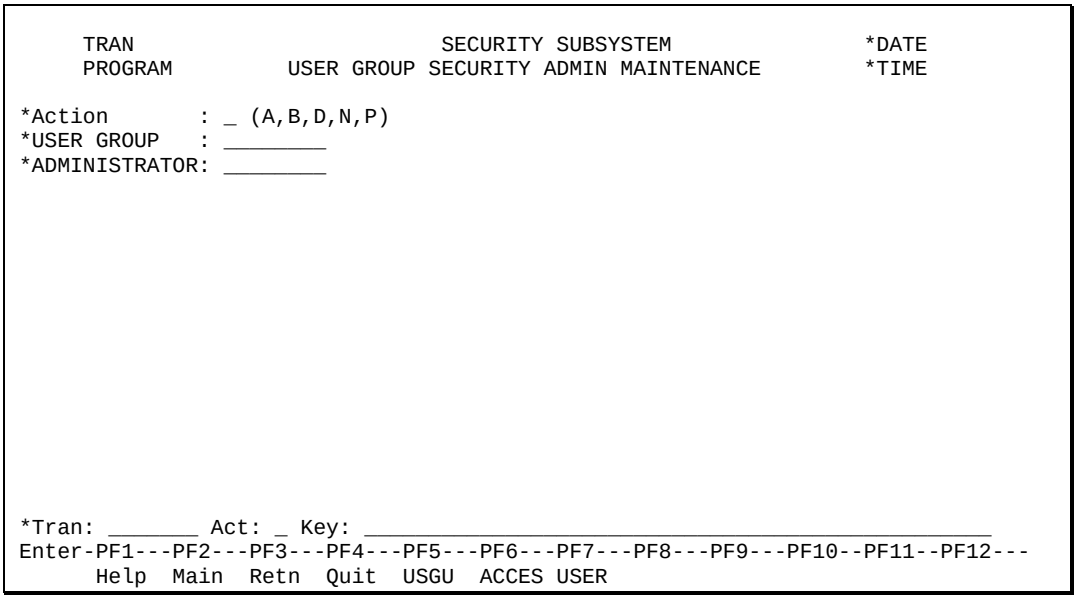
Processing

The following will be deleted when the user group is deleted:

- All user group/user records associated with the deleted group.
- All user group security administrator records associated with the deleted group.

- All user group/transaction group records associated with the deleted group.

User Group Security Administrator



Security administrators of a given user group are the only users who may link users to the group (the actual linkage of users to user groups is done using the user group/user transaction - PF6). This transaction is used to define the user id's of the security administrator(s) for a user group.

Any given user group may have many security administrators. The list action on this screen may be used to display all of the users that have been defined as security administrators for the group.

Be aware, until a security administrator is defined for the user group, no linkage of users to the group may be done.

The fields on the maintenance screen are described below :

Action (A1)
The action codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

USER GROUP (A8)
The unique code which is used to identify the user group for which an administrator is being appointed. The user group's description will be displayed adjacent to the field.

ADMINISTRATOR (A8)
The unique code used to identify the user who is being appointed as the user group's security administrator. Be aware that a user group can have an infinite number of administrators. The user's name will be displayed adjacent to the field.

Additional PF Keys

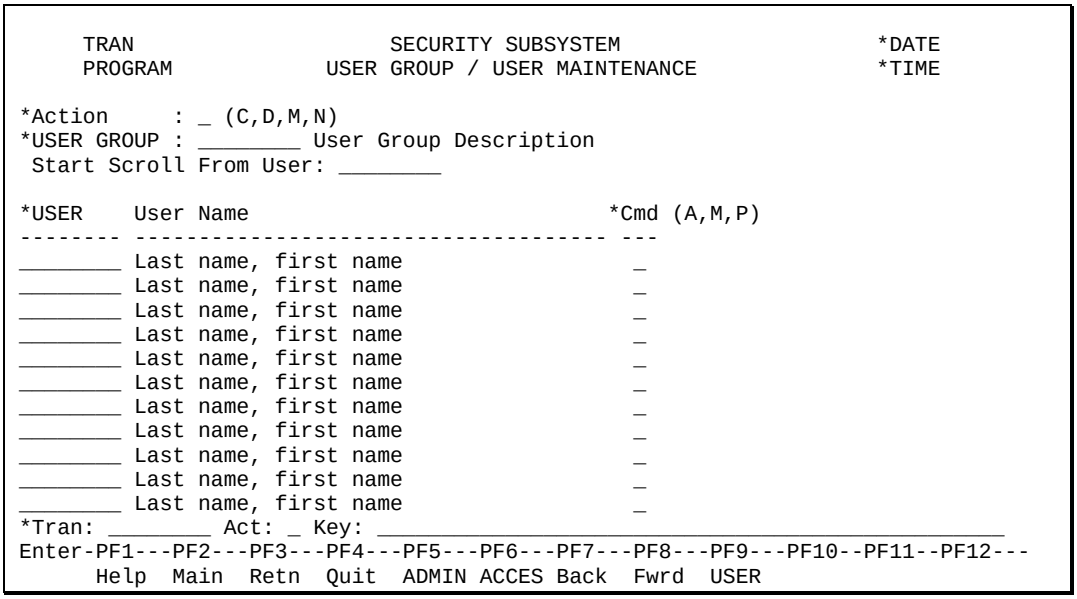
PF Key 5 - USGU
Invokes the transaction that maintains the users in a user group.

PF Key 6 - ACCES
Invokes the user group access to transaction group transaction.

PF Key 7 - USER
Invokes the transaction that maintains users.

Processing

User Group/User



This transaction is used to link users to user groups. An explanation as to why one does this should be offered. Typically, all users within a user group have the same security access level (i.e., they can all access the same transactions and execute the same action codes). Having a user group allows one to define the security requirements for the group of users rather than for each individual, thus saving time and effort. This means that when a new user is defined to the system, typically they need only be linked to an existing group; this action causes the user to immediately inherit all of the abilities and limitations already defined for the group.

This is multi-line transaction that may be used to inquire upon AND maintain the list of users in a group. Please be aware, only security administrators of the group may actually change the list of users in a user group.

There is no limit to the number of users that can belong to a user group and a user can belong to an unlimited number of user groups. When a user belongs to multiple groups, he/she is assumed to have the access of all of the groups.

The fields on the maintenance screen are described below :

Action (A1)

The action codes for this field are (C)lear, (D)isplay, (M)odify, (N)ext.

USER GROUP (A8)

The unique code which is used to identify the user group to which users are being linked. The user group's description will be displayed adjacent to the field.

Start Scroll From User (A8)

The value entered in this field is the starting point for the alphabetic listing of the users shown under the “USER” heading.

USER (A8)

This column contains the identity of each of the users in the user group. Each user id entered must exist on the user file. The user's name will be displayed adjacent to the field.

Cmd (A1)
This column contains the instruction that specifies the action to be performed on the corresponding user information. See the System Wide Facilities for how this field is used.

Additional PF Keys

PF Key 5 - ADMIN
Invokes the user group security administrator transaction.

PF Key 6 - ACCES
Invokes the user group access to transaction group transaction.

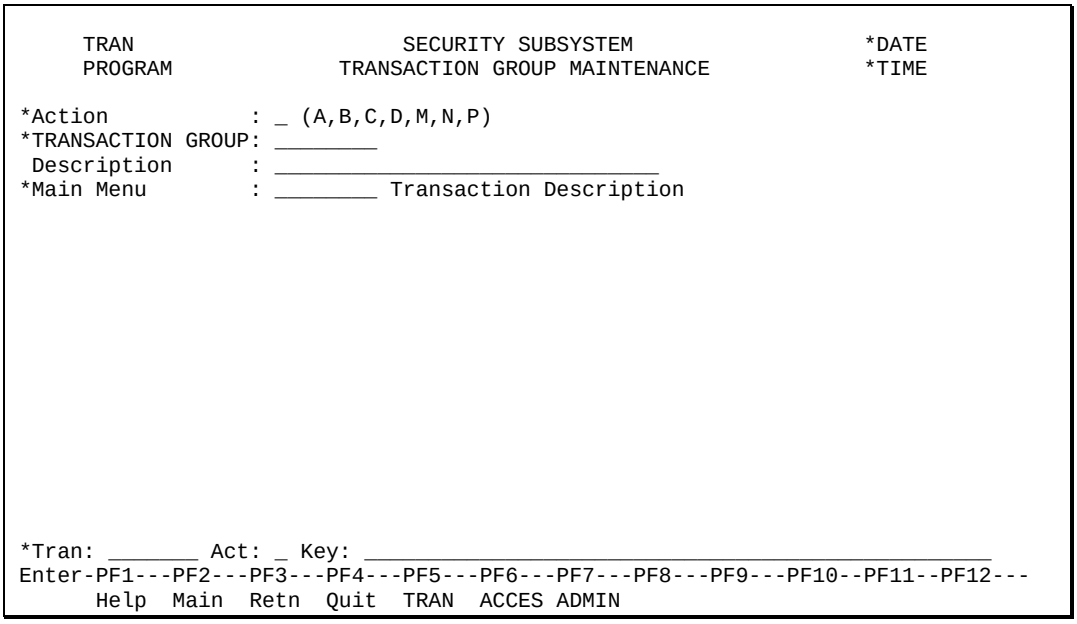
PF Key 9 - USER
Invokes the transaction that maintains users.

Processing

Whenever you are added or deleted from a user group, your security buffer will be deleted.

The administrators of a user group are the only users who may change the user group/user relationship.

Transaction Group



A transaction group is a generic id under which transactions will be grouped. For example, every inquiry in the MEDS system could be categorized under a transaction called “Inquiries”.

This transaction is used to identify and maintain transaction group id's. A different transaction (invoked by pressing PF6) is used to actually link the transactions to the group.

The fields on the maintenance screen are described below :

Action (A1)
The action codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

TRANSACTION GROUP (A8)
The unique code which is used to identify the transaction group.

Description (A30)
The description of the transaction group.

Main Menu (A8)
The id of the menu that will be invoked if PF4 (main) is pressed from a transaction belonging to the transaction group. If entered, the transaction must exist on the transaction file. The transaction's description will be displayed adjacent to the field. This is an optional field.

Additional PF Keys

PF Key 5 - TRAN
Invokes the transaction maintenance program.

PF Key 6 - ACCES
Invokes the user group access to transaction group transaction.

PF Key 7 - ADMIN
Invokes the transaction group security administrator transaction.

Processing

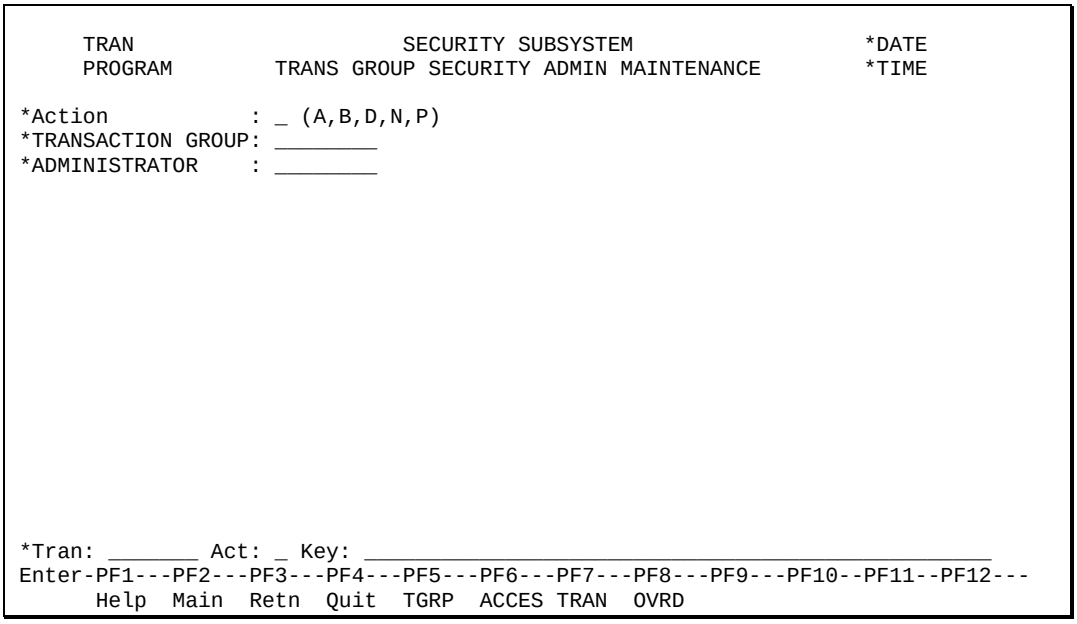
Referential integrity:

- A transaction group may not be deleted if any transaction refers to it.

The following will be deleted when the transaction group is deleted:

- All transaction group security administrator records associated with the transaction group.
- All user group/transaction group records associated with the deleted group.
- All access flags linked to the deleted group.

Transaction Group Security Administrator



Transaction group security administrators are the users that are responsible for defining who can access which transactions and the level of security the users have in the transaction (i.e., they hold the key to the system's security). The actual definition of the access relationship is performed in the “access maintenance” transaction, invoked by pressing PF7. This transaction is used to simply define the user id's of the security administrator(s) for a transaction group.

Specifically, the transaction group security administrator performs the following functions:

- Defines which user groups may access the transaction group(s) that he/she administers. This is done in the access maintenance transaction - PF7.
- Defines which users have override access to transactions that belong to transaction groups that he/she administers. This is done in the override maintenance transaction - PF10. This function is typically done when:
  - A specific user needs access to a limited number of transactions within a group.
  - A specific user is prohibited from accessing a limited number of transactions in a transaction group.
- Defines the “access flags” used by a transaction group. Please be aware that this is a very rare occurrence. Please see the access flag maintenance transaction for more information about this.

Any given transaction group may have many security administrators. The list action on this screen may be used to display all of the users that have been defined as security administrators for the group.

Be aware, until a security administrator is defined for the transaction group, no linkage of user groups may be done, thus preventing any access to transactions in the group.

The fields on the maintenance screen are described below :

Action (A1)

The action codes for this field are (A)dd, (B)rowse, , (D)isplay, , (N)ext, (P)urge.

TRANSACTION GROUP (A8)
The unique identity which is used to identify the transaction group for which an administrator is being appointed. The transaction group's description will be displayed adjacent to the field.

ADMINISTRATOR (A8)
The unique code used to identify the user who is being appointed as the transaction group's security administrator. Be aware that a transaction group can have an infinite number of administrators.

Additional PF Keys

PF Key 5 - TGRP
Invokes the transaction group maintenance transaction.

PF Key 6 - ACCES
Invokes the user group access to transaction group transaction.

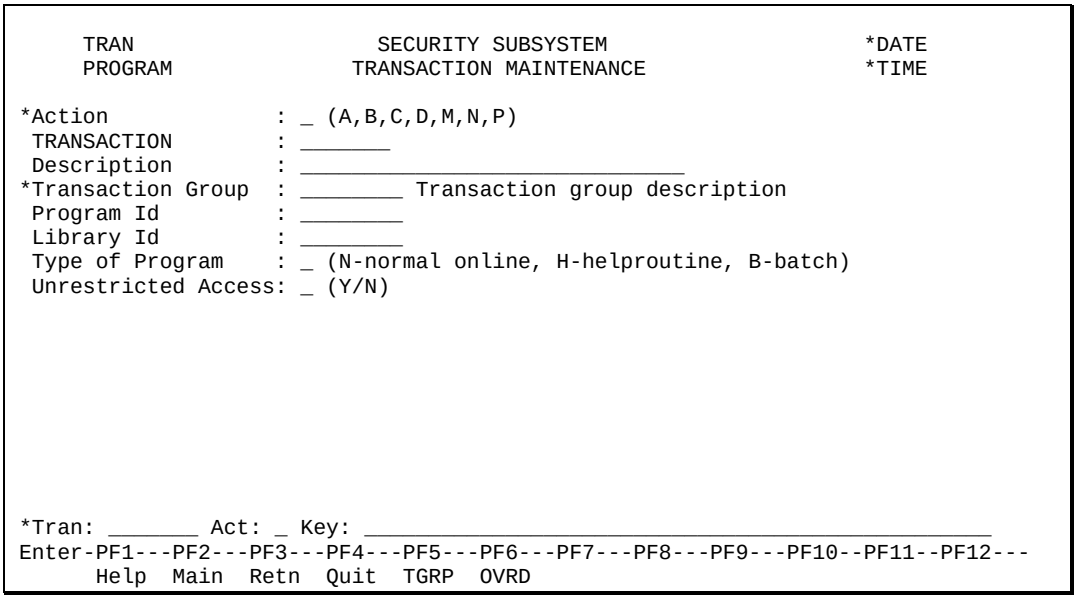
PF Key 7 - TRAN
Invokes the transaction that maintains transactions.

PF Key 8 - OVRD
Invokes the transaction that maintains user/transaction overrides.

Processing

The administrator name will be derived from information on the user file. Please see the common routines section for how to access this information.

Transaction



Every transaction in the system will be identified using this screen. This includes normal on-line programs, help routines and batch reports. Before a transaction can be identified one must know its Natural program id, the Natural library in which the program resides and the transaction group to which the transaction belongs. The transaction group defined on this screen is the sole group to which the transaction belongs. Remember, when a user group has access to a transaction group, they have access to all of the transactions in the group (unless there is an override).

The value of the transaction id is the program name that is executed.

The fields on the maintenance screen are described below:

Action (A1)
: The action codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

TRANSACTION (A8)
: The unique code that is used to identify the transaction. The unique identity of the program that will be invoked when this transaction is requested.

Description (A30)
: The description of the transaction.

Transaction Group (A8)
: The transaction group to which the transaction belongs. The entered value must exist on the transaction group file. The description of the transaction group will be displayed adjacent to the field.

Program Id (A8)
: The unique identity of the program that will be invoked when this transaction is requested.

Library Id (A8)
: The name of the NATURAL library that contains the program specified in the program id field, above.

Type of Program (A1)
Code indicating the type of program. This field is used to prevent the access of transactions that are not accessible through the on-line system. Valid values are : N,H,B (normal on-line, helproutine, batch).

Unrestricted Access (A1)
Used to indicate whether any user may access the transaction. If set to 'Y', all users will be allowed to execute the transaction, except those with an override user / transaction record that disallows access. Valid values are 'Y' and 'N'.

Additional PF Keys

PF Key 5 - TGRP
Invokes the transaction group maintenance transaction.

PF Key 6 - OVRD
Invokes the users that have override access to a transaction maintenance.

Processing

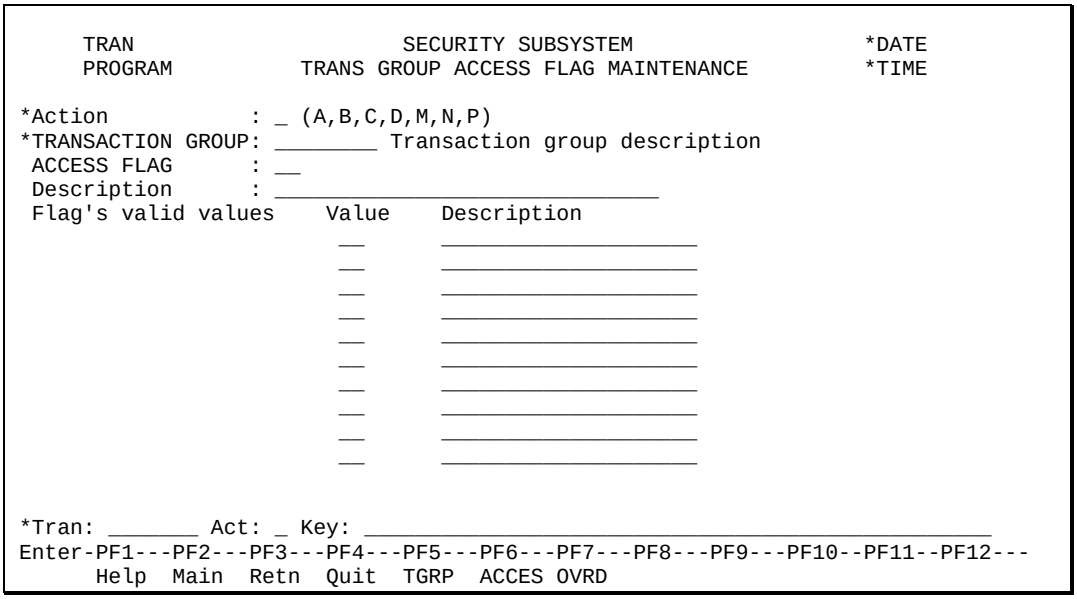
Referential integrity:

- A transaction may not be deleted if it is indicated as being a main menu for a user or a transaction group.
- A transaction may not be deleted if it is indicated as being a hot key for a user.

The following will be deleted when the transaction is deleted:

- All transaction override records for the transaction.

Transaction Group Access Flag



An access flag is a means of controlling security on individual fields within a transaction group. For example, an access flag could be used to:

- Control data displayed on a transaction according to the user's level of security. For example, let's say there exists a transaction that displays all information about a case including the application date. You could use an access flag to restrict when the application date was displayed on the screen.
- Limit the values a user can enter on the transaction. For example, let's say there exists a transaction with fields that are maintained by two distinct user groups. You could use an access to define who can update which fields.

Please be aware that installing security of this nature requires more than just the entry of an access flag. One must also program the specific functionality into the system. But this should not be a big concern because this type of security is an infrequent requirement.

Each transaction group can have up to 10 different access flags. Each access flag can contain one of 10 different values, each of which has some pre-defined meaning. This maintenance transaction allows a description to be entered for each access flag within a transaction group and it also allows the definition of valid values for the specified access flag. Each valid value defined is required to have a description.

The fields on the maintenance screen are described below :

Action (A1)
: The action codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

TRANSACTION GROUP (A8)
: The unique identity of the transaction group to which the access flag is linked. The transaction group's description will be displayed adjacent to the field.

ACCESS FLAG (N2)
: The unique identity of the access flag being defined. Currently, the system caters for a maximum of 10 access flags per transaction group therefore this field must be a number between 1 and 10.

Description (A30)
The description of the access flag.

Valid values
This is a group of fields which represent the valid values that may be specified against an access flag on the user group/transaction group (access) and the user/transaction (override) maintenance transactions. Max occurrences : 10, Typical occur: 2.

Value (N2)
A valid value for the specified access flag. If value is specified, the description must be entered.

Description (A20)
The description of what the value signifies. If a description is entered, a non-zero value must be specified.

Additional PF Keys

PF Key 5 - TGRP
Invokes the transaction group maintenance transaction.

PF Key 6 - ACCES
Invokes the user group access to transaction group transaction.

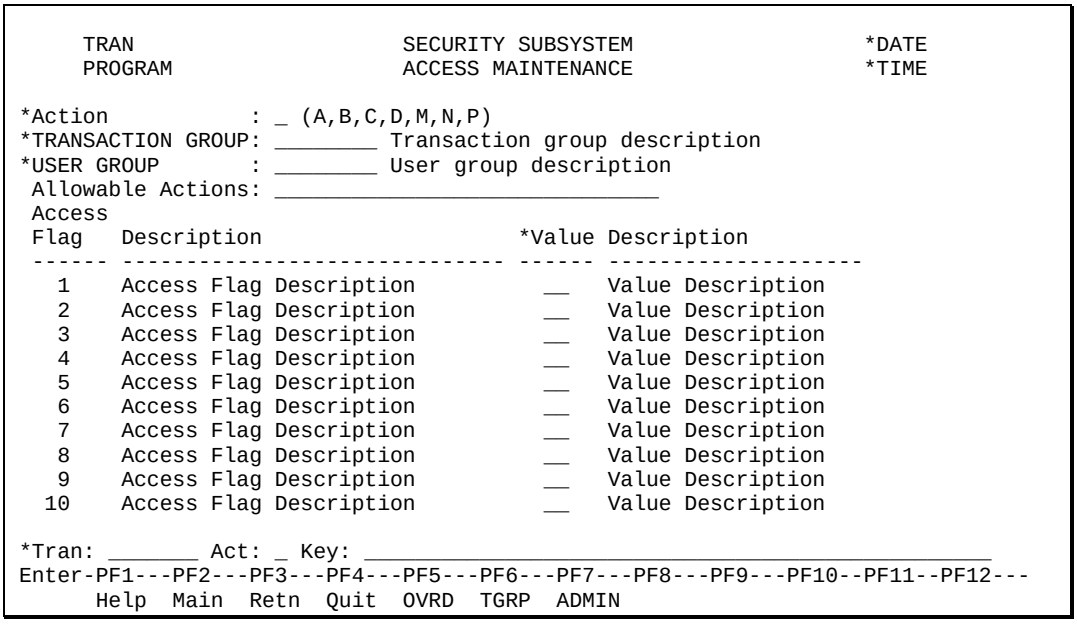
PF Key 7 - OVRD
Invokes the transaction that maintains users that have override access to a transaction.

Processing

Add and Change - The valid values are stored in a PE. The program must collapse all gaps in the PE. It is not necessary for the program to make the valid values ascending.

Only users that are the security administrator of the transaction group specified may perform any maintenance functions against the access flags.

User Group Access To Transaction Groups (Access Maintenance)



```
     TRAN                      SECURITY SUBSYSTEM                    *DATE
     PROGRAM                   ACCESS MAINTENANCE                    *TIME

*Action           : _ (A,B,C,D,M,N,P)
*TRANSACTION GROUP: ________ Transaction group description
*USER GROUP       : ________ User group description
 Allowable Actions: _____________________________
 Access
 Flag   Description                    *Value Description
 ------ ------------------------------ ------ --------------------
   1    Access Flag Description          __   Value Description
   2    Access Flag Description          __   Value Description
   3    Access Flag Description          __   Value Description
   4    Access Flag Description          __   Value Description
   5    Access Flag Description          __   Value Description
   6    Access Flag Description          __   Value Description
   7    Access Flag Description          __   Value Description
   8    Access Flag Description          __   Value Description
   9    Access Flag Description          __   Value Description
  10    Access Flag Description          __   Value Description

*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  OVRD  TGRP  ADMIN
```

This transaction is the heart of the security subsystem. It is on this transaction where one defines the user groups that may access transaction groups. Adding a record on this screen immediately provides all users in the user group access to all transactions in the transaction group (unless there are overrides).

When one links user groups to transaction groups, the allowable actions are also defined. For example, when the users in the 'LAMI worker group' are given access to the transactions in the case transaction group, you may wish to give them inquiry only access [i.e., only specify actions D (display), B (browse), N (next)]. This way these users cannot change any of the case data.

One can also define the access flag values, IF ANY, that the user group has when they access the transaction group. It is very rare for a transaction group to use access flags therefore you shouldn't worry about this functionality overly much. Also, if a transaction group does use access flags, the system will guide you through the entry process.

Only users defined as security administrators for the transaction group may give user groups access to the transaction group.

Any given user group may have an unlimited number of transaction groups that may access it and vice versa. The list action on this screen may be used to display all of the user groups that may access the transaction group. The inquiry "User Groups With Access To Transaction Groups" provides a list of the transaction groups that a given user group may access.

The fields on the maintenance screen are described below :

Action (A1)

The action codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

TRANSACTION GROUP (A8)

The transaction group for which access is being maintained. The transaction group must exist on the transaction group file. The transaction group's description will be displayed adjacent to the field.

USER GROUP (A8)

The user group whose access to the transaction group is being maintained. The user group must exist on the user group file. The user group's description will be displayed adjacent to the field.

Allowable Actions (A20)

This field specifies the actions that the user group is permitted to perform when executing transactions in the transaction group. Valid actions will be strung together into this field. For example, if the user group can Add and Change records, this field will contain “AC”. If the user group is allowed to access all actions, the word “ALL” may be entered. This is an optional field.

Access Flag Group

This group of fields defines the values of the access flags for the user group in terms of the transaction group. Only those access flags that are currently linked to the transaction group are displayed (i.e., those access flags defined on the access flag transaction). The access flag description comes from the access flag file. Adjacent to the access flag is the value of the flag for the transaction group/user group. The description of the value also comes from the access flag file. Max occur: 10, Typical occur: 2.

Value (N2)

This column contains the value of the specific access flag for the user group. A value may only be entered if the access flag is currently linked to the transaction group. If entered, the value must be a valid value as defined on the access flag file.

Additional PF Keys

PF Key 5 - OVRD

Invokes the transaction that maintains that transaction override entity.

PF Key 6 - TGRP

Invokes the transaction that maintains transaction groups.

PF Key 7 - ADMIN

Invokes the transaction that maintains the security administrators for a transaction group.

Processing

The dialogue of this transaction is a little unusual in that first a record is added for the transaction group and user group combination with all of the access flags stored with whatever value is on the screen. The system then automatically refreshes the screen with the access flags that are linked to the transaction group. At this point the value of each flag may be changed.

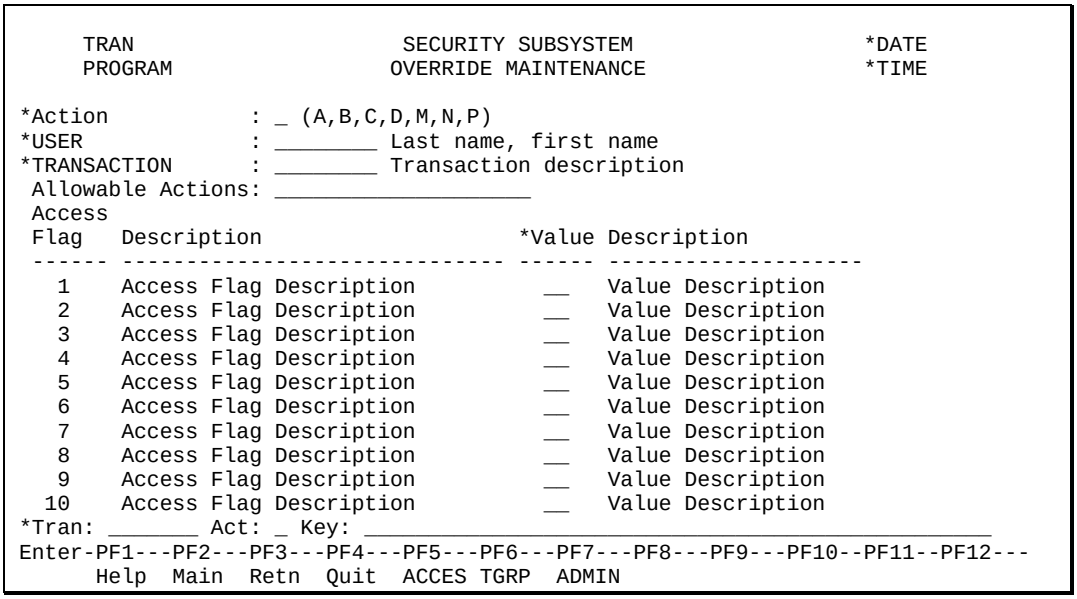
When a record is added, do NOT validate the access flags on the screen because they could be hanging over from a previously displayed transaction group (the validation is dependent upon the transaction group being added). Simply store the record with the access flags that are specified (it might be a good idea to reset any flag that is not defined for the transaction group being added).

When an access record, which affects you, is added, deleted or changed, your security buffer will be refreshed.

If the value of the access flag is not defined on the access flag file, output “** Undefined **” for the value description.

The administrators of a transaction group are the only users who may maintain (A, M, P) the user group/transaction group relationship.

User Override Access To Transaction (Override Maintenance)



```
     TRAN                     SECURITY SUBSYSTEM                    *DATE
     PROGRAM                 OVERRIDE MAINTENANCE                   *TIME

*Action           : _ (A,B,C,D,M,N,P)
*USER             : ________ Last name, first name
*TRANSACTION      : ________ Transaction description
 Allowable Actions: ____________________
 Access
 Flag   Description                    *Value Description
 ------ ------------------------------ ------ --------------------
   1    Access Flag Description          __   Value Description
   2    Access Flag Description          __   Value Description
   3    Access Flag Description          __   Value Description
   4    Access Flag Description          __   Value Description
   5    Access Flag Description          __   Value Description
   6    Access Flag Description          __   Value Description
   7    Access Flag Description          __   Value Description
   8    Access Flag Description          __   Value Description
   9    Access Flag Description          __   Value Description
  10    Access Flag Description          __   Value Description
*Tran: _______ Act: _ Key: ____________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  ACCES TGRP  ADMIN
```

This transaction is used to identify overrides (exceptions) to the access rules that have been defined for the user's user group. For example, the need may arise for a particular user to have special access to a transaction not normally granted for the user group. Or conversely, one may wish to remove a user's access to a specific transaction altogether.

This transaction has the same functionality as defined for the access maintenance transaction. The only difference between the two is that access maintenance is operating at the user GROUP/transaction GROUP level and this program operates at the specific user and transaction level.

Only users defined as security administrators for the transaction's transaction group may give users override access.

Any given user may have an unlimited number of transaction overrides and vice versa. The list action on this screen may be used to display all of the transactions for which a user has overrides. The inquiry “Transactions With Overrides” provides a list of the users that have overrides for a given transaction.

The fields on the maintenance screen are described below:

Action (A1)
: The action codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

USER (A8)
: The identity of the user for whom the override is being maintained. The user id must exist on the user file. The user's name will be displayed adjacent to the field.

TRANSACTION (A7)
: The identity of the transaction for which the user has an override. The transaction must exist on the transaction file. The transaction's description will be displayed adjacent to the field.

Allowable Actions (A20)
: This field specifies the actions that the user is permitted to perform when executing the transaction. Valid actions will be strung together into this field. For example, if the user can Add and Change

records, this field will contain “AM”. If the user is allowed to access all actions on the transaction, the word “ALL” may be entered. If the user is not allowed to access the transaction, the word “NONE” must be entered. This is an optional field.

Access Flag Group

This group of fields defines the values of the access flags for the user in terms of the transaction. Only those access flags that are currently linked to the transaction's transaction group are displayed (i.e., those access flags defined on the access flag transaction). The access flag description comes from the access flag file. Adjacent to the access flag is the value of the flag for the transaction/user. The description of the value also comes from the access flag file. Max occur: 10, Typical occur: 2.

Value (N2)

This column contains the value of the specific access flag for the user. A value may only be entered if the access flag is currently linked to the transaction's transaction group. If entered, the value must be a valid value as defined on the access flag file.

Additional PF Keys

PF Key 5 - ACCES

Invokes user group access to transaction group transaction

PF Key 6 - TGRP

Invokes the transaction that maintains transaction groups

PF Key 7 - ADMIN

Invokes the transaction that maintains the security administrators for a transaction group

Processing

The dialogue of this transaction is a little unusual in that first a record is added for the transaction and user combination with all of the access flags stored with whatever value is on the screen. The system then automatically refreshes the screen with the access flags that are linked to the transaction's transaction group. At this point the value of each flag may be changed.

When a record is added, the override counter on the user's record and the override counter on the transaction record are incremented by one.

Do NOT validate the access flags on the screen because they could be hanging over from a previously displayed transaction (the validation is dependent upon the transaction being added). Simply store the record with the access flags that are specified (it might be a good idea to reset any flag that is not defined for the transaction group being added).

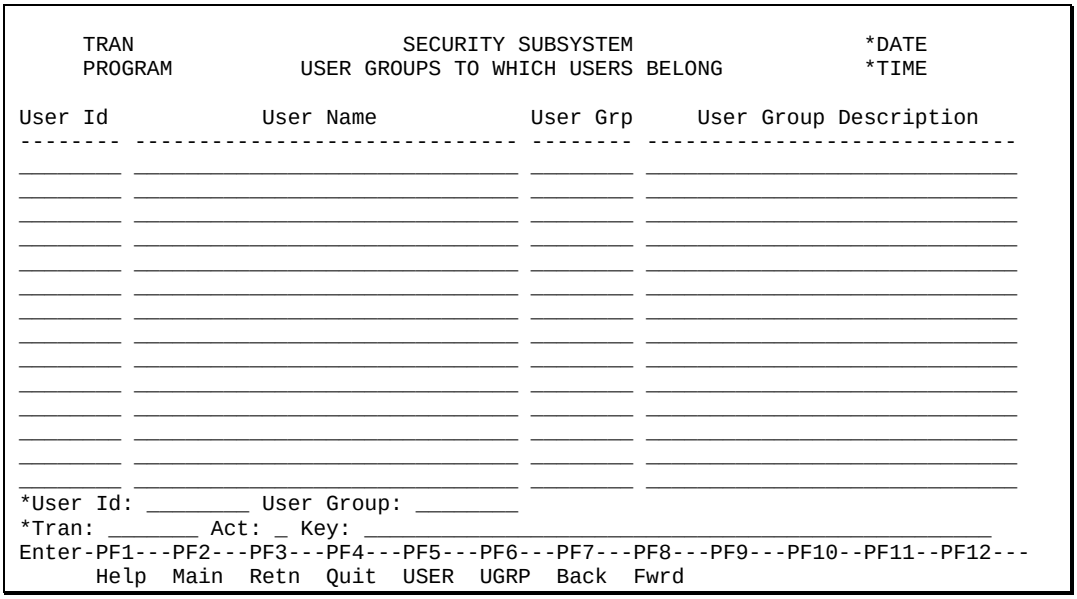
When an override record is added, deleted, or changed, for yourself, your security buffer will be refreshed.

Whenever a record is deleted, the override counter on the user's record and the override counter on the transaction record are to be decreased by one.

The administrators of the transaction's transaction group are the only users who may maintain (A,M,P) the transaction/user relationship.

Inquiries

User Groups To Which A User Belongs



This inquiry displays the user groups to which a user belongs. It will display the user's name as well as the user group id and the description of all user groups of which the user is a member.

Additional PF Keys

PF Key 5 - USER

This is a cursor sensitive key that will invoke the user maintenance transaction

PF Key 6 - UGRP

This is a cursor sensitive key that will invoke the user group maintenance transaction

Processing

Read the user group/user file in user id, user group order.

User Groups With Access To Transaction Groups

```
     TRAN                     SECURITY SUBSYSTEM                 *DATE
     PROGRAM          USER GROUP ACCESS TO TRANS. GROUPS         *TIME  1 >

 User                                  Tran
 Group       User Group Description    Group  Transaction Group Description
--------- ------------------------------ -------- -----------------------------
-
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________
_________ _____________________________ ________ ____________________________

*User Group: ________ Transaction Group: ________
*Tran: _______ Act: _ Key: _________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  UGRP  ACCES Back  Fwrd        Left   Rght
```

```
     TRAN                     SECURITY SUBSYSTEM                 *DATE
< 1  PROGRAM           USER GROUP ACCESS TO TRANS. GROUPS        *TIME

 User      Tran                            ----- Access Flag Values ----
 Group     Group     Actions Allowed     01 02 03 04 05 06 07 08 09 10
-------- --------  -------------------- -- -- -- -- -- -- -- -- -- --
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __
_______  _______  ____________________ __ __ __ __ __ __ __ __ __ __

*User Group: ________ Transaction Group: ________
*Tran: _______ Act: _ Key: _________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12
      Help  Main  Retn  Quit  UGRP  ACCES Back  Fwrd        Left   Rght
```

This inquiry displays all of the transaction groups to which a user group has access. Scrolling to the right will display additional information, repeating the user group id and transaction group id and further showing all actions allowed as any access flag values specific to that linking.

Additional PF Keys

PF Key 5 - UGRP

This is a cursor sensitive key that will invoke the user group maintenance transaction

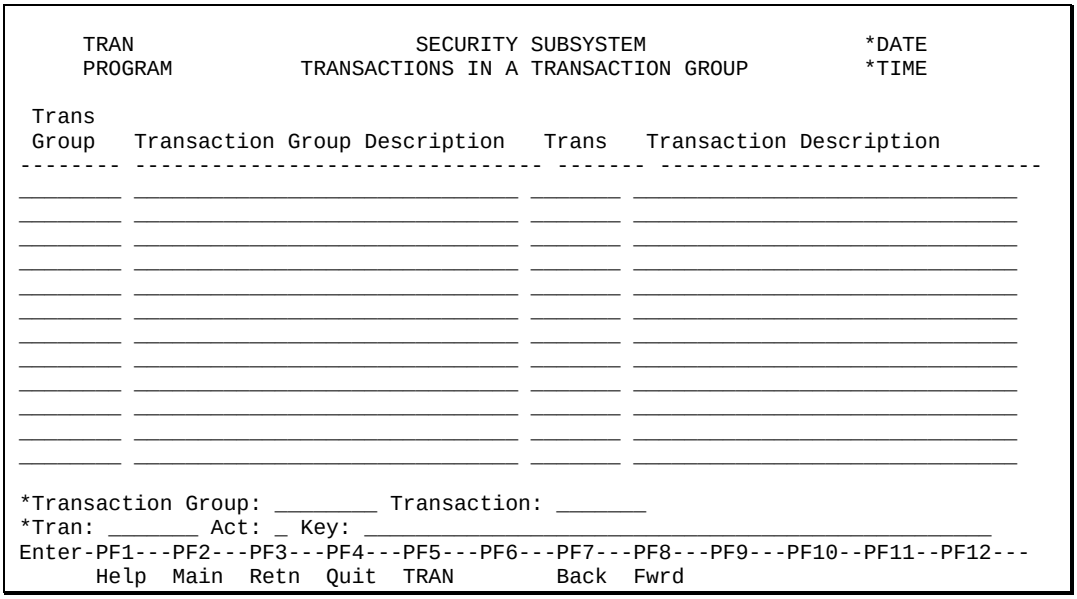
PF Key 6 - ACCES

This is a cursor sensitive key that will invoke the access maintenance transaction

Processing

Read the user group/transaction group file in user group, transaction group order.

Transactions In A Transaction Group



This inquiry displays the transactions that are members of a transaction group.

Additional PF keys

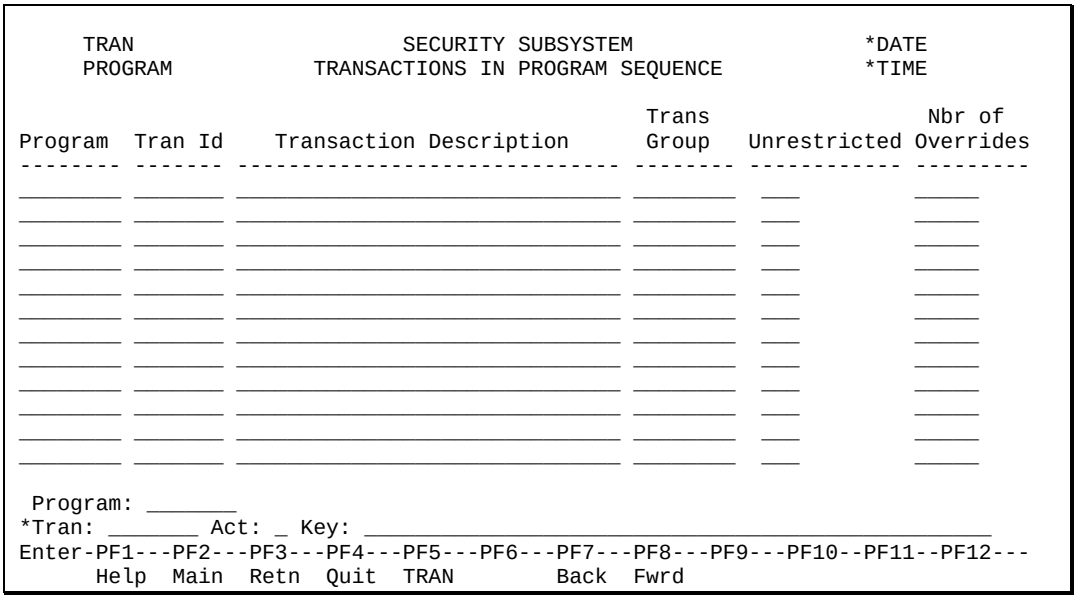
PF Key 5 - TRAN

This is a cursor sensitive key that will invoke the transaction maintenance transaction

Processing

Read the transaction file in transaction group/transaction order.

Transactions In Program Sequence



This inquiry displays the transactions in program id sequence. Aditionally, if the transaction has unrestricted access. “Yes” will appear under the heading “Unrestrict” and the number of overrides that exist for the particular transaction will be displayed where applicable.

Additional PF keys

PF Key 5 - TRAN

This is a cursor sensitive key that will invoke the transaction maintenance transaction

Processing

Read the transaction file in program id, transaction id order.

If the transaction has unrestricted access (check the flag), output the word “Yes”, otherwise leave the column blank. Zero suppress the transaction override count.

Transactions With Overrides

```
     TRAN                      SECURITY SUBSYSTEM                *DATE
     PROGRAM              TRANSACTIONS WITH OVERRIDES            *TIME    1 >

Tran Id           Tran Desc           User Id           User Name
------- ------------------------------ -------- -----------------------------
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________
_______ ______________________________ ________ _____________________________

*Transaction: _______ User: ________
*Tran: _______ Act: _ Key: _____________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  OVRD        Back  Fwrd        Left   Rght
```

```
     TRAN                      SECURITY SUBSYSTEM                *DATE
< 1  PROGRAM             TRANSACTIONS WITH OVERRIDES             *TIME

                       Allowable        ----- Access Flag Value  ----
Tran Id User Id         Actions         01 02 03 04 05 06 07 08 09 10
------- --------  --------------------  -- -- -- -- -- -- -- -- -- --
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
_______ ________  ____________________  __ __ __ __ __ __ __ __ __ __
*Transaction: _______ User: ________
*Tran: _______ Act: _ Key: _____________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  OVRD        Back  Fwrd        Left   Rght
```

This inquiry displays the users with overrides for a particular transaction. Scrolling to the right shows the actions allowed for that user and any access flag values.

Additional PF Keys

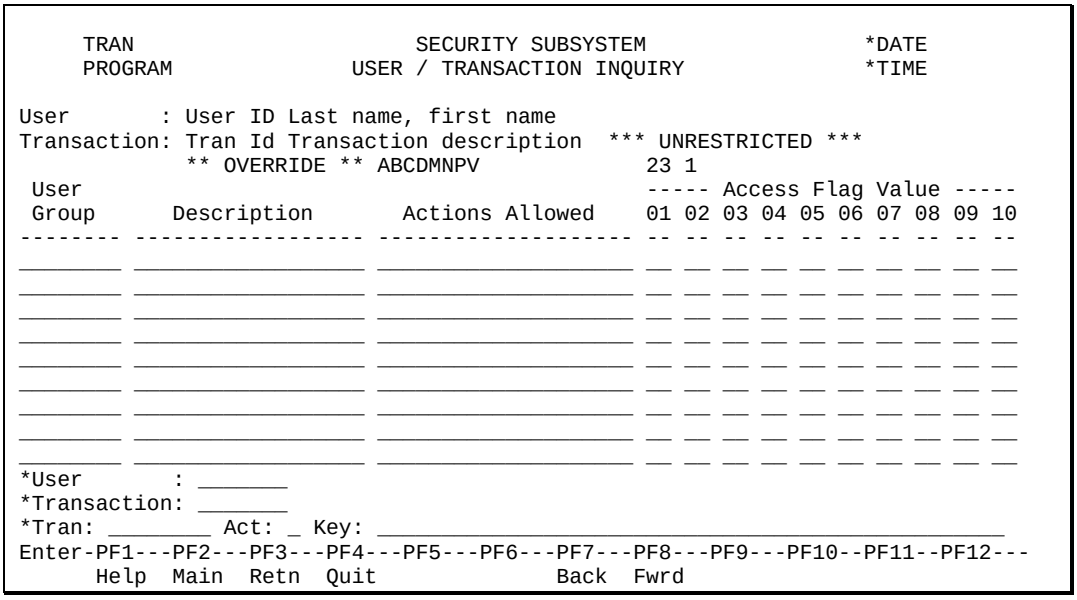
PF Key 5 - OVRD

This is a cursor sensitive key that will invoke the override maintenance transaction

Processing

Read the override file in transaction, user id order.

User/Transaction Inquiry



This inquiry displays the access, if any, a user has to a given transaction. This access may be given through the system explicitly, through the override transaction, or implicitly by defining the access for the user's user group.

If the transaction is unrestricted, this is indicated. The remainder of the display shows any user groups to which the specific user belongs that have access to the specific transaction via its transaction group. Available actions for this linkage and any access flags are shown here.

Additional PF Keys

Processing

At the top of the screen display:

- The user id and the user's name.
- The transaction id and the transaction's description.
- If the transaction has unrestricted access, write out '*** UNRESTRICTED ***' in red.
- If the user id has an override for the input transaction, write out the '** OVERRIDE' in red, and display the allowable action and access flags on the override record.

Help Routines

User

This will be a simple help routine that will read the user file in last name, first name, user id order. The key fields will be the only information displayed.

User Multi-line

This will be a simple help routine that will read the user file in last name, first name, user id order. It will set up an index parameter to return to the calling program. The key fields will be the only information displayed.

User Group

This will be a simple help routine that will read the user group file in user group order. The user group id and its description will be displayed.

Transaction Group

This will be a simple help routine that will read the transaction group file in transaction group order. The transaction group id and its description will be displayed.

Transaction

This will be a simple help routine that will read the transaction file in transaction order. The transaction id and its description will be displayed.

Normal Online Transactions

This will be a simple help routine that will read the transaction file in transaction type transaction order. Only read records of transaction type 'N'. This helproutine will be used on the main menu fields. The transaction id and its description will be displayed.

Normal Online Transactions for Tran Line

This will be a simple help routine that will read the transaction file in transaction type transaction order. Only read records of transaction type 'N'. This helproutine will be used on the Tran field of all transactions. It is identical to the normal help except that it will not set up any return parameters. The transaction id and its description will be displayed.

Access Flag Valid Values

This will be a subprogram type of help in that it will have to be passed a parameter identifying the transaction group and the access flag number for which help is required. This routine will attempt to find the record on the access flag file for the specified parameters. If nothing can be found, output an appropriate message in the window (or in a return message). If something is found, display the elements of the valid values PE in a window that is identical to a normal help routine's window.

Batch Reports

User - Transactions Report

Parameters

User id or “ALL” or '*'. The wild card '*' may also be used to print a subset of user id's, for example to report on all user id's beginning with A, input A*.

Sort Sequence

User id
Transaction group
Transaction id
User group

Contents

User id

User name (first name, middle initial and last name)

A line for each transaction in each transaction group for which the user's group has access. A line for each transaction that has been made available to the user through the override function.

The transaction lines will be organized by transaction group, and each group will have a heading of:

Transaction group

Transaction group description

Each transaction line will contain:

Transaction id

Transaction description

Actions allowed (or “NONE”)

An indication of whether the line represents access from an override record for this transaction.

An indication of whether the line represents access from an access record for the transaction's transaction group.

An indication of whether access parameters are present.

User Group id

User Group description

SAMPLE REPORT

```
         1         2         3         4         5         6         7         8         9         0         1         2         3
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
*PROGRAM                                          SECURITY SUBSYSTEM                                                  Page: 0001
MM/DD/YY                                      USER / TRANSACTIONS REPORT                                              HH:MM AM

User      : XXXXXXXX last name, first name

                                                 User
          Tran Id          Description           Group           Description          Override   Actions Allowed    Flags
          ------- ------------------------------ -------- ------------------------------ -------- -------------------- -----
Tran Group: XXXXXXXX tran group description
          XXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX          XXXXXXXXXXXXXXXXXXX  yes
                                                 XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX          XXXXXXXXXXXXXXXXXXX
          XXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXX                                        yes     XXXXXXXXXXXXXXXXXXX  yes
                                                 XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX          XXXXXXXXXXXXXXXXXXX  yes
                                                 XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX          XXXXXXXXXXXXXXXXXXX

Tran Group: XXXXXXXX tran group description
          XXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX          XXXXXXXXXXXXXXXXXXX
          XXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX          XXXXXXXXXXXXXXXXXXX
          XXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX          XXXXXXXXXXXXXXXXXXX  yes

-- page break --

UNRESTRICTED TRANSACTIONS:

          Tran Id          Description
          ------- ------------------------------
Tran Group: XXXXXXXX tran group description
          XXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Common Routines

Output User Name

This routine is used to set up a user name output field.

Input:
User Id (A8)

Output:
Error (L)

User Name (A37)

Initial Processing

Reset the output variables.

No validation of input.

Mainline Processing

Attempt to find the record on the user file with a key equal to the input value. If nothing found set User Name to '*** Undefined ***' and proceed as described in error processing.

Populate user name in 'last name, first name' format.

Error Processing

Assign the error flag to true and return to the calling program.

Retrieve Transaction Information

This routine is used to set up a transaction output field, to validate the transaction id, and to return information about the transaction.

Input:

Tran Id (A7)

Type of Program (A1)

Validate (L)

Access by Type (L)

Output:

Error (L)

Description (A30)

Tran Group (A8)

Override Count (P5)

Unrestricted (A1)

Initial Processing

Reset the output variables.

No validation of input.

Mainline Processing

This program has two different uses depending upon the value of the validate indicator. If validate is true then do not populate the transaction output fields, simply set up the error indicator if the transaction is not on file. If validate is false then set up all output fields.

If the access by type indicator is set then access the transaction file using the Type of Program input field.

Error Processing

Assign the error flag to true, if not validating then set up description as *** Undefined ***, and return to calling program.

Output Access Flag and Valid Value Descriptions

This routine is used to set up a description for a transaction group access flag and a specified valid value.

Input:

Transaction Group (A8)

Flag Value (N2)

Output:

Error (L)

Flag Desc (A30)

Value Desc (A30)

Initial Processing

Reset the output variables.

No validation of input.

Mainline Processing

Read the transaction group access flag file with a key corresponding to the input values. If no access flag found for an input valid value set flag desc to '*** Undefined ***' and proceed with error logic. If no valid value found set the value desc to '*** Undefined ***' and proceed with error logic.

Populate the output variables with the attributes found on the file and return.

Error Processing

Assign the error flag to true and escape routine.

Store Log Records

These routines are used to store log records. They will be needed for the following files: user group transaction administrator, user group user, transaction group administrator, transaction group access flag, access, and override.

Input:

View

Action (A1)

Program (A8)

Initial Processing

Set up all standard log fields. Use the input program and action to set up log program and log action.

No validation of input.

Mainline Processing

Move the input view to the log view and store. Do not ET.

Error Processing

None

Transaction Access Control

One of the objectives of the security control system is to provide a centralized, standard way to effect the transfer of control between transactions. Another objective is to notify the applicable program(s) of the actions that the user may perform when executing the transaction, and of any user specific parameters that have been recorded for the transaction.

The mechanics of the access control process can be divided into two parts.

Firstly, a transference subprogram will be invoked each time control is to be transferred to a new transaction. The logic to invoke the subprogram will reside in copy code that will be included in each program, subprogram, helproutine, and subroutine that needs to transfer control to a new transaction.

The transference subprogram will be passed the name of the new transaction, as well as the key (if any) entered on the command line. The subprogram will establish whether the user is allowed to execute the transaction that was requested.

If the user may execute the transaction, the subprogram will pass control to the applicable program. If the user is not allowed to execute the transaction, control will be passed back to the program that invoked the security control subprogram, and an error message will be displayed.

The second component of the access control process consists of a set of copy code that is to be included at the top of each program, subprogram, subroutine or helproutine that is subject to control under the security control system. This copy code will provide the program, subprogram, subroutine or helproutine with the actions and parameters specific to the transaction. It will also double check that the user is allowed to execute the transaction.

Transference subprogram

In order to minimize file access, the transference subprogram maintains a large portion of its information in the user's security buffer.

The security buffer is defined to have:

- A table of the 20 most recently accessed transactions.

  Each entry in this transaction table contains:

  - The transaction id.
  - The program id (from the transaction record).
  - The library.
  - The actions that the user may perform when he executes the transaction.
  - Any user specific access flags to be used when the transaction is processed.
  - A table of the 10 most recently accessed transaction groups. Each entry in this transaction group table contains:
    - The transaction group id.
    - The actions that the user may perform when he executes transactions in the group.
    - Any user specific access flags to be used when the user executes transactions in this group.
- Information pertaining to the user. The following user information is stored in the security buffer:
  - The user's transaction override count.
  - The name of the user's main menu program.
- A table containing the last 10 transactions invoked, stored in the order in which they were invoked. This table is used to return the user to previously executed transaction when the return key is pressed.

The transference subprogram is invoked whenever control is transferred between transactions. It is passed the identity of the requested transaction on a parameter list. The subprogram functions as follows:

1. When the subprogram is invoked for the first time during a user's session, it reads the user file and stores the user's transaction override count and main menu program name in the user's security buffer.
2. If the user presses PF4 to return to the main menu this can mean one of two things:
   - Return to the main menu associated with the transaction on which they are working.
   - Return to the user's main menu (this happens when the transaction id of the current program is equal to the transaction id of the current program's transaction group's main menu).
3. If the user presses PF3 to go back to the previous transaction, the transference subprogram checks the transaction table in the user's security buffer to find the previous transaction processed, and passes control to that transaction.

4. When the transference subprogram is invoked with a request for a transaction, it searches the transaction table in the user's security buffer to see if the transaction is one of the last 20 that the user executed. If the transaction is in the table, no further processing is necessary and control is passed to the transaction.

5. If the transaction is not found in the security buffer transaction table, the transference subprogram reads the transaction record from the transaction file.

6. If the transaction is unrestricted, the transaction information is added to the transaction table in the user's security buffer, and control is passed to the transaction.

7. If the transaction's override counter on the transaction record is not zero, and the user override count (from the security buffer) is not zero, there is a chance that the user has an override record for the transaction. In this case, an attempt is made to read an override record for this user and this transaction.

8. If the override record is found, and the user has not been disallowed access to the transaction, the transaction information is added to the transaction table in the user's security buffer, and control is passed to the transaction.

9. If no override record exists for the user for this transaction, the transaction group table in the user's security buffer is inspected.

10. If the transaction group is found in the table, the transaction information is added to the transaction table in the user's security buffer, and control is passed to the transaction. If the transaction group is not in the user's security buffer transaction group table, the access file is searched for an entry that specifies access to the transaction group for any one of the user's user groups. If such an entry is found, the user may execute the transaction, and is allowed to perform the actions specified on the access record.

11. If the user belongs to more than one user group that has an access record for the transaction group, the transference subprogram will assume that the user is allowed to perform any of the actions specified on any of the access records. The access records will be read, and the highest level of authority found for each of the user's access flags will be set up (in the security buffer).

12. The transaction group information is entered into the transaction group table in the user's security buffer.

13. The transaction information is added to the transaction table in the user's security buffer, and control is passed to the transaction.

Note

The transference subprogram will, wherever possible, find transaction and/or transaction group information from the temporary area tables. These tables are refreshed whenever CICS is brought down, and when certain maintenance actions are performed on the Security files. The “Programming Notes” paragraphs in the MAINTENANCE TRANSACTIONS section of this document discuss the conditions under which a user's temporary storage area is refreshed.

Populate security parameters

Each NATURAL object (that is program, helproutine, subprogram or subroutine) that is subject to control through this security system is to include the following:

- A local data area that contains the fields to be used to hold security information pertaining to this program, subprogram, subroutine or helproutine. This information includes:
  - The name of this NATURAL object.
  - An indicator that specifies whether or not the user may execute this NATURAL object.
  - The actions that this user may perform.
  - The access flags that are specific to this user and this transaction group.

  In this discussion this Local Data Area is referred to as the **security LDA**.
- A LOCAL DATA AREA that specifies the area to be used to pass information to the transference subprogram. This information includes the name of the transaction to which control is to be passed, and the key (if any) to be passed to the transaction. In this discussion this Local Data Area is referred to as the **transference LDA**.
- A call to a security checking subprogram. The security checking subprogram populates the security LDA with information from the transaction table in the user's security buffer, and has special logic that is executed if the NATURAL object is not found in the user's security buffer. (In this case, the subprogram performs a subset of the transference subprogram's logic to check whether the user has access to the object.)
- Code that checks the values in the security LDA. If the user is not permitted to access this object, this code outputs an error message, and returns the user to the transaction's menu. If the user is permitted to access this object, the valid action codes are set up for display on the transaction screen, and for internal validation.
- Code that invokes the transference subprogram, and passes it information via the transference LDA, whenever control is to be transferred to another transaction.

The Local Data Areas described above are defined outside the programs. Each program, subprogram, helproutine and subroutine includes these local data area definitions through the USING clause.

The calls to the transference and security checking subprograms, and the code that checks the security LDA values will be coded once. Each program, subprogram, helproutine and subroutine will INCLUDE this “copy” code.

Installation Procedure

Installation Transaction, SCINSTF

This transaction will perform the initialization of the security environment when the system has been delivered to the user site. This procedure is to be executed when all programs have been loaded, the security files have been created and the data definition modules customized for the environment.

The user will log onto the security library and execute SCINSTF. The user will then be asked to enter an initial user id. The program will check to make sure the initial program has not previously been run. Once that has been determined, initial entries will be added to the following files:

SC-USER will be created with an entry for the user id entered above, a last name of 'SECURITY USER', a first name of 'INITIAL', a main menu tran id of 'SC' and a functional group of 'INST'.

SC-FUNCTIONAL-GROUP will be created with a functional group of 'INST' and a description of 'INSTALL'.

SC-TRAN-GROUP will be created with a tran group of 'SECURITY' and a description of 'SECURITY TRANSACTIONS'.

SC-TRAN-GROUP-ADMIN will be created with a tran group of 'SECURITY' and an admin user id of the user entered above.

Processing

SC-TRAN entries will be created for every transaction in the security subsystem with tran id, description, program name and type of program entered. Tran group will be 'SECURITY'. Unrestricted will be 'Y', and library id will be *LIBRARY-ID.

Information Output On The Screen

As each entity is added a message will appear on the screen, informing the operator that an entry has been added. The message will show the type of entity added and its value. For example: 'Tran group SECURITY added'. For each transaction the message should be 'Transaction XXXXXXXX (YYYYYYYYYYYYYY) added Unrestricted', where 'XXXXXXXX' is the transaction id and 'YYYYYYYYYYYYYY' indicates a batch program, an on-line program or a helproutine.

When all entries have been added, closing messages should be displayed. These are: '***** Security System Initialization Complete *****' and 'All transactions have been added without any restrictions.'

Logging

Standard logging.

Workflow Tracking (Review List) Subsystem

Introduction

There is often a need to advise a user that something in the system requires attention. Some simple examples are: a reminder of a scheduled meeting, a note that a purchase order of a significant value is being processed or notice that a vendor has been suspended. The review list (workflow tracking subsystem) provides this facility.

The review list can also be used to perform more complicated functions, such as tracking the progress of a document or monitoring a process. Examples of such functions are notification of each stage at which a requisition requires approval, or advice that a month close has begun.

This chapter includes the following sections:

- Glossary of Terms and Review List Characteristics
- Review List Processing
- Review List Maintenance
- Review List Browses
- Tracking

Glossary of terms and Review List characteristics

Glossary of terms

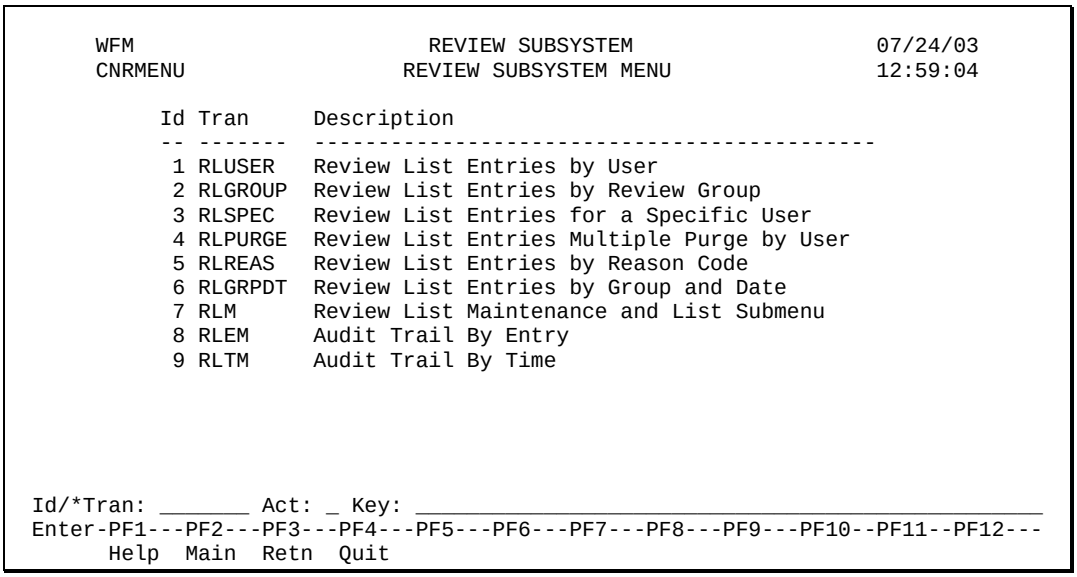
Review list entry	An item on the review list that draws a user's attention to something in the system, usually a function, document or process.
Review group	A group of users who can take action with respect to review list entries addressed to their group. A review group is used to define a set of users who have a specific capability in terms of the review list subsystem. This provides the mechanism to address a review list entry to a group of users, any one of whom can take action on it. This is useful when there is a 'pool' of users who have equivalent capability in terms of the review subsystem (e.g., a buyer pool).
Related object	Anything in the system that a review list entry refers to (e.g., a purchase order, a requisition line or a buyer).
Tracking table	A user-defined table that details review list entries that are to be created when the value of a specific field on a document held in the system changes
Tracking entity	An entity being monitored by the tracking table. The entity can be a field or a combination of fields, a document or a process.
Designated user	The user to whom a review list entry is addressed, or a member of the review group to which it is addressed.
Power of Attorney	The ability to act on behalf of another user within the review list. This enables a user to select or modify a review list entry addressed to another user.

Review List characteristics

- A review list entry can be addressed either to a review group or to a user.
- An entry may contain a reason for the review and identification of the related object.
- The creator of the entry can specify the date on which the entry is to appear on the review list.
- Any user can browse through all review list entries, but only a designated user or a user who has been granted power of attorney by the designated user can modify or select an entry for review.
- The addressee of a review list entry can be changed, thereby redirecting the entry to someone else. An entry can also be copied.
- When a review list entry is selected, the subsystem can, under certain circumstances, invoke the necessary transaction so that the review can be done immediately.
- There are two types of review list entries: reminding review list entries and tracking review list entries. A reminding entry can be generated by a user, or from the tracking table, or by an application program. Reminding entries are deleted from the review list upon selection. A tracking entry is generated from the tracking table. Tracking entries are automatically deleted from the review list only when the tracking is complete.
- Entries created from the tracking table determine the addressee from the tracking table itself. For other entries, the addressee is supplied by the user, or is determined from the supplied reason code.
- The review list is accessible from the main system menu.
- A full audit trail is kept of every review list entry, recording when it was created, modified or deleted.

Review Subsystem Menu

The Workflow Subsystem menu can be accessed from the MEDS main menu. This menu gives you access to several submenus.



Review Subsystem Menu

To choose a submenu, enter its code in the Code field and press ENTER.

CODE	DESCRIPTION
1	Review List Entries by User This option lets you view review list entries addressed to users. You can scroll through the entries, and possibly select one for further review. At this screen you can also create or modify review list entries.
2	Review List Entries by Review Group This option lets you view review list entries addressed to review groups. You can scroll through the entries, and possibly select one for further review. At this screen you can also create or modify review list entries.
3	Review List Entries for a Specific User This option lets you view review list entries addressed to a specific user and the review groups to which the user belongs. You can scroll through the entries, and possibly select one for further review. At this screen you can also create or modify review list entries
4	Purge Multiple Review List Entries by User This option allows purges and modifies of multiple review lists for a user at one time.
5	Review List Entries by Reason Code This option lets you view review list entries in order of their reason code. You can scroll through the entries, and possibly select one for further review. At this screen you can

also create or modify review list entries

6 Review List Entries by Group and Date

This option lets you view review list entries addressed to review groups by date. You can scroll through the entries, and possibly select one for further review. At this screen you can also create or modify review list entries.

7 Review List Maintenance and List Submenu

This option lets you enter, maintain and display various files used to facilitate review list processing.

8 Audit Trail by Entry

This option gives you access to a submenu from which you can list the audit trails on the review list in entry sequence.

9 Audit Trail by Time

This option gives you access to a submenu from which you can list the audit trails for the review list in chronological sequence.

Review List Processing

Review List Entries by User

When you enter '1' at the Review List Menu, you see the Review List Entries by User screen. At this screen you can see the review list entries that are addressed to a user Id. You can add, change, delete, and forward review list entries from this screen and, in some cases, select a review list entry for further review.

```
Position cursor or ENTER screen value to select
     TRAN                      REVIEW SUBSYSTEM                  *DATE
     PROGRAM             REVIEW LIST ENTRIES BY USER             *TIME    1 >

User                          Reason /
Id       Py Eff Date      Related Object
-------- -- -------- ---------------------------------------
SPLEF     1 08/12/90 PURCHASE ORDER FOR MORE THAN $10000
          1 08/15/90 REVIEW THE PURCHASE ORDER
                     Purchase Order: P100000012
          2 08/15/90 APPROVE THIS PO PLEASE
                     Purchase Order: PO00287562
          5 08/15/90 REVIEW THE REQUISITION
                     Requisition: RQ00488129
SPLJS     1 08/15/90 Look at PO12345678 and call me
          1 08/15/90 Error in PRPMNTF PURCHASE ORDER NEEDS AP
                     Purchase Order: P100000012
                        ****** End of Data ******

*User Id: ________ Mark for Future: _  or For this Date: ________
*Tran: _______ Act: _ Key: ____________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  modif       Back  Fwrd  delet Left  Rght  creat
```

Review List Entries by User Screen - 1

This selection list has two screens. At both screens you can enter a User Id to start the scroll. If you don't enter a code and simply press ENTER, the display begins with review list entries addressed to your user Id. You can get active help for user Id. The review list entries are displayed in sequence of addressee user id, priority and date. Normally only review list entries with an effective date not greater then the current date are displayed. If you want to see review list entries dated into the future, enter any character in the 'Mark for Future Review List entries' field. To turn this feature off, enter a space in this field. Enter a specific date to see only those review lists for that user effective on that date. The date can be in the past, present or future.

For each review list entry, you see the following information:

Action

You can perform the following actions on each review list item:

- Add - use the add action if you want to copy the review list item.
- Change - use the change action if you want to update the item.
- Delete - use the delete action if you want to delete the item.
- Select - use the select action if you want to be transferred to the transaction associated with this review list. When you exit the transaction, you are returned to this review list transaction. Note that the select action may not be used in conjunction with other actions.

- Forward - use the forward action if you want to forward the item. Note that in order to forward you must also specify the Forward To User at the bottom of the screen.

User Id
: The Id of the designated user.

Py
: Priority of the entry. This is used to sequence the review list entries in some sort of priority over date.

Eff. Date
: The effective date. Normally, only review list entries with an effective date up to the present are displayed. To see entries dated into the future, enter any character in the Mark for Future Entries field. To see entries for a particular date, enter a valid date in For this Date field.

Reason/ Related Object
: This indicates the purpose of the review list entry and what (if anything) it relates to.

On the screen to the right, you see the following:

```
Position cursor or ENTER screen value to select
     TRAN                      REVIEW SUBSYSTEM                    *DATE
< 1  PROGRAM              REVIEW LIST ENTRIES BY USER              *TIME

                                                 Delete
User Id  Prty  Eff. Date  No.   Type    With PFKey
-------- ----  ---------  ---  -------  ----------
SPLEF       1  Aug 13,90   1   Remind      Yes
            1  Aug 16,90   2   Track       No
            2  Aug 16,90   3   Track       No
            2  Aug 16,90   4   Track       No
            5  Aug 11,90   5   Remind      No
SPLJS       1  Aug 12,90   6   Remind      Yes
            1  Aug 13,90   7   Track       No
            1  Aug 15,90   8   Remind      Yes
            1  Aug 15,90   9   Remind      Yes
            1  Aug 15,90  10   Track       Yes
            1  Aug 15,90  11   Track       No
            1  Aug 15,90  12   POA         Yes
            1  Aug 15,90  12   POA         Yes
*User Id: ________ Mark for Future: _  or For this Date: ________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  modif       Back  Fwrd  delet Left  Rght  creat
```

Review List Entries by User Screen - 2

Type
: This indicates the type of review list entry. Valid types are reminding, tracking, or notification of use of power of attorney.

Delete With PFKey
: This indicates whether or not the review list entry will be deleted when you enter the line action Delete.

This is a standard selection list screen, but it has a few special PFKeys for review list processing.

Use of PFKeys on the Review List

In addition to the standard PFkeys, the following are available on the review list:

PF1
: Displays help text description for the review list reason code.

PF5
: Modify a review list entry.

PF9
: Delete a review list entry

PF12
: Add a new review list entry.

Adding a Review List entry

To create a new entry, press PF12 at any of the review list processor screens. To use the field values of an existing entry as the basis for creating a new entry (i.e. copy the entry), use the line action Add.

```
Position cursor or ENTER screen value to select
     TRAN                      REVIEW SUBSYSTEM                   *DATE
     PROGRAM             REVIEW LIST ENTRIES BY USER               *TIME    1 >
    +-------------------------------------------------------------------+
    |                  Create Review List Entry        *DATE            |
   U|    PROGRAM                                       *TIME            |
    |                                                                   |
 _ S|    *Reason Code: ________                                         |
 _  |         Purpose: ________________________________________         |
 _ S|        *User Id: ________                                         |
 _  |   *Review Group: ________                                         |
 _  |        Priority: ___                                              |
 _  |  Effective Date: __ __ ____                                       |
 _  |    *Transaction: ________                                         |
 _  |  Related Object:                                                  |
 _  |                                                                   |
 _  |                                                                   |
 _  | Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--|
    |       help        retrn                                           |
 *U +-------------------------------------------------------------------+
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  modif       Back  Fwrd  delet Left  Rght  creat
```

Create a Review List Entry Screen

The following rules govern the creation of review list entries. These rules apply when an entry is created by a user, and when the system creates a review list entry from the tracking table or from an application program.

Each review list entry includes the following fields:

Reason Code
: The reason code is used to look up the reason for the review list entry. This is validated against the reason table. You must enter either a reason code or purpose (next field).

Purpose
: The description of the reason for the review list entry. See above.

User Id
: You must enter an addressee, either a review group (next field) or a user Id. The user Id you enter is validated against the user file.

Review Group
: The entry in this field is validated against the review group file. See above.

Priority
: This is used to sequence the review list entries in some sort of priority over date. If you do not enter a value, a default of '5' is used unless a reason code is specified in which case the default priority of the reason code would be used.

Effective Date
: The date on which the entry is to appear on the review list. If you don't enter a date, the current date is used.

Transaction

The Id of the transaction that is invoked when the review list entry is selected. The entry in this field is validated against the transaction file.

Related Object

This is only present on a copy and cannot be modified. It identifies the specific object to which this review list entry is linked (e.g., a requisition number or a vendor Id).

If you don't enter a user Id or review group, and you do enter a reason code, the user Id or review group is retrieved from the reason code file. If you don't enter a reason code, you must enter purpose and a user Id or review group.

Changing a Review List entry

To change a review list entry, enter the Change line action. The following screen appears.

```
Position cursor or ENTER screen value to select
     TRAN                          REVIEW SUBSYSTEM                  *DATE
      +-----------------------------------------------------------+ *TIME 1 >
      |Please enter the changes to this Review List               |
      |                      Modify Review List Entry    *DATE    |
  Us  |  PROGRAM                                         *TIME    |
  --  |                                                           | -
c UA  |   *Reason Code: DEPNBND_                                  |
      |        Purpose: ____________________________________      |
      |       *User Id: UA#0074_                                  |
      |  *Review Group: ________                                  |
      |       Priority: 1__                                       |
      | Effective Date: 12 15 1995                                |
      |   *Transaction: DEPN___                                   |
      | Related Object: Account: 511 DOE, NANCY                  |
      |                                                           |
      |                                                           |
      |Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9-- |
      |      help        retrn                                    |
      +-----------------------------------------------------------+
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  modif       Back  Fwrd  delet Left  Rght  creat
```

Modify a Review List Entry Screen

The most common use of this function is to forward a review list entry (in other words, to "send" it to another user). The fields on this screen and the rules for their use are described in the section *Creating a Review List Entry.*

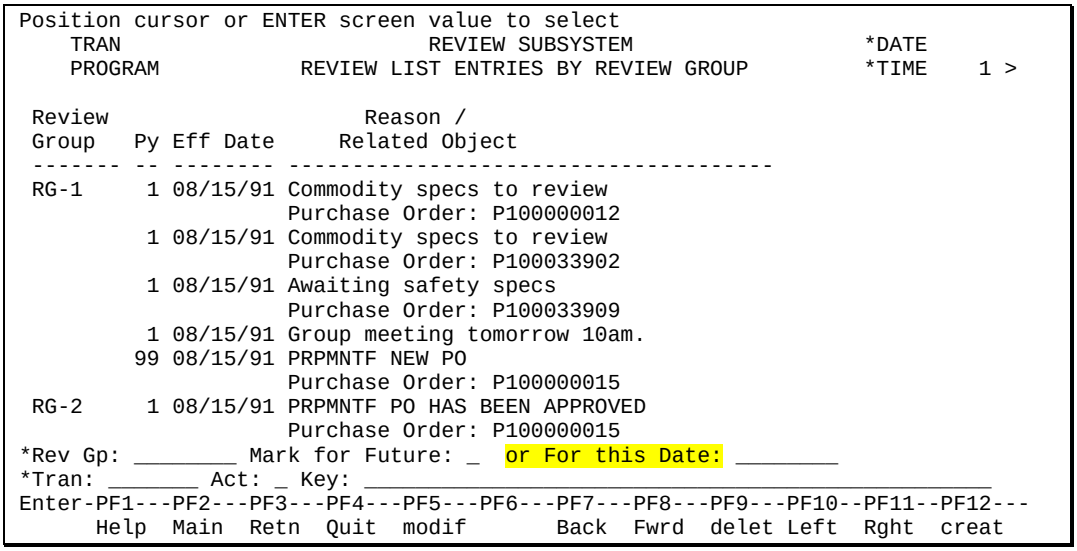
Forwarding a Review List entry

To forward a review list entry, enter the Forward action on the entry you wish to forward. Also you must populate the Forward To User at the bottom of the screen. The review list item will then be automatically forwarded to the user specified.

Technical Note: forwarding a review list item is equivalent to presenting the above screen and changing the User Id. However there is no reason to display the change window, simply update the record.

Review List Entries by Review Group

When you enter '2' at the Review List Submenu, you see the Review List Entries by Review Group screen. At this screen you can see the review list entries that are addressed to a review group. You can add, change, delete, and forward review list entries from this screen, and in some cases select a review list entry for further review.



```
Position cursor or ENTER screen value to select
    TRAN                        REVIEW SUBSYSTEM                  *DATE
    PROGRAM           REVIEW LIST ENTRIES BY REVIEW GROUP         *TIME    1 >

 Review                     Reason /
 Group   Py Eff Date     Related Object
 ------- -- -------- -------------------------------------
 RG-1     1 08/15/91 Commodity specs to review
                     Purchase Order: P100000012
          1 08/15/91 Commodity specs to review
                     Purchase Order: P100033902
          1 08/15/91 Awaiting safety specs
                     Purchase Order: P100033909
          1 08/15/91 Group meeting tomorrow 10am.
         99 08/15/91 PRPMNTF NEW PO
                     Purchase Order: P100000015
 RG-2     1 08/15/91 PRPMNTF PO HAS BEEN APPROVED
                     Purchase Order: P100000015
*Rev Gp: ________ Mark for Future: _  or For this Date: ________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  modif       Back  Fwrd  delet Left  Rght  creat
```

Review List Entries by Review Group - 1

This selection list has two screens. At both screens you can enter a review group to start the scroll. If you don't enter a code and simply press ENTER, the display begins with review list entries addressed to the first review group. You can get active help for review group. The review list entries are displayed in sequence of addressee review group, priority and date. Normally, only review list entries with an effective date that is not greater than the current date are displayed. If you want to see review list entries dated into the future, enter any character in the 'Mark for Future Review List entries' field. To turn this feature off, enter a space in this field. . Enter a specific date to see only those review lists for that review group effective on that date. The date can be in the past, present or future.

On the right of the screen, you see the following:

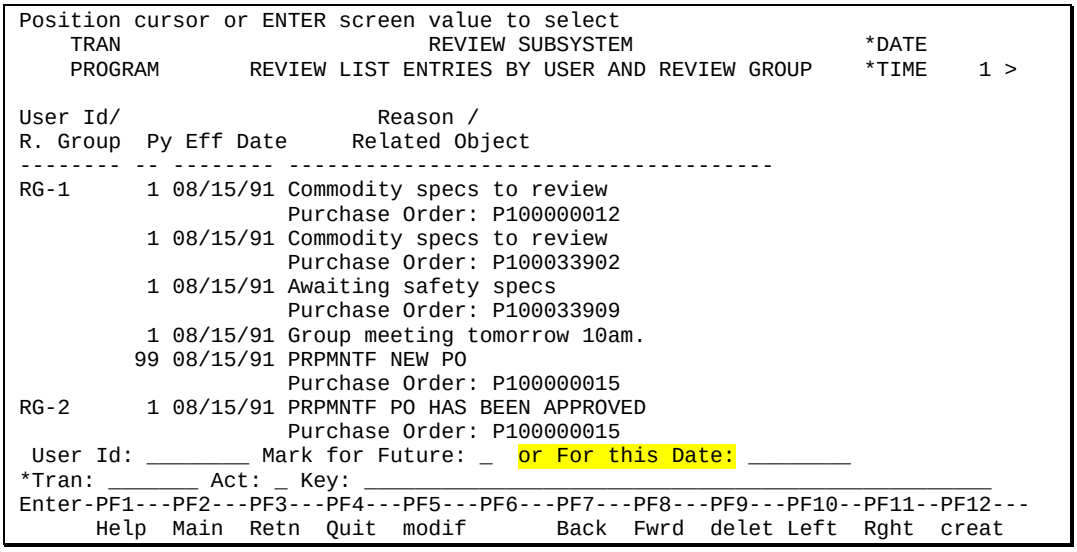
```
Position cursor or ENTER screen value to select
     TRAN                      REVIEW SUBSYSTEM                    *DATE
< 1  PROGRAM              REVIEW LIST ENTRIES BY REVIEW GROUP      *TIME

 Review                                      Delete
 Group   Prty  Eff. Date  No.   Type    With PFKey
-------- ----  ---------  ---  -------  ----------
RG_1       1   Aug 13,90   1   Remind      Yes
           1   Aug 16,90   2   Track       No
           2   Aug 16,90   3   Track       No
           2   Aug 16,90   4   Track       No
           5   Aug 11,90   5   Remind      No
RG-2       1   Aug 12,90   6   Remind      Yes
           1   Aug 13,90   7   Track       No
           1   Aug 15,90   8   Remind      Yes
           1   Aug 15,90   9   Remind      Yes
           1   Aug 15,90  10   Track       Yes
           1   Aug 15,90  11   Track       No
           1   Aug 15,90  12   POA         Yes
           1   Aug 15,90  12   POA         Yes
*User Id: ________ Mark for Future: _  or For this Date: ________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  modif       Back  Fwrd  delet Left  Rght  creat
```

Review List Entries by Review Group - 2

The above screens shows exactly the same information as the Review List Entries by User, described earlier in this chapter, and has the same special review list processing.

Review List Entries for a Specific User

When you enter '3' at the Review List Submenu, you see the Review List Entries for a Specific User screen. At this screen you can see the review list entries that are addressed to a specific user Id and to all the review groups of which that user is a member. You can add, change, delete, and forward review list entries from this screen, and in some cases select a review list entry for further review.



```
Position cursor or ENTER screen value to select
    TRAN                        REVIEW SUBSYSTEM                  *DATE
    PROGRAM       REVIEW LIST ENTRIES BY USER AND REVIEW GROUP     *TIME    1 >

User Id/                    Reason /
R. Group  Py Eff Date     Related Object
-------- -- -------- ------------------------------------
RG-1      1 08/15/91 Commodity specs to review
                     Purchase Order: P100000012
          1 08/15/91 Commodity specs to review
                     Purchase Order: P100033902
          1 08/15/91 Awaiting safety specs
                     Purchase Order: P100033909
          1 08/15/91 Group meeting tomorrow 10am.
         99 08/15/91 PRPMNTF NEW PO
                     Purchase Order: P100000015
RG-2      1 08/15/91 PRPMNTF PO HAS BEEN APPROVED
                     Purchase Order: P100000015
 User Id: ________ Mark for Future: _  or For this Date: ________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  modif       Back  Fwrd  delet Left  Rght  creat
```

Review List Entries for a Specific User - 1

At this screen you must enter a valid User Id to start the scroll. If you don't enter a code and simply press ENTER, the display begins with review list entries addressed to your user Id. You can get active help for user id. The review list entries addressed to the user Id are displayed in sequence of priority and date. Thereafter, the review list entries are displayed for each review group to which the specified users belong. The sequence is addressee review group, priority, effective date. Normally only review list entries with an effective date earlier than the current date are displayed. If you want to see review list entries dated into the future, enter any character in the 'Mark for Future Review List entries' field. To turn this feature off, enter a space in this field. . Enter a specific date to see only those review lists for that user effective on that date. The date can be in the past, present or future.

On the right of the screen, you see the following:

```
Position cursor or ENTER screen value to select
    TRAN                          REVIEW SUBSYSTEM                  *DATE
< 1 PROGRAM       REVIEW LIST ENTRIES BY USER AND REVIEW GROUP       *TIME

User Id/                                      Delete
R. Group Prty  Eff. Date  No.   Type    With PFKey
-------- ----  ---------  ---  -------  ----------
RG-1       1   Aug 13,90   1   Remind      Yes
           1   Aug 16,90   2   Track       No
           2   Aug 16,90   3   Track       No
           2   Aug 16,90   4   Track       No
           5   Aug 11,90   5   Remind      No
RG-2       1   Aug 12,90   6   Remind      Yes
           1   Aug 13,90   7   Track       No
           1   Aug 15,90   8   Remind      Yes
           1   Aug 15,90   9   Remind      Yes
           1   Aug 15,90  10   Track       Yes
           1   Aug 15,90  11   Track       No
           1   Aug 15,90  12   POA         Yes
           1   Aug 15,90  12   POA         Yes
*User Id: ________ Mark for Future: _  or For this Date: ________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  modif       Back  Fwrd  delet Left  Rght  creat
```

Review List Entries for a Specific User - 2

Purge Multiple Review List Entries by User

When you enter '4' at the Review List Menu, you see the Purge Multiple Review List Entries by User screen. At this screen you can see the review list entries that are addressed to a user Id. You can delete and forward review list entries from this screen, selecting multiple at a time. Access to this screen is restricted to a small group of user-ids.

```
Position cursor or ENTER screen value to select
     TRAN                    REVIEW SUBSYSTEM                  *DATE
     PROGRAM    PURGE MULTIPLE REVIEW LIST ENTRIES BY USER      *TIME    1 >

       User                        Reason /
Action Id       Py Eff Date      Related Object
------ -------- -- -------- ---------------------------------------
  _     SPLEF     1 08/12/90  PURCHASE ORDER FOR MORE THAN $10000
  _               1 08/15/90  REVIEW THE PURCHASE ORDER
                               Purchase Order: P123
  _               2 08/15/90  APPROVE THIS PO PLEASE
                              Purchase Order: P123
  _               5 08/15/90  REVIEW THE REQUISITION
                                        Requisition: R1239
  _     SPLJS     1 08/15/90  Look at P123 and call me
  _               1 08/15/90  Error in PRPMNTF PURCHASE ORDER NEEDS AP
                             Purchase Order: P123
                       ****** End of Data ******

*User Id: ________ Mark for Future: _
*Tran: _______ Act: _ Key: ____________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit  modif       Back  Fwrd  delet Left  Rght  creat
```

Review List Entries by User Screen - 1

This selection list has two screens. At both screens you can enter a User Id to start the scroll. If you don't enter a code and simply press ENTER, the display begins with review list entries addressed to your user Id. You can get active help for user Id. The review list entries are displayed in sequence of addressee user id, priority and date. Normally only review list entries with an effective date not greater then the current date are displayed. If you want to see review list entries dated into the future, enter any character in the 'Mark for Future Review List entries' field. To turn this feature off, enter a space in this field.

For each review list entry, you see the following information:

Action
: You can perform the following actions on each review list item:

- Purge - use the ‘P’ action if you want to delete the review list item.
- Modify - use the ‘M’ action if you want to forward the review list to another user.

User Id
: The Id of the designated user.

Py
: Priority of the entry. This is used to sequence the review list entries in some sort of priority over date.

Eff. Date
: The effective date. Normally, only review list entries with an effective date up to the present are displayed. To see entries dated into the future, enter any character in the Mark for Future Entries field.

Reason/ Related Object

This indicates the purpose of the review list entry and what (if anything) it relates to.

On the screen to the right, you see the following:

```
Position cursor or ENTER screen value to select
     TRAN                      REVIEW SUBSYSTEM                    *DATE
< 1  PROGRAM              REVIEW LIST ENTRIES BY USER              *TIME

                                                        Delete
Action  User Id  Prty  Eff. Date      No.   Type     With PFKey
------  -------- ----  -------------  ---  -------  ----------
  -     SPLEF          1   Aug 13,90   1   Remind      Yes
  -                    1   Aug 16,90   2   Track       No
  -                    2   Aug 16,90   3   Track       No
  -                    2   Aug 16,90   4   Track       No
  -                    2   Aug 11,90   5   Remind      No
  -     SPLJS          1   Aug 12,90   6   Remind      Yes
  -                    1   Aug 13,90   7   Track       No
  -                    1   Aug 15,90   8   Remind      Yes
  -                    1   Aug 15,90   9   Remind      Yes
  -                    1   Aug 15,90  10   Track       Yes
  -                    1   Aug 15,90  11   Track       No
  -                    1   Aug 15,90  12   POA         Yes
  -                    1   Aug 15,90  12   POA         Yes
*User Id: ________ Mark for Future: _
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit                Back  Fwrd        Left  Rght
```

Review List Entries by User Screen - 2

Type

This indicates the type of review list entry. Valid types are reminding, tracking, or notification of use of power of attorney.

Delete With PFKey

This indicates whether or not the review list entry will be deleted when you enter the line action Delete.

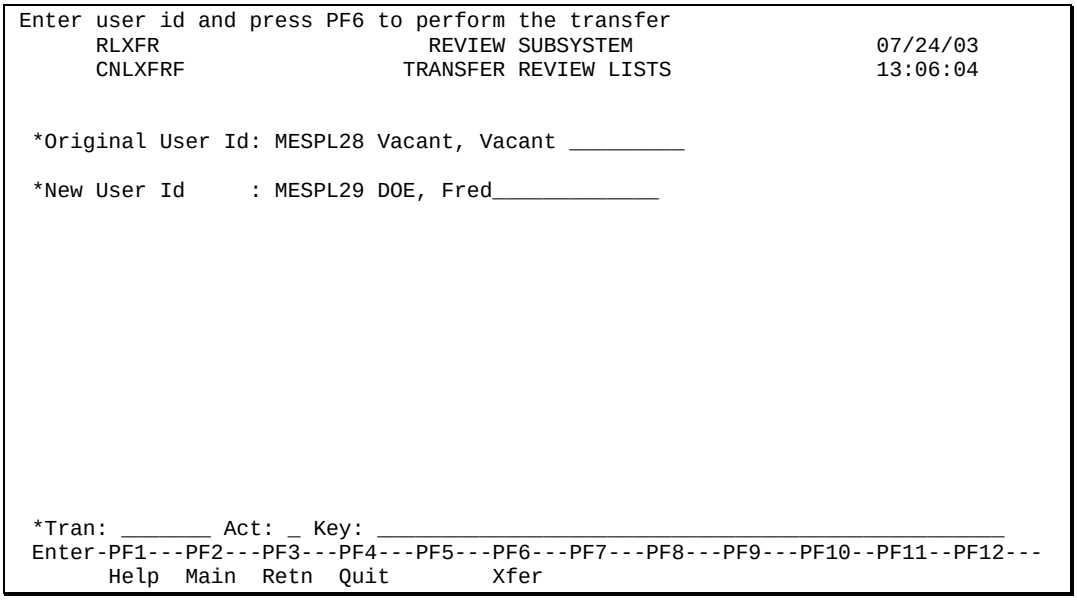
This is a standard selection list screen, but it has a few special PFKeys for review list processing.

Use of PFKeys on the Review List

The standard Pfkeys are available on this screen.

Transfer Review Lists

When you enter '7' at the Review List Submenu, you see the Transfer Review Lists screen. At this screen you can transfer review list entries addressed to one user over to another valid user.



Transfer Review List Screen

Original User Id
: Enter the user id that you want the review list entries forwarded from. The person's name will appear once a valid user id is selected.

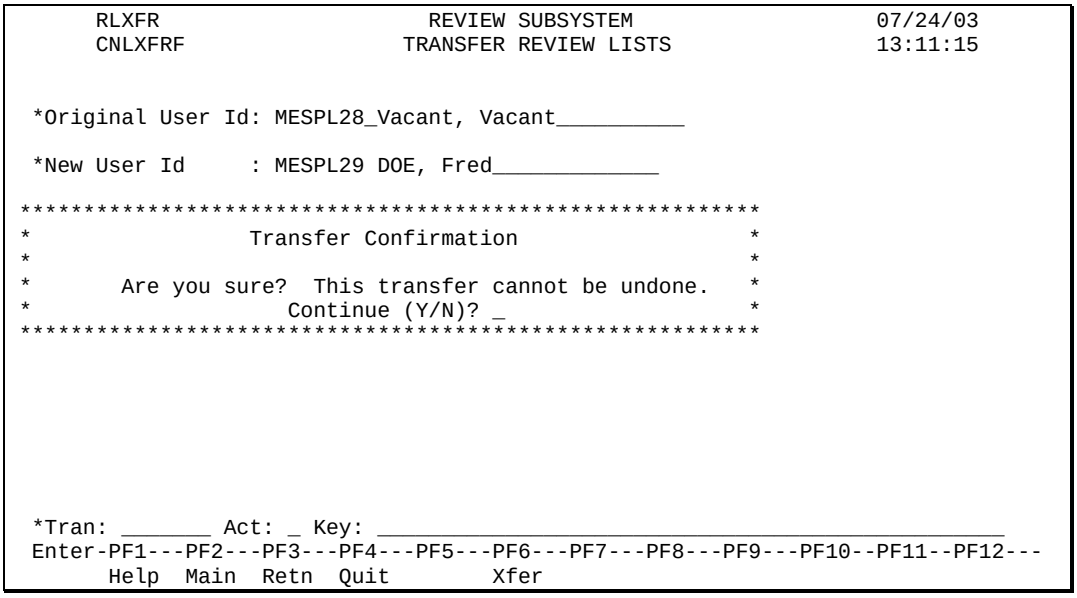
New User Id
: Enter the user id that you want the review list entries forwarded to. The person's name will appear once a valid user id is selected.

Use of PFKeys on the Review List

The standard Pfkeys are available on this screen.

PF6
: Transfer.

When the user presses the Xfer (transfer) PF6 key, the system will popup a window asking the user to confirm the transfer.



The confirmation popup window

When the user enters Y and presses ENTER, the transfer is made.

The screen asks for confirmation because there is not an easy way to undo a transfer made to the wrong User Id. A person would have to manually re-assign the review list entries one-by-one.

This method will transfer ALL of a worker's review list entries to a single user. It will not be possible to pick and choose which workflows are transferred to the new worker, this is an all-or-nothing function.

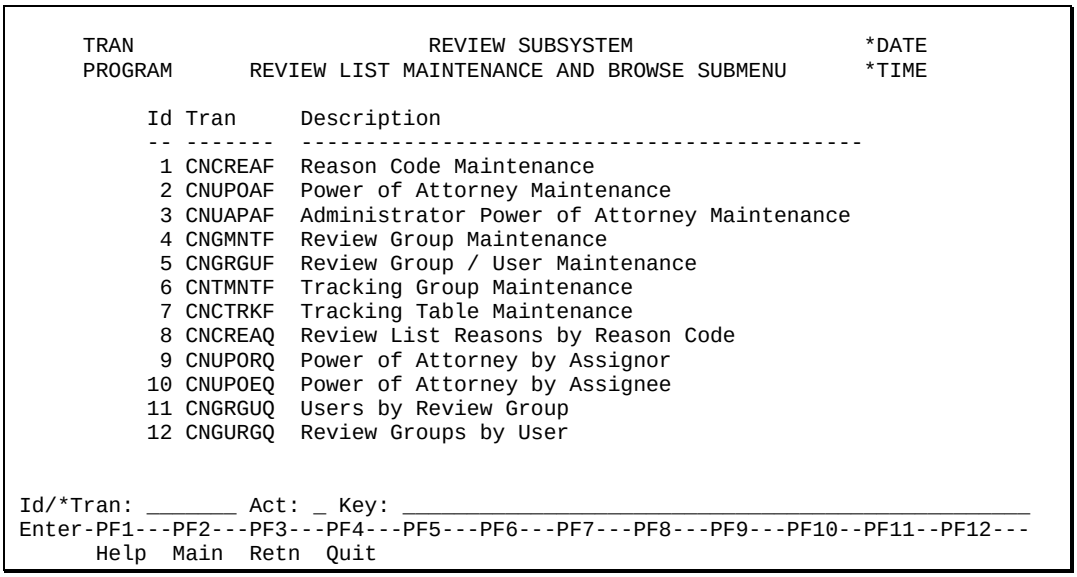
Review *List Maintenance*

There are several files that are used to support the review list subsystem. Some of these are simple code files - namely the reason code, review group, and transaction files. Some represent relationships - namely the user/power of attorney and the review group/user files. The user and transaction files from the Constrain security system are used by the review list subsystem. These files themselves are maintained in the security subsystem. The tracking group and tracking table are described in *Tracking* later in this chapter.

The typical user of these maintenance screens would be the system administrator.

Review List Maintenance And Browse Submenu

When you enter '5' at the Review Subsystem Menu, you see the Review List Maintenance And Browse Submenu.



```
     TRAN                        REVIEW SUBSYSTEM                    *DATE
     PROGRAM      REVIEW LIST MAINTENANCE AND BROWSE SUBMENU         *TIME

          Id Tran     Description
          -- -------  -------------------------------------------
           1 CNCREAF  Reason Code Maintenance
           2 CNUPOAF  Power of Attorney Maintenance
           3 CNUAPAF  Administrator Power of Attorney Maintenance
           4 CNGMNTF  Review Group Maintenance
           5 CNGRGUF  Review Group / User Maintenance
           6 CNTMNTF  Tracking Group Maintenance
           7 CNCTRKF  Tracking Table Maintenance
           8 CNCREAQ  Review List Reasons by Reason Code
           9 CNUPORQ  Power of Attorney by Assignor
          10 CNUPOEQ  Power of Attorney by Assignee
          11 CNGRGUQ  Users by Review Group
          12 CNGURGQ  Review Groups by User


Id/*Tran: _______ Act: _ Key: ____________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit
```

Review List *Maintenance* and Browse Submenu

The Review List Maintenance and List submenu gives you the following options. To choose an option, enter its code in the Id/Tran field and press ENTER.

ID	DESCRIPTION
1	Reason Code Maintenance This option lets you enter, maintain or display reason codes
2	Power Of Attorney Maintenance This option lets a user enter, maintain or display power of attorney relationships for himself.
3	Administrator Power Of Attorney Maintenance This option allows administrators to enter, maintain or display power of attorney relationships for all users.
4	Review Group Maintenance This option lets you enter, maintain or display review groups.
5	Review Group/User Maintenance This option lets you enter, maintain or display information in the review group/user file.
6	Tracking Group Maintenance This option lets you enter, maintain or display tracking group records. This

maintenance screen is described in detail in *'Tracking'* later in this chapter.

7 Tracking Table Maintenance

This option lets you enter, maintain or display tracking tables. This maintenance screen is described in detail in *'Tracking'* later in this chapter.

8 Review List Reasons by Reason Code

This option lets you list reason codes in reason code sequence.

9 Power of Attorney by Assignor

This option lets you see a listing of users and those to whom they have granted power of attorney.

10 Power of Attorney by Assignee

This option lets you see a listing of users and those on whose behalf they can operate (the users for whom they have power of attorney).

11 Users by Review Group

This option lets you see a listing of the review groups and the users that belong to them.

12 Review Groups by User

This option lets you see a listing of the users and the review groups to which they belong.

Reason Code Maintenance Screen

When you enter '1' at the Review List Maintenance Submenu, you see the Reason Code Maintenance screen. The reason code file contains valid reason codes for review list entries. Fields in this file determine the reason or purpose to be displayed and certain default values to be used when creating review list entries which use this reason code.

```
     TRAN                      REVIEW SUBSYSTEM                    *DATE
     PROGRAM                REASON CODE MAINTENANCE                *TIME

*Action              : _ (A,B,C,D,N,M,P)
 REASON CODE         : ________
 Description         : ________________________________________
*Default User Id     : ________
*Default Review Group: ________
*Default Transaction : ________
 Default Priority    : ___
 Help Description    : ____________________________________________________ +









*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit
```

Reason Code Maintenance Screen

The fields on the reason code maintenance screen are described below.

Action
: The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

REASON CODE
: The code that identifies the reason for a review list entry.

System Reason
: **Display Only.** The verbiage “system reason” will be output if the reason “hardcoded” within the system. Note that system reasons cannot be deleted.

Description
: Description of the reason, to be displayed on the review list screen when this code is specified on a review list entry.

Default User Id
: The default user Id, to be used when creating a review list entry with this reason code. If you enter a value in this field, it is validated against the user file. You can enter a value in this field or the next (Review Group), but not both. You can get active help for user Id.

Dafault Review Group
: The default review group to be used when creating a review list entry with this reason code. If you enter a value in this field, it is validated against the review group file. You can enter a value in this field or the previous one (User Id), but not both. You can get active help for review group.

Default Transaction

The default transaction to be used when creating a review list entry with this reason code. This is an optional field, for which you can get active help. The value you enter must be a valid transaction in the transaction file.

Default Priority (N3)

The Default Priority for workflow entries that have this Reason Code. '1' is the highest Priority.

Help Description (A50/1:10)

This is a help text description for this Reason Code that can be invoked using PF1 from the Review List Entries browse screens.

Power of Attorney Maintenance Screen

When you enter '2' at the Review List Maintenance Submenu, you see the Power of Attorney Maintenance screen. You can grant or revoke power of attorney to another user to review your entries. The effect of granting power of attorney is to give another user all your rights and privileges within the review list subsystem, as described in this manual. Conversely, revoking power of attorney immediately takes away all those rights and privileges. You can give power of attorney to more than one user, and you can be granted power of attorney by more than one user.

This is a multi-line relationship, in which the header is defined by the Id of the user who is granting power of attorney (the assignor), and the line is defined by the Id of the user to whom it is granted (the assignee).

```
     TRAN                      REVIEW SUBSYSTEM                      *DATE
     PROGRAM              POWER OF ATTORNEY MAINTENANCE              *TIME

*Action     : _ (C,D,M,N)
*USER ID    : ________
 Start Scroll From User: ________
                                                    Advice of
*POA USER User Name                                 Use (Y/N) *Cmd (A,M,P)
--------- ------------------------------------- --------- ----
________                                            _       _
________                                            _       _
________                                            _       _
________                                            _       _
________                                            _       _
________                                            _       _
________                                            _       _
________                                            _       _
________                                            _       _
________                                            _       _
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit
```

Power of Attorney Maintenance Screen

The Header fields on the Power of Attorney Maintenance screen are described below.

User Id
: The user Id of the assignor. This is the Id of the user on whose behalf the users specified below can operate.

Start Scroll from User
: Enter a user Id to begin the scroll. This is an optional field.

The line fields on the power of attorney maintenance screen are described below.

POA User Id
: The Id of the user to whom power of attorney is granted. The POA user may act on behalf of the user mentioned at the top of the screen. The Id entered must exist on the user file. You can get active help for this field.

User Name
: The name of the POA user displayed from the user file. This field cannot be modified.

Advise of Use
: This indicates whether the user should be advised whenever this POA user exercises this power of attorney. When this field is set to Y, each time this POA user reviews something on behalf of the assignor, the subsystem will create a reminding review list entry addressed to assignor, indicating

this. The reminding review list entry will contain most of the details (reason, related object, effective date, etc.) that the original review list entry contained.

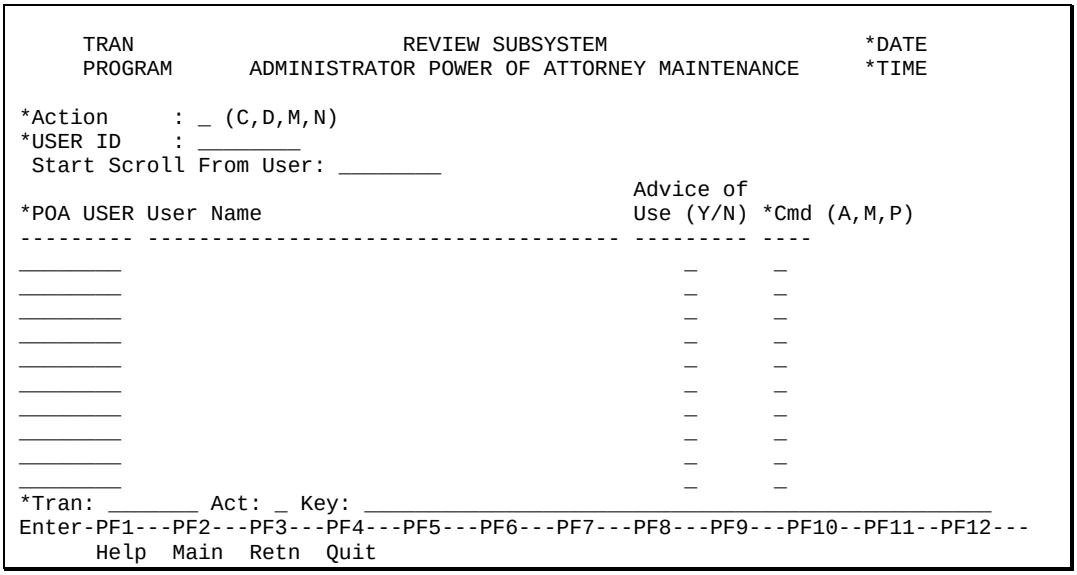
Field Level Security

By default, you can grant/revoke power of attorney for your user id; other people can not give themselves power of attorney over your review lists. However, those users with sufficient security clearance are able to change everyone's else's power of attorney. This is governed by access flag 1 for POA's transaction group. If the value for the user or user group is greater than 0, that users or users in that user group have this capability.

Administrator Power of Attorney Maintenance Screen

When you enter '3' at the Review List Maintenance Submenu, you see the Administrator Power of Attorney Maintenance screen. You can grant or revoke power of attorney to another user to review a persons' entries. The effect of granting power of attorney is to give another user all rights and privileges within the review list subsystem, as described in this manual. Conversely, revoking power of attorney immediately takes away all those rights and privileges. You can give power of attorney to more than one user, and you can be granted power of attorney by more than one user.

This is a multi-line relationship, in which the header is defined by the Id of the user who is granting power of attorney (the assignor), and the line is defined by the Id of the user to whom it is granted (the assignee).



Administrator Power of Attorney Maintenance Screen

The Header fields on the Power of Attorney Maintenance screen are described below.

User Id
: This is the Id of the user on whose behalf the users specified below can operate.

Start Scroll from User
: Enter a user Id to begin the scroll. This is an optional field.

The line fields on the power of attorney maintenance screen are described below.

POA User Id
: The Id of the user to whom power of attorney is granted. The POA user may act on behalf of the user mentioned at the top of the screen. The Id entered must exist on the user file. You can get active help for this field.

User Name
: The name of the POA user displayed from the user file. This field cannot be modified.

Advise of Use
: This indicates whether the user should be advised whenever this POA user exercises this power of attorney. When this field is set to Y, each time this POA user reviews something on behalf of the assignor, the subsystem will create a reminding review list entry addressed to assignor, indicating

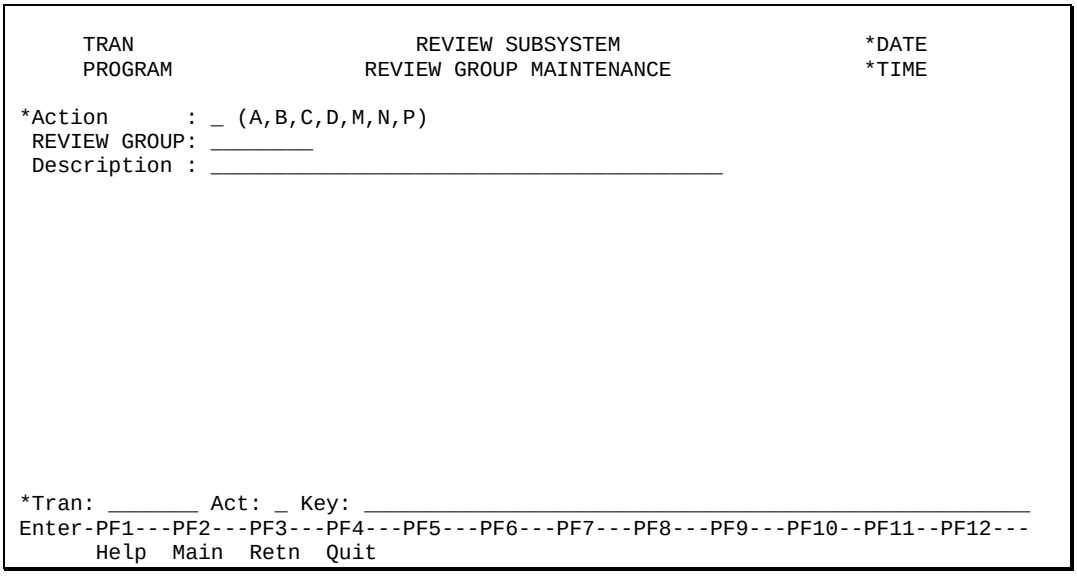
this. The reminding review list entry will contain most of the details (reason, related object, effective date, etc.) that the original review list entry contained.

Field Level Security

Only the administrators can grant/revoke power of attorney for your user id; other people can not give themselves power of attorney over your review lists. However, those users with sufficient security clearance are able to change everyone's else's power of attorney.

Review Group Maintenance Screen

When you enter '4' at the Review List Maintenance Submenu, you see the Review Group Maintenance screen. This screen gives you access to the review group file. A review group is used to define a variable set of users who have a specific capability in terms of the review list subsystem. This provides the mechanism to address a review list entry to a group of users, any one of whom can take action on it. This is useful when there is a 'pool' of users who have equivalent capability in terms of the review subsystem (e.g., a buyer pool).



Review Group Maintenance Screen

The fields on the review group maintenance screen are described below.

Action

The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

Review Group

The code that identifies the review group.

Description

The name or description of the group.

Review Group/User Maintenance Screen

When you enter '5' at the Review List Maintenance Submenu, you see the Review Group/User Maintenance screen. At this screen you can define which users make up a review group. A review group can include any number of users.

This is a multi-line relationship, in which the header is defined by the review group displayed at the top of the screen, and the line is defined by the user id who belongs to the group.

```
     TRAN                        REVIEW SUBSYSTEM                    *DATE
     PROGRAM              REVIEW GROUP / USER MAINTENANCE            *TIME

*Action      : _ (C,D,M,N)
*REVIEW GROUP: ________
 Start Scroll From User: ________

*USER ID User Name                              *Cmd (A,M,P)
-------- ------------------------------------ ---
________                                        _
________                                        _
________                                        _
________                                        _
________                                        _
________                                        _
________                                        _
________                                        _
________                                        _
________                                        _
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit
```

Review Group/User Maintenance Screen

The fields on the Review Group/User Maintenance screen are described below.

Action
The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.Review Group

The Id of the review group.

Start Scroll From User

Enter a user Id to begin the scroll. This is an optional field.

User Name

The name of the user is displayed from the user file. This field cannot be modified.

USER ID

This identifies the user as belonging to this review group. The Id entered must exist on the user file. You can get active help for this field.

Review List Reasons by Reason Code

When you enter '8' at the Review List Browse Submenu, you see the Review List Reasons by Reason Code.

```
    TRAN                     REVIEW SUBSYSTEM                     *DATE
    PROGRAM           REVIEW LIST REASONS BY REASON CODE          *TIME

 Reason                                        Default    Default    Default
  Code               Description               User Id  Review Group  Trans
--------  ------------------------------------ -------- ------------ -------
COMAPP    COMMODITY WAITING FOR APPROVAL       SPLBB
COMSPAPP  COMMODITY IN SPECIFICATION PROCESS   SPLBB
COMSPRCT  COMMODITY SPECIFICATION REJECTED     SPLBB
POAMT     PURCHASE ORDER FOR MORE THAN $10000  DEVMW                POMAINT
POAPP     PURCHASE ORDER WAITING FOR APPROVAL  SPLBB
PONTRCV1  PO OVER ONE MONTH WITH NO RECEIPTS   DEVMW                POMAINT
PONTRCV2  PO OVER TWO MONTHS WITH NO RECEIPTS  DEVMW                POMAINT
POVADCHG  PO VENDOR ADDRESS CHANGED AFTER APPR           VENVERGP
REQAMT    REQUISITION FOR MORE THAN $1000      SPLBB
REQAPP    REQUEST WAITING FOR APPROVAL         SPLBB
REQQTY    REQUISITION FOR MORE THAN 10000 UNIT DEVMW                POMAINT
REQQTYCH  REQUISITION QUANTITY CHANGE AFTER AP           COMRVIEW
Reason Code: ________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit              Back  Fwrd
```

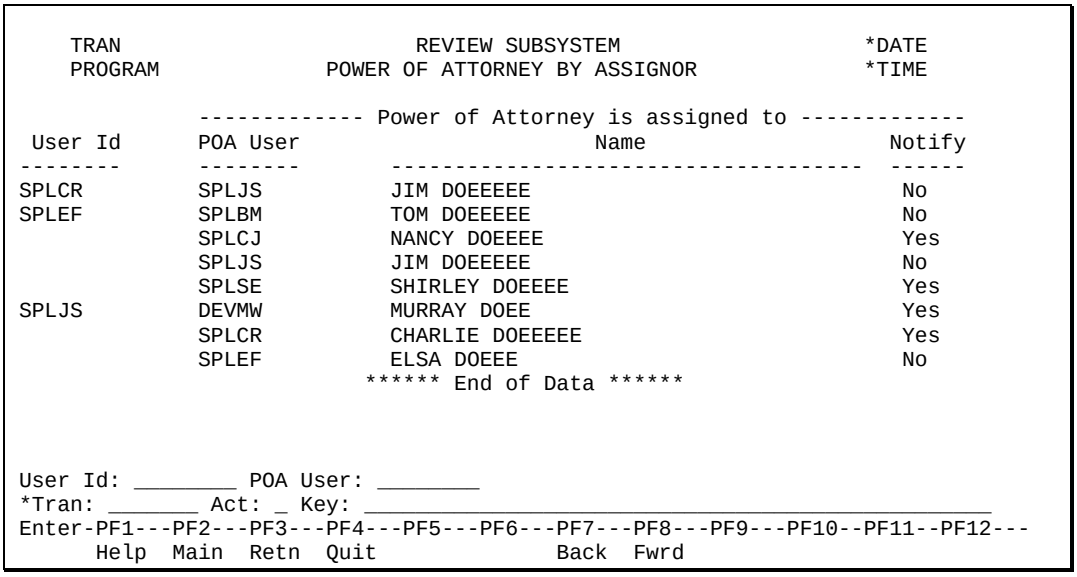
Review List Reasons by Reason Code

This screen lets you enter a reason code to begin the scroll. The screen displays reason codes in alphanumeric order and details of the reason code. If you don't enter a reason code and simply press ENTER, the display begins with the first reason code.

See *Reason Code Maintenance Screen* for an explanation of the information displayed on the list screen.

Power of Attorney by Assignor

When you enter '9' at the Review List Browse Submenu, you see the screen that lets you list the user/power of attorney relationship by assignor. This shows a list of all the users who have been granted power of attorney to act on behalf of a specific user. The list by assignor is shown below.



```
    TRAN                        REVIEW SUBSYSTEM                    *DATE
    PROGRAM              POWER OF ATTORNEY BY ASSIGNOR              *TIME

              ------------- Power of Attorney is assigned to -------------
 User Id      POA User                      Name                     Notify
--------      --------       -----------------------------------    ------
SPLCR         SPLJS           JIM DOEEEEE                            No
SPLEF         SPLBM           TOM DOEEEEE                            No
              SPLCJ           NANCY DOEEEE                           Yes
              SPLJS           JIM DOEEEEE                            No
              SPLSE           SHIRLEY DOEEEE                         Yes
SPLJS         DEVMW           MURRAY DOEE                            Yes
              SPLCR           CHARLIE DOEEEEE                        Yes
              SPLEF           ELSA DOEEE                             No
                            ****** End of Data ******



User Id: ________ POA User: ________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit             Back  Fwrd
```

Power of Attorney by Assignor

This screen lets you enter a User Id, POA User Id or both to begin the scroll. The User Id takes precedence over the POA User in positioning the scroll. If you don't enter anything, and simply press ENTER, the display begins with the first User Id/POA User.

The screen displays user Ids in alphanumeric order and lists for each one, the Ids of other users for whom he holds power of attorney.

See *Power of Attorney Maintenance Screen* for an explanation of the information displayed.

Power of Attorney by Assignee

When you enter '10' at the Review List Browse Submenu, you see the screen that lets you list the power of attorney relationship by assignee.

```
    TRAN                        REVIEW SUBSYSTEM                    *DATE
    PROGRAM              POWER OF ATTORNEY BY ASSIGNEE              *TIME

               -------------- Can act on behalf of---------------
POA User       User Id                      Name
--------       --------     ------------------------------------
DEVMW          SPLJS          JIM DOE
SPLBM          SPLEF          ELSA DOE
SPLCJ          SPLEF          ELSA DOE
SPLCR          SPLJS          JIM DOE
SPLEF          SPLJS          JIM DOE
SPLJS          SPLCR          CHARLIE DOE
               SPLEF          ELSA DOE
SPLSE          SPLEF          ELSA DOE
                             ****** End of Data ******



POA User: ________ User Id: ________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit              Back  Fwrd
```

Power of Attorney by Assignee

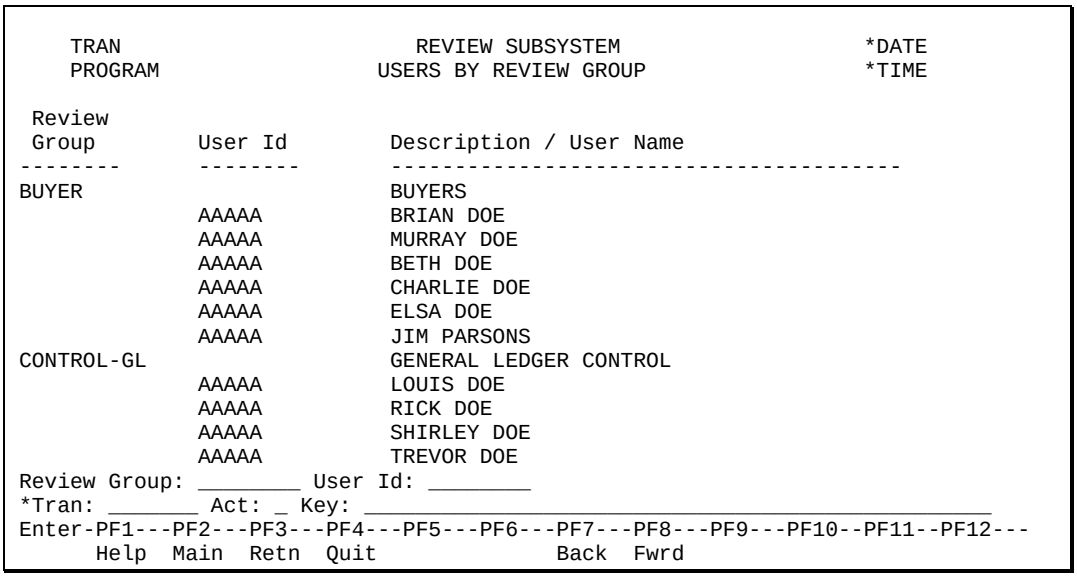
This screen lets you enter a POA User Id, a User Id or both to begin the scroll. The POA User takes precedence over the User Id in positioning the scroll. If you don't enter a user Id and simply press ENTER, the display begins with the first POA user Id (assignee).

The screen displays user Ids in alphanumeric order and lists for each one, the Ids of other users for whom he holds power of attorney.

See *Power of Attorney Maintenance Screen* for an explanation of the information displayed.

Users by Review Group

When you enter '11' at the Review List Browse Submenu, you browse through the review group/user relationship, to see a listing of review groups and the uses that belong to them.



```
    TRAN                        REVIEW SUBSYSTEM                    *DATE
    PROGRAM                   USERS BY REVIEW GROUP                 *TIME

 Review
 Group         User Id         Description / User Name
--------       --------        ---------------------------------------
BUYER                          BUYERS
               AAAAA           BRIAN DOE
               AAAAA           MURRAY DOE
               AAAAA           BETH DOE
               AAAAA           CHARLIE DOE
               AAAAA           ELSA DOE
               AAAAA           JIM PARSONS
CONTROL-GL                     GENERAL LEDGER CONTROL
               AAAAA           LOUIS DOE
               AAAAA           RICK DOE
               AAAAA           SHIRLEY DOE
               AAAAA           TREVOR DOE
Review Group: ________ User Id: ________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit              Back  Fwrd
```

Users by Review Group

This screen lets you enter a review group or User Id or both to begin the scroll. The review group takes precedence over the User Id in positioning the scroll. If you don't enter anything and simply press ENTER, the display begins with the first review group.

The screen displays Review Groups in alphanumeric order listing for each one, the users who belong to the group.

See *Review Group/User Maintenance Screen* for an explanation of the information displayed on the list screen.

Review Groups by User

When you enter '12' at the Review List Browse Submenu, you see the screen that lets you browse through the review group/user relationship to see a listing of users and the review groups to which they belong.

```
    TRAN                        REVIEW SUBSYSTEM                    *DATE
    PROGRAM                  REVIEW GROUPS BY USER                  *TIME

               Review
User Id        Group          User Name / Description
--------       --------       ----------------------------------------
UA#9999                       DOE,CHARLIE
               AAAAA          SPL SAN FRANCISCO REVIEW GROUP
UA#9999                       DOE,HWAYOUNG
               AAAA           ELENA'S REVIEW GROUP
UA#9999                       DOE,ELSA
               AAAAA          SPL SAN FRANCISCO REVIEW GROUP
UA#9999                       DOE,JENNIFER
               AAAA           ELENA'S REVIEW GROUP
UA#9999                       DOE,ELENA X
               AAAA           ELENA'S REVIEW GROUP
UA#9999                       DOE,KAREN
               AAAAA          SPL SAN FRANCISCO REVIEW GROUP
                           ****** End of Data ******
User Id: ________ Review Group: ________
Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit              Back  Fwrd
```

Review Groups by User

This screen lets you enter a user Id, review group or both to begin the scroll. The user id takes precedence over the review group in positioning the scroll. If you don't enter a user Id and simply press ENTER, the display begins with the first user Id. The screen displays user Ids in alphanumeric order listing for each one, the review groups to which he belongs.

See *Review Group/User Maintenance Screen* for an explanation of the information displayed.

Tracking

Introduction

There is sometimes a need in a system to monitor the progress of a document or process. For example, when a month end procedure is commenced, it may be useful to advise several users of this fact, as well as keeping them informed as the month close progresses. As another example, a document in the system can have many statuses, and as each status is reached, certain users must be advised. Not only are they advised, but some may be required to act upon the document in order to move it along to the next status, and the subsystem must keep reminding them that their input is required. The use of the review subsystem to achieve this is termed 'Tracking'.

Tracking requires the definition of a 'Tracking Group' which identifies what the system is monitoring. In the examples above, the tracking groups would be a month close indicator and a document status respectively. In addition, tracking requires the specification of what is to be done when the tracking group attains a certain value. This is done using the tracking table, which details what review list entries are to be created. Lastly, tracking requires that the application program invoke the tracking table processor (with the appropriate data) when any field that is being tracked changes value. This is the only part of the tracking process that is custom written.

The actual procedure of tracking is best illustrated by an example. Suppose a document can attain the following statuses during its life cycle:

U	Unfinished
F	Finished
I	Approved by buyer
Z	Awaiting Authorization
R	Approval rejected
A	Authorized

The reasons that the document appears on the various review lists might be as follows:

INP	In progress
FIN	Entry complete, beginning approval phase
SUP	Awaiting buyer's approval
RLA	Awaiting review/approval
REJ	Rejected in approval phase
APP	Awaiting final approval and authorization

The tracking group for this example might be REQST (a code for requisition status). Each of the above statuses is a tracking value. The tracking table entries for REQST could be as follows:

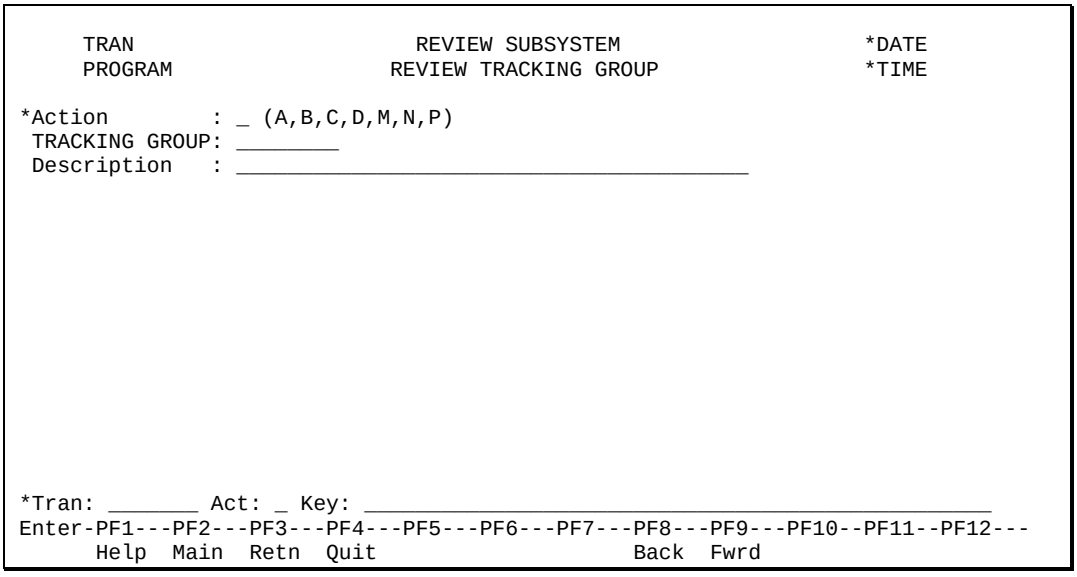
Track Group	Value	Reason Code	User ID	Review Group	Transaction	Delete Sel Value	
REQST	U	INP	AE8900		REQMAINT	N	Y
	F	FIN	AE89012			Y	N
	F	SUP		BUYERS	REQMAINT	N	Y
	I	RLA		CONADM	REQMAINT	N	Y
	Z	APP	AE99999		REQAUTH	N	Y
	R	REJ	AE89012			Y	Y

The tracking table is interpreted as follows:

When a requisition is in status of U (Unfinished), a review list entry is addressed to AE89000 (the user who is entering the requisition). When the status changes to F (Finished), the review list entry addressed to AE89000 is deleted and new ones are created, one addressed to AE89012 (the requisitioner), and one to the group BUYERS. From there as its status changes it will be sent for review to Contract Administration (group CONADM), and finally, for authorization to user AE999999. If it is at any time rejected, the requisitioner (user AE89012) is advised. If at any time, the status is changed back to a previous status, the requisition will once again appear on the review list of the user or group responsible for the document in that status. Note that there is no tracking table entry for the status of A (Authorized), as the users are no longer required to take any action on it and no review list entries are created.

Tracking Group Maintenance Screen

When you enter '6' at the Review List Maintenance Submenu, you see the Tracking Group Maintenance screen. At this screen you can identify all entities that are tracked via the review list subsystem. A tracking entity can be a single field or a combination of several fields. You use this maintenance transaction in conjunction with the Tracking Table to indicate what should be done when a tracking group attains a specific value. Some examples of tracking groups are the status field on a requisition, the 'Do not pay' indicator on a vendor record, the 'month open' tag on a run control file.



Tracking Group Maintenance Screen

The fields on the tracking group maintenance screen are described below.

Action
: The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

Tracking Group
: The code that identifies the entity being tracked.

Description
: The description of the tracking group.

Tracking Table Maintenance Screen

When you enter '7' at the Review List Maintenance Submenu, you see the Tracking Table Maintenance screen. At the screen you can specify the review list entry that is to be created when a tracking group attains a particular value. For example, you can set up the tracking table so that when the status field of any requisition is set to the value Finished, a review list entry is created for the chief buyer and another for the budget review group.

```
     TRAN                        REVIEW SUBSYSTEM                  *DATE
     PROGRAM                  TRACKING TABLE MAINTENANCE           *TIME

*Action                            : _ (A,B,C,D,M,N,P)
*TRACKING GROUP                    : _______
 Tracking Field Value              : ____________________________________________
 Sequence Number                   : ___


*Reason Code                       : _________
 Purpose                           : ____________________________________________
*User Id                           : _________
*Review Group                      : _________
 Priority                          : ___
 Number of Changes                 : ___
*Transaction                       : ________
 Delete if Tracking Value Changes  : _  (Y/N)
 Allow Manual Delete               : _  (Y/N)


*Tran: ________ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit                    Back  Fwrd
```

Tracking Table Maintenance Screen

The fields on the tracking table maintenance screen are described below.

Action
: The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

Tracking Group
: This defines the entity that is being tracked and must be a valid tracking group.

Tracking Field Value
: The value that the entity must attain to trigger the creation of the review list entry specified.

Sequence Number
: Used when more than one review list entry is to be created when this value is attained.

Review list details:

FIELD	DESCRIPTION
Reason Code &Purpose	These fields are optional and you can enter either a reason code or purpose, but not both. Reason code must be valid. You can get active help for reason code.
User Id & Review Group	These are both optional and you can enter either a user Id or a review group, but not both. The code entered must be valid. You can get active help for user Id and for review group. If nothing is entered in these fields, the system uses the default user Id or review group contained on the reason code table.
Priority	This is used to sequence the review list entries in some priority over date. It is optional and a default value of '1' is assigned.
Number of Days	Generally, the effective date will be the date the review list entry was created. To defer the appearance of the entry, specify the number of days to be added to the current date for calculating the effective date.
Transaction	The transaction to be invoked when the entry is selected. This is an optional field for which you can get active help.
Delete Flags	These indicate when the entry can be deleted from the review list file. You can indicate whether it should be deleted upon selection or when there is a change in the value of the tracking group.

Once the entries in the tracking table have been set up, application programs must invoke the tracking table processor to generate the specified review list entries. This is detailed in the Application Programmer's Guide for Review List.

Technical note: review list needs some cleanup:

The LDA (CNCRCDL) will be used to:

- *store all necessary reasons during the load (can you think of a way to load the tracking items?). Also populate the system descriptions.*
- *As reference where the reason is used (within application logic).*

Application Programmer's Guide

Introduction

The Review List system is designed to allow the system or user to pass documents (i.e., records) on-line to other users, and to provide a prompting service (by date and time). The Review List system could be regarded as an on-line in-basket.
For more information on Review List terms and characteristics please see sections Glossary of Terms and Review List Characteristics (p. 2-3).
The following review list functions are available to application programs:

- *Store tracking item*
- *Store reminding item*
- *Delete an item.*
- *Modify an item.*

There is a separate subprogram to perform each of the above.

Updating the Review List

All review list entries are maintained via subprograms, which receive data via the LDA named CNXCREL. This LDA contains all the fields that make up a review list entry. The data passed is detailed below:

Fields on a Review List entry

FIELD	*DESCRIPTION*
Source (A2)	*This indicates the nature of the RL entry. Valid values are T (tracking) P (advice of use of power of attorney) and R (reminder).*
Surrogate Key (N9)	*An internally assigned sequence number to uniquely identify RL entries. Note that this may be used by applications to record the Id of the review list entry, in case future retrieval may be required.*
Related Object (A40)	*If this RL entry relates to something elsewhere, then this field contains the Id of that object. This can be a concatenation of fields. Typically this field would be a means of further identifying the RL entry, but also an indication of the data that will be passed to the transaction that will be invoked when this RL entry is selected.*
Object - Display format (A40)	*The same information as held in the related object, specially formatted so that the user can interpret it when it is displayed on the review list screen.*
Stack Data (A40)	*Data that needs to be placed in the stack in order to call the specified transaction and have the appropriate data displayed. Typically this might include an action code and the Id from the related object, specially formatted for the stack.*
Effective Date (N8)	*The entry only appears on the review list on and after this date. Format YYYYMMDD.*
Addressee	*This will be either a User or a Review Group, it cannot be both.*

FIELD	*DESCRIPTION*
User Id (A8)	*The Id of the user to whom this RL entry is addressed.*
Review Group (A8)	*The Id of the review group to which this RL entry is addressed*
Priority (N3)	*Used to sequence the review list entries into some sort of priority over date.*
Reason Code (A8)	*A code indicating the reason for this review list entry.*
Purpose/Verbiage (A40)	*Narrative that can be used to describe the reason for the review list entry.*
Transaction (A8)	*The identity of the transaction that will be invoked if this entry is selected for review.*
Delete with PFKey (A1)	*This indicates whether the RL entry is deleted when the user positions the cursor on the entry and presses the 'delete' PFKey.*
Delete when TFV changes (A1)	*This indicates whether the RL entry is deleted when the value of a related tracking field changes.*
Tracking Group (A6)	*This is present only on those review list entries that arise from tracking table entries, and identifies the field being tracked. It is used to find the review list entry for subsequent deletion.*

The LDA CNXCREL contains all these fields, and is listed on the following pages.

Subprograms that Update the Review List

Any application program may need to create, modify or delete review list entries. All updating of the review list files should be done using the following subprograms:

CNXCREN	*This stores a new review list entry using data supplied in the parameter data area.*
CNXMODN	*This updates a specific review list entry with the data supplied in the parameter data area.*
CNXDELN	*This deletes a specific review list entry.*
CNLCREN	*This allows the user to enter review list data, which is validated before the new entry is stored.*

An application may need to create 'Tracking' type review list entries, when the value of a specific field changes. This is done using the following subprogram:

CNTTFVN	*This checks for entries in the tracking table and deletes and creates the necessary review list entries.*

When review list data is passed to these subprograms, it is expected as a single parameter. The LDA (CNXCREL) contains all the required fields.

```
01 #RL-ENTRY
  02 SOURCE
  02 SURROGATE-KEY
  02 RELATED-OBJECT
  02 REL-OBJ-DISPLAY
  02 STACK-DATA
  02 EFF-DATE
  02 RL-USER-ID
  02 RL-REVIEW-GROUP
  02 RL-PRIORITY
  02 RL-VERBIAGE
  02 RL-TRANSACTION
  02 RL-TEXT-NAME
  02 RL-TEXT-TYPE
  02 DEL-WITH-PFKEY
  02 DEL-TFV-CHANGE
  02 RL-REASON-CODE
  02 RL-TRACKING-GROUP
```

Using CNXCREN - to create review list entry

This subprogram can be used whenever an application needs to create a review list entry. The programmer must supply the data for the review list as specified below. It is important to note that the subprogram will not return any error codes and will always create a review list entry. If any invalid data is passed, the name of the calling program and an error number are put into the purpose field to record the source of the error, and the review list assigned the necessary default values.

The calling syntax is:

CALLNAT 'CNXCREN' #PROGRAM #RL-ENTRY

where #PROGRAM contains the name of the application that does the call and #RL-ENTRY contains the fields as described in the LDA CNXCREL

It is the responsibility of the application to issue an ET (or BT) at a time that is appropriate to its processing.

The subprogram will validate the data passed and apply defaults as indicated:

1. *Reason code or purpose must be supplied. Reason code must be on file.*
2. *An addressee is required - either user Id or review group can be supplied, never both. The data supplied must be valid. If no data is supplied, the subprogram will look up the reason code (if present) and use the default addressee from there. If no default can be found, it will assign a user id of '99999999', but will indicate that bad data was passed.*
3. *Priority is optional. The default is '1'.*
4. *Effective date is optional. The default is current date.*
5. *Transaction is optional, but must be valid. If no transaction is supplied, the subprogram checks for a default from the reason code file. If there is data that should be stacked into the transaction when the review list is selected, the application can set this up (see note 6). If, for any reason, the transaction code written to the review list is different from the transaction set up in the parameter data area, the stack data is reset.*
6. *Related Object. This has three forms and all are optional. All three forms should carry the same basic information i.e. the Id of whatever the review list relates to.*

   The normal form can be used programmatically to identify the review list entry.

   The display form appears wherever the review list is shown, and serves to assist the user in identifying the purpose of the review list entry. It should be edited in such a way that the information is meaningful to the user.

   The stack version should only contain data when there is a transaction associated with the review list and the programmer should make certain that the format is correct. If, for any reason, the transaction code written to the review list is different from the transaction set up in the parameter data area, the stack data is cleared, but the other related objects are left intact.
7. *Source is optional. The default is 'R'.*
8. *Delete flags are optional, but cannot both be 'N'. The default for 'Delete with PFKey' is 'Y', and 'Delete when TFV changes' is 'N'.*
9. *Tracking group is optional and is not validated, as the use of this field is determined by the application.*

Using CNXMODN - to modify a review list entry

This subprogram can be used whenever an application needs to modify a review list entry. The programmer must supply the surrogate key of the review list entry (see note below on finding the surrogate key), as well as the data for the review list. The subprogram replaces the review list entry with the data in the PDA, so the data passed must be a complete view of the review list entry. It is important to note that the subprogram will not return any error codes, and will update the review list entry unless it has been deleted. If any invalid data is passed, the name of the calling program and an error message are put into the purpose field to record the source of the error and the review list assigned the necessary default values.

The calling syntax is:

```
CALLNAT 'CNXMODN'  #PROGRAM  #RL-ENTRY
```

where #PROGRAM contains the name of the application that does the call, #RL-ENTRY contains the fields as described in the LDA CNXCREL including the key (SURROGATE-KEY) containing the Id of the record to be updated.

It is the responsibility of the application to issue an ET (or BT) at a time that is appropriate to its processing.

The subprogram will validate the data passed and apply defaults in the same way as indicated in the use of the subprogram CNXCREN.

Using CNXDELN - to delete a review list entry

This subprogram can be used whenever an application needs to delete a review list entry. The programmer must supply the surrogate key of the review list entry (see note below).

The calling syntax is:

CALLNAT 'CNXDELN' #SURROGATE-KEY

where #SURROGATE-KEY contains the Id of the record to be deleted.

It is the responsibility of the application to issue an ET (or BT) at a time that is appropriate to its processing.

The subprogram does no validation except to check that the record exists before deleting it. No error codes are returned, and there is no confirmation of the delete.

Finding the Surrogate Key of the review list entry.

There are several ways the application programmer can find the surrogate key of the required review list entry before calling the CNXMODN or CNXDELN.

The following are descriptors on the file:

Reason code

User Id

Review group

The following indices have been created:

User Id - Priority - Eff.date

Review group - Priority - Eff.date

Tracking Group - Related Object - Del. when TFV changes

None of these provide a unique key.

If the application is responsible for creating the review list, and subsequently modifying or deleting it, the application can store the surrogate key for future retrieval of the RL record (it is available in the PDA upon return). If this is not practical the application can consider setting up type, tracking group or related object in such a way that the record is uniquely identifiable using the indices mentioned above.

Using CNTTFVN - the tracking table processor

The description of this subprogram assumes that the application programmer has some knowledge of tracking in the review list subsystem. Please refer to the section on tracking in the document Review List Subsystem.

This subprogram deletes and creates review list entries for a specific object. It is invoked when the application program detects a change in the value of a field for which tracking is required. The call to this subprogram will be written in conjunction with appropriate entries in the tracking group and tracking table files. It is important to note that the subprogram will not return any error codes, and will always delete and create the necessary review list entries. If any invalid data is passed, the name of the calling program and an error message are put into the purpose field to record the source of the error, and the review list assigned the necessary default values.

The calling syntax is:

CALLNAT 'CNTTFVN' #TRACKING-GROUP

#NEW-VALUE

#RELATED-OBJECT

#PROGRAM

#RL-ENTRY

where #TRACKING-GROUP (A6) identifies the tracking group defining the field(s) which are being tracked; #NEW-VALUE (A40) contains the new value of the field(s); #RELATED-OBJECT contains the Id of the related object: #PROGRAM contains the name of the application that does the call, #RL-ENTRY contains a 'skeleton' review list using the fields as described in the LDA CNXCREL.

~~The subprogram does its own ET's simply because it may delete and create many records. Hence the application must place the CALLNAT appropriately~~.

The 'skeleton' RL entry provides default values that can be used for every review list entry created from the call to the tracking table processor. It can contain no data at all, except for the normal and display form of related object. These should be present as it is the related object data that associates a review list entry with a specific object (such as a document Id).

The subprogram does the following:

1. *Deletes any review list entries that were generated for the related object by the tracking table processor for this tracking group (i.e., it deletes the 'tracking' entries created for the related object for the previous value of the tracking group).*
2. *Determines what (if any) review list entries to create by checking the tracking table for this tracking group + value combination. For each entry in the tracking table a review list entry is created as follows:*
    a. *Reason code and purpose: Set up from the tracking table detail if present. If not it takes the default from the skeleton.*
    b. *An addressee is required - either user Id or review group is taken from the tracking table detail if present. If not, it takes the default addressee from the reason code used above. If this is not available, it takes the default from the skeleton.*
    c. *Priority is taken from the tracking table detail or the skeleton.*

d. Effective date is calculated by adding the No. of days specified in the tracking table detail to the current date. If the number of days was not specified, it takes the effective date from the skeleton.

e. Transaction is taken from the tracking table detail, if present. If not, it takes the default transaction Id from the reason code used above. If this is not available, it takes the default from the skeleton. See the note below on the stack data and transaction.

f. Related Object. This is set up using the data in the skeleton as follows:

The normal form - taken from normal form of Related Object in the skeleton.

The display form - taken from the display from in the skeleton. The application must set this up so that it is meaningful to the user.

The stack data - taken from the stack form of the related object in the skeleton. The application must set this up so that what is put into the stack is correctly formatted for the transaction which is to be invoked. Because the transaction Id can be determined from a number of different places, the application cannot be certain of the appropriate format. To ensure that nonsense is not stacked, the assumption is made that the application can only set up the stack form of related data if the associated transaction is the application itself. If the program for the transaction Id determined above is NOT = #PROGRAM, the stack data is reset. Otherwise it is taken directly from the skeleton. See note below on an alternative to this.

g. Type (source) is always 'T'.

h. Delete flags are taken from the tracking table detail if present. If not the defaults are 'N' for Del-upon-Sel, and 'Y' for Del-when-TFV-changes.

i. Tracking group is taken from #TRACKING-GROUP.

3. This data is passed to the standard create subprogram (CNXCREN) where it is validated and further defaults are assigned if necessary.

Note that if the application wants to create a 'tracking' review list entry that will invoke a transaction other than itself with variable stacked data, the application must do a call directly to CNXCREN rather than using the tracking table processor. The application can format the stack data appropriately and set up the review list entry data exactly as if it had been generated from this subprogram. In this way, the review list entry will be deleted upon the next call to the tracking table processor when the tracking value on the related object changes.

Review List error codes

For the purposes of debugging, the review list create subprogram caters for the following error conditions. When an error is detected, the review list entry is created and the error condition is stored in the purpose field.

Error Number	*Explanation*
1	*Neither reason code nor verbiage supplied.*
2	*Both reason code and verbiage supplied.*
3	*Reason code supplied, not on file.*
4	*Both user Id and review group addressee supplied.*
5	*User Id supplied is not on file.*
6	*Review group supplied is not on file.*
7	*No addressee supplied.*
8	*Date supplied is not valid.*
9	*Year of date supplied is less than 1900*
10	*Transaction code supplied is not on file.*
11	*Both delete indicators are set to No.*

Installation of review list into a program

These examples are taken from a system called 'PR', which provides for maintenance of Purchase Orders using a CONSTRUCT generated program.

Installing a tracking entry into a construct program

This is an example of how to use the review list system to maintain a list of unfinished purchase orders (i.e., Purchase Orders with an Entry Status equal to 'U'). The review list will be addressed to a specific user.

Update the review list tables

Enter the review list system and update the following tables:

- *Add the purchase order header transaction to the transaction table. Transaction equals 'PRPMNTF', Library equals the library that PR is in, Program equals 'PRPMNTF'. This step need only be done the first time this transaction is referred to in the review list system.*
- *Add a reason code called 'POUNFIN' to the reason code table. This step is optional and is done so that the verbiage, invoked transaction and addressee will default rather than have to be coded in the calling program. Reason Code equals 'POUNFIN', Description equals 'UNFINISHED PURCHASE ORDER', Default User Id equals 'XXXXXXXX' (the user to whom the review list entry will be addressed), Default Transaction equals 'PRPMNTF'.*
- *Add a tracking group called 'POSTAT' to the tracking group table. The tracking group table contains an entry for each "thing" that you are tracking, in this case the status of the purchase order.*
- *Add an entry to the tracking table for each status that should be tracked. In this example, we are only tracking one value - 'U' (unfinished). Tracking Group equals 'POSTAT', Tracking Field Value equals 'U', Sequence equals 1, Reason Code equals 'POUNFIN', Delete If Tracking Value Changes equals 'Y', Delete with PFKey equals 'N'. equals 'N'.*

Update the CONSTRUCT program

Update the CONSTRUCT program as follows. Please note, if the program does not have BEFORE-ET, please insert the code in the subroutine BEFORE-ET immediately before the END TRANSACTION.

```
DEFINE DATA
:
**SAG DEFINE EXIT LOCAL-DATA
01 #TRACKING-GROUP(A6)
01 #NEW-VALUE(A40)
01 #RELATED-OBJECT(A40)
LOCAL USING CNXCREL /* #RL-ENTRY
**SAG END-EXIT
END-DEFINE
*
REPEAT
  REPEAT
  /*
  INPUT MAP
  /*
  PERFORM PROCESS-ACTIONS
  /*
  DEFINE PROCESS-ACTIONS
    :
    PERFORM BEFORE-ET
    :
  END-DEFINE
  **SAG DEFINE EXIT MISCELLANEOUS-SUBROUTINES
  ** User subroutines
  **SAG END-EXIT
  END-REPEAT
END-REPEAT
END
```

The only user exit that contains review list specific code is BEFORE-ET. This then calls the tracking manager subprogram (CNTTFVN).

Before ET

This user exit is invoked after the primary file has been updated. Any tracking review list entries for this entity must now be updated.

```
**SAG DEFINE EXIT BEFORE-ET
**
                    *
IF PR-PO-HDR.ENTRY-STATUS NE HOLD-ENTRY-STATUS
  * Set up basic review list variables:
  * 1) what is being tracked, e.g., the PO status
  * 2) the value of the thing being tracked, e.g., the current status
  * 3) the key of the entity
  ASSIGN #TRACKING-GROUP = 'POSTAT'
  ASSIGN #NEW-VALUE = PR-PO-HDR.ENTRY-STATUS
  IF #ACTION = #PURGE
    RESET #NEW-VALUE
  END-IF
  ASSIGN #RELATED-OBJECT = PR-PO-HDR.PO-ID-LATEST
  *
  * Set up any tracking table overrides (there are many possible).
  * Typical overrides are:
  * 1) the key to be displayed on the RL screen
  * 2) what to stack into the transaction when the RL item is
  * selected on the RL screen
  COMPRESS 'PO Id:' PR-PO-HDR.PO-ID-LATEST
    INTO #RL-ENTRY.REL-OBJ-DISPLAY
  COMPRESS 'D,' PR-PO-HDR.PO-ID-LATEST ','
    INTO #RL-ENTRY.STACK-DATA LEAVING NO SPACE
  *
  * Call the tracking program
  CALLNAT 'CNTTFVN'
#TRACKING-GROUP     /* the type of thing being tracked
#NEW-VALUE    /* the value of tracking group
#RELATED-OBJECT     /* the key of the entity
#PROGRAM            /* the calling program
#RL-ENTRY           /* the override group
END-IF
**SAG END-EXIT
```

Adding new values to track

If further values of Entry Status are to be tracked, no additional programming is required, only tracking tables need be updated. For example, say you want to send closed purchase orders (e.g., Purchase Orders with an Entry Status of 'C') to a review group called 'POCLOSED'. Two users have access to this review group, they are 'SPLRW' and 'SPLBM'.

Enter the review list system and update the following tables:

- *Add a review group called 'CLOSEDPO' to the review group table. Review Group equals 'CLOSEDPO', Description equals 'CLOSED PURCHASE ORDERS'.*
- *Add the users to the review group/user table, namely 'SPLRW' and 'SPLBM'.*
- *Add a reason code called 'POCLOSE' to the reason code table. This step is optional and is done so that the verbiage, invoked transaction and addressee will default rather than have to be coded in the calling program. Reason Code equals 'POCLOSE', Description equals 'CLOSED PURCHASE ORDERS', Default Review Group equals 'CLOSEDPO', Default Transaction equals 'PRPMNTF'.*
- *Add an entry to the tracking table to track the closed status. Tracking Group equals 'POSTAT', Tracking Field Value equals 'C' (closed), Sequence equals 1, Reason Code equals 'POCLOSE', Delete If Tracking Value Changes equals 'Y', Delete When Selected equals 'Y' (this means the RL item will be removed when it is selected on the RL screen).*

Now, whenever a PO has its entry status changed to 'C', any review list entries for the old value of the status will be deleted. An entry will appear for the review group 'CLOSEDPO' and users in this group can select the item.

CON-TACT tracking entry, complex example

This is a more complex example of how to use the review list system to maintain a list of authorized purchase orders whose amount is greater than or equal to $1,000,000.00. The review list will be addressed to a specific user.

Update the Review List tables

Enter the review list system and update the following tables:

- *Add the purchase order header transaction to the transaction table. Transaction equals 'PRPMNTF', Library equals the library that PR is in, Program equals 'PRPMNTF'. This step need only be done the first time this transaction is referred to in the review list system.*
- *Add a tracking group called 'POAMT' to the tracking group table. Tracking Group equals 'POAMT', Description equals 'PURCHASE ORDER AMOUNT'.*
- *Add an entry to the tracking table to indicate PO's with an authorized amount >= $1,000,000.00 will be tracked. Tracking Group equals 'POAMT', Tracking Field Value equals '>=$1,000,000.00', Sequence equals 1, User Id equals 'XXXXXXXX' (the user to whom the RL entry will be addressed), Transaction 'PRPMNTF', Delete If Tracking Value Changes equals 'Y', Delete With PFKey equals 'N'. Remember, you could have sent this to a Review Group rather than a user in which case all of the users in the group could select the review list entry.*

Update the CONSTRUCT program

In addition to the variables listed with the previous example, add the following variable:

```
01 #TEMP-AMOUNT(A14) /*  used to edit mask the authorized amount
```

The BEFORE-ET user exit should look as follows:

```
**SAG DEFINE EXIT BEFORE-ET
**
*
* Set up basic review list variables:
* 1) what is being tracked, e.g., the PO amount
* 2) the value of the thing being tracked, e.g., the amount
* 3) the key of the entity
IF PR-PO-HDR.ENTRY-STATUS = 'A'  /* only track authorized PO's
  ASSIGN #TRACKING-GROUP = 'POAMT'
  IF PR-PO-HDR.TOTAL-AMOUNT-AUTH >= 1000000.00
    ASSIGN #NEW-VALUE = '>=$1,000,000.00'
  ELSE
    RESET #NEW-VALUE
  END-IF
  ASSIGN #RELATED-OBJECT = PR-PO-HDR.PO-ID-LATEST
*
* Set up any tracking table overrides (there are many possible).
* 1) the key to be displayed on the RL screen
* 2) what to stack into the transaction when the RL item is selected
    on the RL screen
* 3) verbiage to be displayed on the RL screen
  COMPRESS 'PO Id:' PR-PO-HDR.PO-ID-LATEST
    INTO #RL-ENTRY.REL-OBJ-DISPLAY
  COMPRESS 'D,' PR-PO-HDR.PO-ID-LATEST ','
    INTO #RL-ENTRY.STACK-DATA LEAVING NO SPACE
  /*
  MOVE EDITED TOTAL-AMOUNT-AUTH (EM=ZZZ,ZZZ,ZZ9.99) TO #TEMP-AMOUNT
  MOVE LEFT JUSTIFIED #TEMP-AMOUNT TO #TEMP-AMOUNT
  COMPRESS 'Authorized amount $' #TEMP-AMOUNT
    INTO #RL-ENTRY.RL-VERBIAGE
*
* Call the tracking program
  CALLNAT 'CNTTFVN'
                                                  #TRACKING-GROUP     /* the type
of thing being tracked
                                                  #NEW-VALUE          /* the value
of tracking group
                                                  #RELATED-OBJECT     /* the key of
the entity
                                                  #PROGRAM            /* the
calling program
                                                  #RL-ENTRY           /* the
override group
END-IF
**SAG END-EXIT
```

Batch Submission Subsystem

System Overview

The Batch Submission System allows users to request batch jobs through an on-line transaction, thus avoiding the process of preparing and submitting JCL streams. The system is composed of six components:

- Standard Request Maintenance
- User Request Maintenance
- System Administrator User Request Maintenance
- Online Submission of Requests
- JCL Member Maintenance
- Batch Execution

Standard Request Maintenance is performed by the System Administrator and defines the programs that may be requested through the Batch Submission System. Default settings, such as printer assignments, number of copies and selection criteria are identified.

Execution of a program is requested using User Request Maintenance. Default settings may be overridden with parameters in effect for a single submission of the request.

System Administrator User Request Maintenance allows the system administrator to create, view and update user requests for any user.

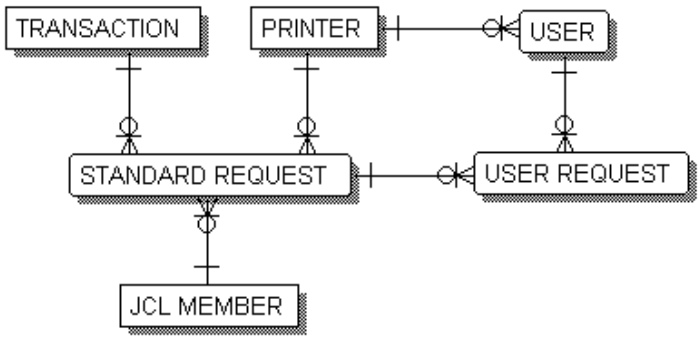
The Online Submission of Requests allows user to submit reports and batch processes from the online MEDS application.

JCL Member Maintenance allows the System Administrator to create JCL Members for execution of the User Requests.

Batch Execution processes the outstanding requests, submitting JCL to the spooler for execution. This program can be set up to run automatically at specified intervals during normal working hours.

If a parameter validation subprogram is needed, it is created. A new standard request is added with the same name as the batch transaction to be run. Once a standard request has been created, an authorized user may request for this job to be submitted in batch.

Entity Diagram



Description of the Entity Diagram

Each **standard request** can be the model for several **user requests**.

Each **user** may generate several user requests.

Each **printer** may be the default for several users. Each printer may also be the default printer for several standard requests.

Each **JCL Member** may be the generated JCL for several standard requests.

Each **transaction** may be initiated by several standard requests.

Menus

Batch Submission Subsystem Main Menu

```
     TRAN                  BATCH SUBMISSION SUBSYSTEM             *DATE
     PROGRAM                        MAIN MENU                     *TIME

            Id  Tran   Description
            --  ----   -------------------------------------------
             1  STREQ  Standard Request Maintenance
             2  USREQ  User Request Maintenance
             3  SYSADM System Administrator User Request Maint
             4  JCL    JCL Maintenance

             5  SRAT   Standard Request Audit Trail By Time
             6  SRAE   Standard Request Audit Trail By Entry









Id/*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit
Message Line ...
```

Maintenance Transactions

Standard Request Maintenance

```
     TRAN                  BATCH SUBMISSION SUBSYSTEM             *DATE
     PROGRAM              STANDARD REQUEST MAINTENANCE            *TIME    1 >

*Action           : _ (A,C,D,M,N,P)
*REQUEST NAME     : _________
 Request Desc     : ____________________________________
*JCL Member       : ________________
 #PROGRAM         : _________
*#PRINTER/FONT    : _________
 #COPIES          : __
 #PRINT BOTH SIDES: _ (Y,N)
 #JOBNAME         : _________

 Parameters Req'd : _ (Y,N)
 Validation Rtn   : _________




*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit             SYSAD USREQ       Left  Rght
Message Line ...
```

```
     TRAN                  BATCH SUBMISSION SUBSYSTEM             *DATE
 < 1 PROGRAM              STANDARD REQUEST MAINTENANCE            *TIME


 *Action           : _ (A,C,D,M,N,P)
 *REQUEST NAME     : _________
  Request Desc     : ____________________________________
 #PRINT FIRST PAGE : _ (Y,N)
 #REPORT FREQUENCY : ____________________________________________________
 #PURPOSE OF REPT  : ____________________________________________________
 #ACTION REQUIRED  : ____________________________________________________
 #REPT RECIPIENTS  : ____________________________________
                     ____________________________________
                     ____________________________________
                     ____________________________________
                     ____________________________________
                     ____________________________________
                     ____________________________________
                     ____________________________________
                     ____________________________________
                     ____________________________________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Help  Main  Retn  Quit             SYSAD USREQ       Left  Rght
Message Line ...
```

This transaction defines the programs that may be requested using the Batch Submission System. Default parameters and other settings are identified. It may only be accessed by the System Administrator and technical support personnel.

Field Descriptions

Panel 1

Action
The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

REQUEST NAME (A8)
A logical name for the report or batch job. The request name must exist as a transaction in the security system.

Request Description (A30)
A description of the request.

JCL Member (A8)
The name of the JCL member used to execute the request.

#PROGRAM (A8)
The program to execute.

#PRINTER /FONT(A8)
The default destination for printing output.

#COPIES (N2)
The default number of copies to print.

#JOBNAME (A8)
The Job Name to use in the JCL and when the output is printed. **Optional.** If left blank, the system will generate a jobname based on the user's userid and a suffix.

Parameters Required (A1)
Does the standard request require run time parameters? **Valid values:** [Y, N.]

Validation Rtn (A8)
The name of the subprogram used to prompt for and validate run-time parameters or selection criteria. If Parameters Required is Y, this field is required, else it must be blank.

Panel 2:

Action
The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

REQUEST NAME
See panel 1.

Request Description
See panel 1.

#PRINT FIRST PAGE (A1)
Indicates whether the standard first page of a report should be produced by this request. **Valid values:** [Y,N].

#REPORT FREQUENCY (A50)
Free form description of how often the report is produced. This will appear on the standard first page of a report. If Print First Page is N, this field must be blank, otherwise it is **optional.**

#PURPOSE OF REPT (A50)
Free form description of the report's purpose. This will appear on the standard first page of a report. If Print First Page is N, this field must be blank, otherwise it is **optional.**

#ACTION REQUIRED (A50)
Free form description of any action to be performed based on the production of this report. If Print First Page is N, this field must be blank, otherwise it is **optional.**

#REPT RECIPIENTS (A30)
Free form entry identifying the recipients of the report being produced. If Print First Page is N, this field must be blank, otherwise it is **optional. Max occur: 10, Typical occur: 3.**

Programming Notes

Add

A standard request can be added only if it has been previously defined as a transaction in the security subsystem.

Add and Change

If a parameter validation subprogram has been specified, pop up a window to enter run-time parameters. If it is entered and the subprogram is not on the Natural system file, the Natural error will be intercepted and a window displayed taking the user to his default menu.

Logging

Standard logging.

Security

No field level security.

User Request Maintenance

```
     TRAN                  BATCH SUBMISSION SUBSYSTEM             *DATE
     PROGRAM                USER REQUEST MAINTENANCE              *TIME    1 >

*Action             : _ (A,B,C,D,M,N,P)
 USER ID            :
*REQUEST NAME       : _________
 SEQUENCE NUMBER    : _________
*Status             : _ Status Description
*Printer / Font     : _________
 Number of copies   : __
 Job Name           :
 Release Date       : ____ __ __   Release Time  : 00 : 00
 Submitted Date     :              Submitted Time: 00 : 00
 Completed Date     :              Completed Time: 00 : 00
 Run Again?         : _ (Y,N)      After Interval: ___ Days, __ Hours, __ Mins
 Submit Immediately?: _ (Y,N)
 Parameters Required:



*Tran: ________ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11---PF12
      Help  Main  Retn  Quit             STREQ       copy  Left  Rght
Message Line ...
```

```
     TRAN                  BATCH SUBMISSION SUBSYSTEM             *DATE
< 1  PROGRAM                USER REQUEST MAINTENANCE              *TIME

*Action           : _ (A,B,C,D,M,N,P)
 USER ID          :
*REQUEST NAME     : _________     SEQUENCE NUMBER: _________
 Print First Page : _ (Y,N)
 Report Frequency : ____________________________________________________
 Purpose of Rept  : ____________________________________________________
 Action Required  : ____________________________________________________
 Rept Recipients  : _____________________________________
                    _____________________________________
                    _____________________________________
                    _____________________________________
                    _____________________________________
                    _____________________________________
                    _____________________________________
                    _____________________________________
                    _____________________________________
                    _____________________________________
*Tran: ________ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit             STREQ       copy  Left  Rght
Message Line ...
```

User Request Maintenance provides the on-line facility for doing the following:

- Copy a Standard Request, override any default settings or input parameters, and request execution of the job.
- Change requests that have not yet been submitted for execution.
- Request that a batch job be run after a specified date and time.
- Put a request on hold, so that it is not immediately submitted for execution.
- Resubmit a request, by copying a request that executed previously.
- Enter input parameters on-line, and validate them prior to the execution of the job.
- Determine whether and at what time the JCL for a request was submitted for execution.
- Determine whether and at what time a request successfully completed execution.
- Submit a request directly to the spooler.
- Scroll through a list of all the requests on file for your User ID.
- Scroll through a list of all the Standard Requests defined.

Users may copy a Standard Request, which creates a User Request. The user may only create a User Request if he has access to the corresponding Standard Request. He may view any User Requests that he has previously created, but does not see requests that have been created by other users.

When adding a request, the user will enter the request name. If the user is authorized to submit this request, all information from the corresponding standard request will be retrieved and displayed. Modifiable information may be changed and the request can be added with a status of on hold or requested. The user may also indicate that the job be submitted directly to the reader.

If a request is added with status of requested, it will be submitted the next time the batch submission process is run. However, if the release date and time are entered, the running of batch submission following that time will result in the job's being submitted to the spooler. Requests on hold will not be submitted until their status is changed to requested. Once a request has been submitted, its status is changed to submitted. The batch submission process will update the status with the appropriate code based on the successful or unsuccessful completion of the job.

Field Descriptions

Panel 1:

Action

The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

USER ID (A8)

The User Id. of the user. It must be present on the User table.REQUEST NAME (A8)

A logical name of a request, either as a standard request being copied or as a new user request being added.

SEQUENCE NUMBER (N8)

This sequence number is stored on the control record table and incremented when a user request is added.

Status (A1)

Each request has a status that indicates where it stands in the process of being requested, submitted for execution, and completed. **Valid values:** [R, S, C, H, E (requested, submitted, completed, on hold, error in submission).]

Printer (A8)

The printer to which output will be routed. This will default to the user's printer. If the user has none, it will default to the printer from the standard request.

Number of copies (N2)

The desired number of copies of the output.

Job Name (A8)

Display only. The job name to be used. This is copied from the standard request. For security, this field is protected from modification. This will be the name that appears on the job output.

Release Date (YYYY MM DD)

If a request is submitted but must not execute until after a particular time, the user may enter a Release Date, or a Release Date and Time, and set the status to Requested. If none is entered, the request will be submitted when the batch submission program is next executed.

Release Time (N7)

The time the request is to be released to the spooler. Since entering midnight (00:00) is the same as entering no release time, enter a time of 00:01 to request that the job be run at one minute after midnight.

Submitted Date (YYYY MM DD).

Display only. The date the batch execution sent the JCL to the spooler for execution.

Submitted Time (N7)

Display only. The time the batch execution sent the JCL to the spooler for execution.

Completed Date (YYYY MM DD).

Display only. The date that the request completed execution. This is only updated if the JCL Member includes a special step - **this is not done automatically by the system.**

Completed Time (N7)

Display only. The time the request completed execution. This is updated the same as Completed Date.

Run Again? (A1)
Should this job be run again after successful completion? **Valid values:** [Y, N.]

After Interval Days (N3)
The number of days after a successful completion of the job it should be run again.

After Interval Hours (N2)
The number of hours after a successful completion of the job it should be run again.

After Interval Minutes (N2)
The number of minutes after a successful completion of the job it should be run again.

Submit Immediately? (A1)
Should this request be submitted directly to the spooler? **Valid values:** [Y, N.] Cannot be Y if Release Date or Release Time is entered.

Parameters Required (A1)
Display only. Does the corresponding standard request require run time parameters? **Valid values:** [Y, N.]

Panel 2:

Action
The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

USER ID (A8)
The User Id. of the user. It must be present on the User table.

REQUEST NAME
See panel 1.

SEQUENCE NUMBER
See panel 1.

Print First Page (A1)
Indicates whether the standard first page of a report should be produced by this request. **Valid values:** [Y, N.]

Report Frequency (A50)
Free form description of how often the report is produced. This will appear on the standard first page of a report. **Optional.** Only allowed if Print First Page is Y.

Purpose of Rept (A50)
Free form description of the report's purpose. This will appear on the standard first page of a report. **Optional.** Only allowed if Print First Page is Y.

Action Required (A50)
Free form description of any action to be performed based on the production of this report. **Optional.** Only allowed if Print First Page is Y.

Rept Recipients (A30)
Free form entry identifying the recipients of the report being produced. **Optional.** Only allowed if Print First Page is Y. **Max occur: 10, Typical occur: 3.**

Programming Notes

Add or Change

A user may display a request submitted previously by entering the User Request Name and Sequence Number for a request with a status of Completed. By pressing PF9 to copy this request and changing the action to Add, and modifying any input parameters as desired, the new request is added.

Alternatively, he may copy a Standard Request by entering the name of the request, and pressing PF9 to copy. The request is displayed with the default printer, number of copies, and job name. The request also contains the default input parameters. Additionally, any of the information for the report's first page is copied. The status is set to Requested. Keying in an action of "A" and overriding any of the defaults and pressing enter causes the request to be added to the User Request file.

A sequence number, stored on the system control table, is incremented and assigned to the request. This number (along with the Standard Request Name) is entered if display of a specific request is desired.

If input parameters are required, a parameter validation routine is called to prompt for and validate the parameters. By pressing PF3, the routine is exited and further validation is bypassed. If help information has been created, the user can press PF1 while in the parameter window to view the help text.

The validation routine is also called to validate the existing parameters when a request is added or modified with a status of Requested. If the parameters do not pass validation, an error message is given. The request may be saved with the current parameter values by putting it On Hold. Once valid parameter values are known they may be entered and the status changed to Requested.

If the user wishes a request to run again, the user answers "Y" to Run Again? The user supplies a number of days or hours and minutes after which the request will be run again following a successful completion of the current request. Note that the request is only resubmitted if the original request is updated with a Completed status.

If the request is to be submitted immediately to the internal reader, the user responds with a "Y" to the prompt, "Submit Immediately?" When all other validation is successful, the request is written with the submitted date and time supplied and a status of Submitted. The batch submission program is invoked with an option to only submit the supplied request to the spooler.

Change

The user may change the status from "H" (on Hold) to "R" (Requested) and vice versa. However, none of the other statuses may be changed.

Logging

Logging is not required for User Request Maintenance as the user request acts as its own audit trail.

Security

No field level security.

System Administrator User Request Maintenance

```
     TRAN                  BATCH SUBMISSION SUBSYSTEM              *DATE
     PROGRAM     SYSTEM ADMINISTRATOR USER REQUEST MAINTENANCE    *TIME    1 >

 *Action             : _ (A,B,C,D,M,N,P)
  USER ID            : ________
 *REQUEST NAME       : ________
  SEQUENCE NUMBER    : ________
 *Status             : _ Description
 *Printer/Font       : ________ _
  Number of copies   : __
  Print Both Sides   : _ (Y,N)
  Job Name           : ________
  Release Date       : ____ __ __  Release Time  : 00 : 00
  Submitted Date     :            Submitted Time : 00 : 00
  Completed Date     :            Completed Time : 00 : 00
  Run Again?         : _ (Y,N)    After Interval: ___ Days, __ Hours, __ Mins
  Submit Immediately : _ (Y,N)
  Parameters Required:


*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit             STREQ       copy  Left  Rght
Message Line ...
```

```
     TRAN                  BATCH SUBMISSION SUBSYSTEM              *DATE
< 1  PROGRAM     SYSTEM ADMINISTRATOR USER REQUEST MAINTENANCE    *TIME

*Action           : _ (A,B,C,D,M,N,P)
 USER ID          :
*REQUEST NAME     : ________     SEQUENCE NUMBER: ________
 Print First Page : _ (Y,N)
 Report Frequency : ____________________________________________________
 Purpose of Rept  : ____________________________________________________
 Action Required  : ____________________________________________________
 Rept Recipients  : ____________________________________
                    ____________________________________
                    ____________________________________
                    ____________________________________
                    ____________________________________
                    ____________________________________
                    ____________________________________
                    ____________________________________
                    ____________________________________
                    ____________________________________
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit             STREQ       copy  Left  Rght
Message Line ...
```

The System Administrator User Request Maintenance provides the System Administrator with the facility to add, display, and update User Requests for any user. The job output will be routed according to the account information on file for the user ID entered on the request. The System Administrator may submit a job for a user which would not normally be permitted for the user entered on the job.

Field Descriptions

Panel 1

Action
The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

USER ID (A8)
The User Id. of the user whose request is being displayed or modified. It must be present on the User table.

REQUEST NAME (A8)
A logical name of a request, either as a standard request being copied or as a new user request being added.

SEQUENCE NUMBER (N8)
This sequence number is stored on the control record table and incremented when a user request is added.

Status (A1)
Each request has a status that indicates where it stands in the process of being requested, submitted for execution, and completed. **Valid values:** [R,S,C,H,E (requested, submitted, completed, on hold, error in submission).]

Printer/Font (A8)
The printer where output will be routed. This will default to the user's printer. If the user has none, it will default to the printer from the standard request.

Number of copies (N2)
The desired number of copies of the output.

Print Both Sides (A1)
Whether the report should print on both sides of the paper if the assigned printer has that capability. **Valid values:** [Y, N.]

Job Name (A8)
Display only. The job name to be used. This is copied from the standard request. For security, this field is protected from modification. This will be the name that appears on the job output.

Release Date (YYYY MM DD).
If a request is submitted but must not execute until after a particular time, the user may enter a Release Date, or a Release Date and Time, and set the status to Requested. If none is entered, the request will be submitted when the batch submission program is next executed.

Release Time (N7)
The time the request is to be released to the spooler. Since entering midnight (00:00) is the same as entering no release time, enter a time of 00:01 to request that the job be run at one minute after midnight.

Submitted Date (YYYY MM DD)
Display only. The date the batch execution sent the JCL to the spooler for execution.

Submitted Time (N7)
Display only. The time the batch execution sent the JCL to the spooler for execution.

Completed Date (YYYY MM DD)

Display only. The date that the request completed execution. This is only updated if the JCL Member includes a special step - **this is not done automatically by the system.**

Completed Time (N7)

Display only. The time the request completed execution. This is updated the same as Completed Date.

Run Again? (A1)

Should this job be run again after successful completion? **Valid values:** [Y, N.]

After Interval Days (N3)

The number of days after a successful completion of the job it should be run again.

After Interval Hours (N2)

The number of hours after a successful completion of the job it should be run again.

After Interval Minutes (N2)

The number of minutes after a successful completion of the job it should be run again.

Submit Immediately (A1)

Should this request be submitted directly to the spooler? **Valid values:** [Y, N.]

Parameters Required (A1)

Display only. Does the corresponding standard request require run time parameters? **Valid values:** [Y, N.]

Panel 2:

Action

The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.USER ID

See panel 1.

REQUEST NAME

See panel 1.

SEQUENCE NUMBER

See panel 1.

Print First Page (A1)

Indicates whether the standard first page of a report should be produced by this request. **Valid values:** [Y,N.]

Report Frequency (A50)

Free form description of how often the report is produced. This will appear on the standard first page of a report. **Optional.** Only allowed if Print First Page is Y.

Purpose of Rept (A50)

Free form description of the report's purpose. This will appear on the standard first page of a report. **Optional.** Only allowed if Print First Page is Y.

Action Required (A50)

Free form description of any action to be performed based on the production of this report. **Optional.** Only allowed if Print First Page is Y.

Rept Recipients (A30)

Free form entry identifying the recipients of the report being produced. **Optional.** Only allowed if Print First Page is Y. **Max occur: 10, Typical occur: 3.**

Programming Notes

See User Request Maintenance for further information.

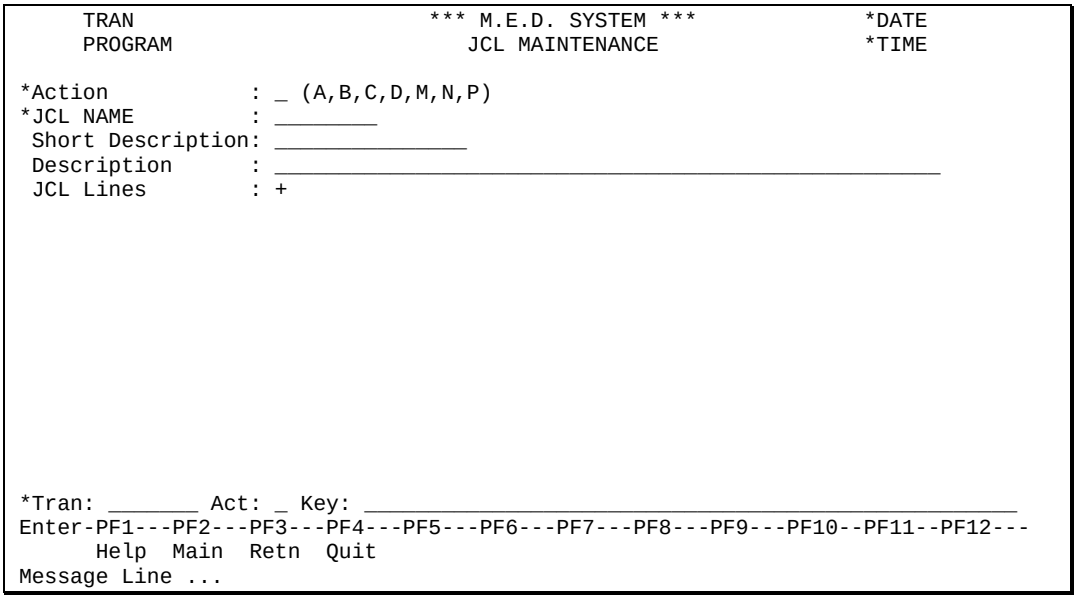
Logging

Logging is not required for System Administrator User Request Maintenance as the user request acts as its own audit trail.

Security

No field level security.

JCL Maintenance



The JCL Maintenance provides the System Administrator with the facility to add, display, and update batch submission JCL. Once created, JCL Members are available for use by the Standard Request Maintenance.

Field Descriptions

Action
: The actions codes for this field are (A)dd, (B)rowse, (C)lear, (D)isplay, (M)odify, (N)ext, (P)urge.

JCL Name (A8)
: The unique identifier of a JCL Member.

Short Description (A15)
: Short description of the JCL.

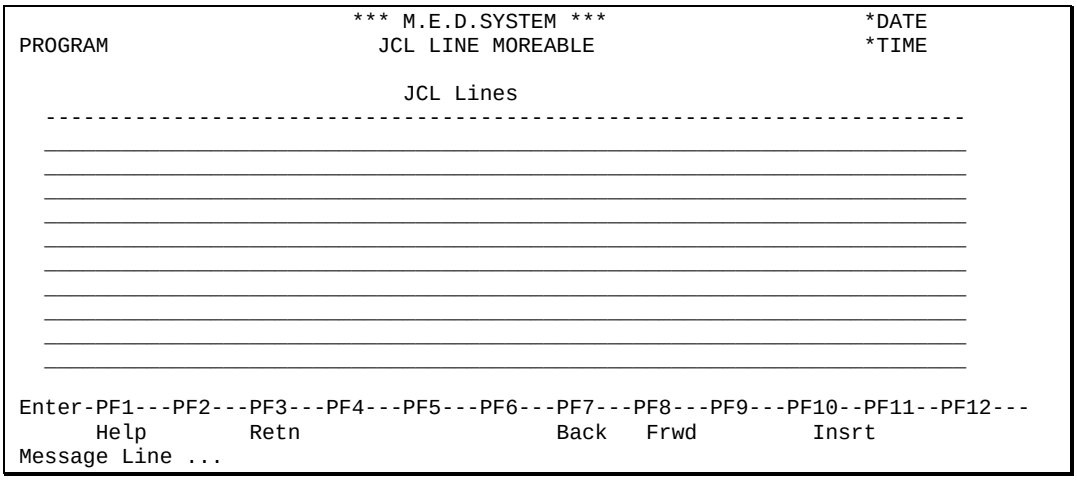
Description (A50)
: Description of the JCL.

JCL Line Indicator (A1)
: The moreable indicator. When the mnoreable indicator is changed to an "M", the JCL Moreable window will appear.

Security

No field level security.

JCL Moreable



Field Descriptions

JCL Lines (A72)

Individual JCL line. The combination of all of the JCL Lines together represents the JCL (job control language) that is submitted to the system spooler for execution. **Max Occurs**: 99.

Additional PF Keys

PF10 – Insrt

A blank line is inserted at the cursor position.

Security

No field level security.

Batch Update Processes

Periodic Submit Process (BSSELP)

The purpose of this program is to scan the user request file and determine which requests are to be submitted to the spooler. It must examine the status and select those whose status is "Requested". However, if a request has a specific release date and time, and that time has not yet arrived, that request is not to be submitted. When a request has been selected to be submitted, subprogram BSSUBN is passed the user request key and is called to submit the request to the spooler.

Processing

Read every user whose status equals "R".

If there is no release date, call the routine to format and submit the JCL. If there is a release date and that date has passed, call the routine to format and submit the JCL. If the release date is the current date and the release time has passed, call the routine to format and submit the JCL.

For each user request that meets the criteria; if the submission has no error, move "S" to status and the current date and time to date and time submitted and update; if there is an error, move "E" to status and update.

Update User Request Status (BSUPDP)

This program is placed in the job stream that is built from the request. It has the ISN of the user request and a constant of "E" or "C" as parameters. This program will be set up in the JCL twice and will run with the "E" status as input only when an error occurs in the batch program. If the batch program completes successfully, another execution of this program will run changing the status of the user request to "Complete". In this case it will also update the date and time completed with the current date and time.

Parameters

#ISN (N10)
Status (A1)

Processing

Read the user request for the ISN supplied. Move the status supplied to the user request. If the status supplied is "C", update the date and time completed for the user request with the current date and time.

If Run Again is "Y", take the Submit After days or hours and minutes and add them to current time giving the new Release Date and Time.

Common Routines

Retrieve Standard Request (BSUSRN)

This subprogram receives the user id and the standard request name. It verifies that the standard request is on the file. The appropriate information to be used by the user request is passed back to the calling program. The Security User file is searched to return the user's printer. If none is on the user file, the printer from the standard request is returned.

Input:

User Id (A8)
Standard Request Name (A8)

Output:

Error (L)
Error Message (A72)
Standard Request Name (A8)
Description (A30)
Parameters Required (A1)
Parameter Validation Subprogram (A8)
Input Parameters (A72) 1:10
Program (A8)
Printer (A8)
Number of Copies (N2)
Job Name (A8)
Print First Page (A1)
Report Frequency (A50)
Report Purpose (A50)
Action Required (A50)
Report Recipients (A30) 1:100

Initial Processing

Reset the output parameters.

Mainline Processing

Access the standard request record for I.Standard Request Name. If the standard request record is present, return all fields from the record to the output area. Access the security user record and return the user's default printer. If none is present, return the printer from the standard request.

Error Processing

If there is no record return **Undef* to O.Standard Request Name. Set O.Error to true.

Validate User Request Maintenance Screen Entries (BSUSRV)

This routine is passed the information keyed into the User Request Maintenance and System Administrator User Request Maintenance screens. It validates this information and passes back any error messages.

Input:

User Request Record
Pre-modification Status (A1)
Maint. Misc. fields
System Misc. fields
Security Population fields

Output:

Error (L)
Error Message (A72)

Initial Processing

Reset the output parameters.

Mainline Processing

Validate the input data. Make sure that if Parameters Required is "Y", a Parameter Validation Subprogram is entered. In these cases call the supplied subprogram and validate the parameters. If the parameter validation subprogram does not return #ACCEPT = True, make the status of the user request = H(old).

Error Processing

Set O.Error to true and construct an error message.

Validate Standard Request Maintenance Screen Entries (BSUSRV)

This routine is passed the information keyed into the Standard Request Maintenance screen. It validates this information and passes back any error messages.

Input:

Standard Request Record
Maint. Misc. fields
System Misc. fields

Output:

Error (L)
Error Message (A72)

Initial Processing

Reset the output parameters.

Mainline Processing

Validate the input data. Make sure the user has access to the transaction represented by the standard request. Make sure that if Parameters Required is "Y", a Parameter Validation Subprogram is entered. In these cases call the supplied subprogram and validate the parameters.

Error Processing

Set O.Error to true and construct an error message.

Build User Request (BSBURN)

This routine is called by any program that wants to submit a user request to the reader. An example would be an on-line program that, via a PF key, would indicate it wanted a report to be printed.

Input:

User Id (A8)
Standard Request Name (A8)
Submit Immediately (A1)
Do Not Input (L)

Input/Output:

Input Parameters (A72) (1:100)
Printer (A8)

Output:

Accept (L)
Error (L)
Error Message (A72)

Initial Processing

Reset the output parameters.

Mainline Processing

Use I.Standard Request Name and I.User Id to call subprogram BSSECVN to access the Standard Request record to verify authorization and subprogram BSUSRN to retrieve the data needed for the User Request. If IO.Input Parameters are supplied, pass them, otherwise use the input parameters from the standard request. If there is an input parameter validation subprogram for the request, it is called to verify the parameters are correct. Pass I.Do Not Input to the parameter validation subprogram. Then the subprogram to get the next user request number is called. Finally a User Request is stored. If the job is to be submitted immediately, BSSUBN will be called.

Error Processing

Set O.Error to true and construct an error message.

Validate Printers (BSPRTV)

This routine is passed the printer id. These are used to access the client's printer file. It verifies that the supplied printer exists on this file.

Input:

Printer Address (A8)

Output:

Error (L)
Error Message (A72)
Printer Location (A60)
Printer Type (A8)
Printer Computer (A3)
Printer IBM Destination (A8)
Printer Spool Command (A79)

Initial Processing

Reset the output parameters.

Mainline Processing

Build the key an access the client's printer file. The only error is if the printer is not on file. Pass the appropriate fields to the output area.

Error Processing

Set O.Error to true and construct an error message.

Format and Submit JCL (BSSUBN)

This subprogram may be called from User Request Maintenance, from System Administrator User Request Maintenance, from the Periodic Submit Process and from the subprogram that builds a user request.

This subprogram is passed all the information for the user request and uses this to retrieve the JCL Member. It then takes any substitution parameters that have been defined for the user request and places them in the appropriate place in the JCL. Each client may have their own process for submitting the JCL from on-line. The Batch Submission installation provides a non-Natural program NATIRDR which submits the JCL to the internal reader. This can be replaced depending on the needs of the client machine.

Input:

User Request record

Output:

Error (L)
Error Message (A72)

Initial Processing

Reset the output parameters.

Mainline Processing

Access the JCL Member using standard request name. Then, for each line of text, substitute the indicated variables with values from the user request. Call NATIRDR (or whatever program is used by the client) with the line of JCL and a return code as parameters.

Error Processing

Set O.Error to true and construct an error message.

Verify User's Access to Standard Request (BSSECV)

This subprogram may be called from Standard Request Maintenance, from User Request Maintenance, from the System Administrator User Request Maintenance and from the subprogram that builds a user request.

This subprogram is passed the User Id and Standard Request Name. It uses this to access security to determine whether the user supplied has access to the transaction supplied (standard request). It then uses security routines to populate security information.

Input:

User ID (A8)
Standard Request Name (A8)

Output:

Security Population Data
Error (L)
Error Message (A72)

Initial Processing

Reset the output parameters.

Mainline Processing

Verify the user has access to the standard request by including copycode SCVALC. Assuming valid access, populate security information by including copycode SCPOXC.

Error Processing

Set O.Error to true and construct an error message.

Display Submission Verification Map (BSSUBW)

This subprogram is called by the Security Subsystem transference subprogram SCXFRN. If a batch transaction has been indicated in a menu, on the transaction line or via a PF key, SCXFRN will build a user request. This subprogram is then called to display a status window along with any error message associated with the submission.

Input/Output:

Error (L)
Error Message (A72)

Initial Processing

Reset the local variables.

Mainline Processing

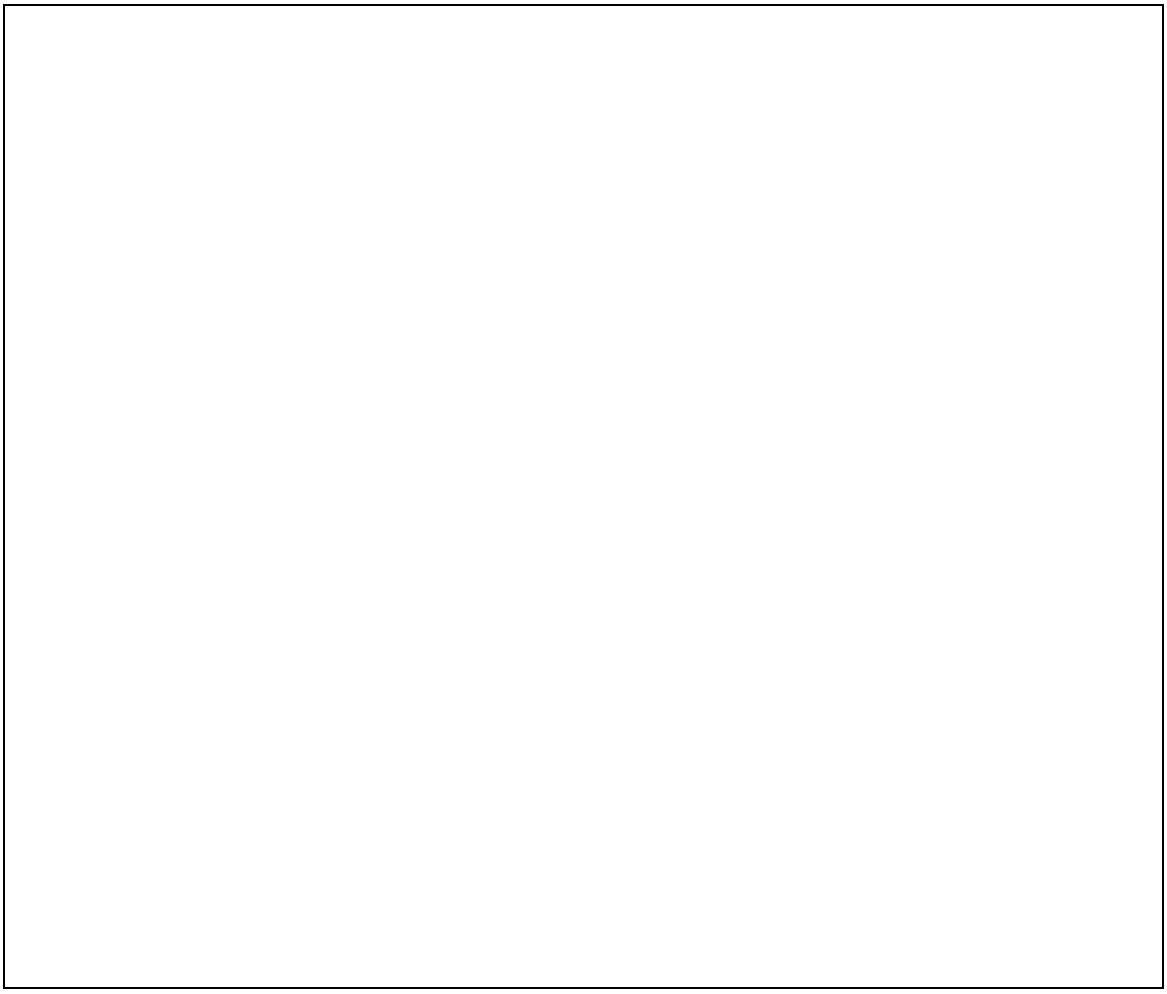
If I.Error is True, display error in submission message and format and display supplied system error message. Otherwise, display a successful submission message.

Error Processing

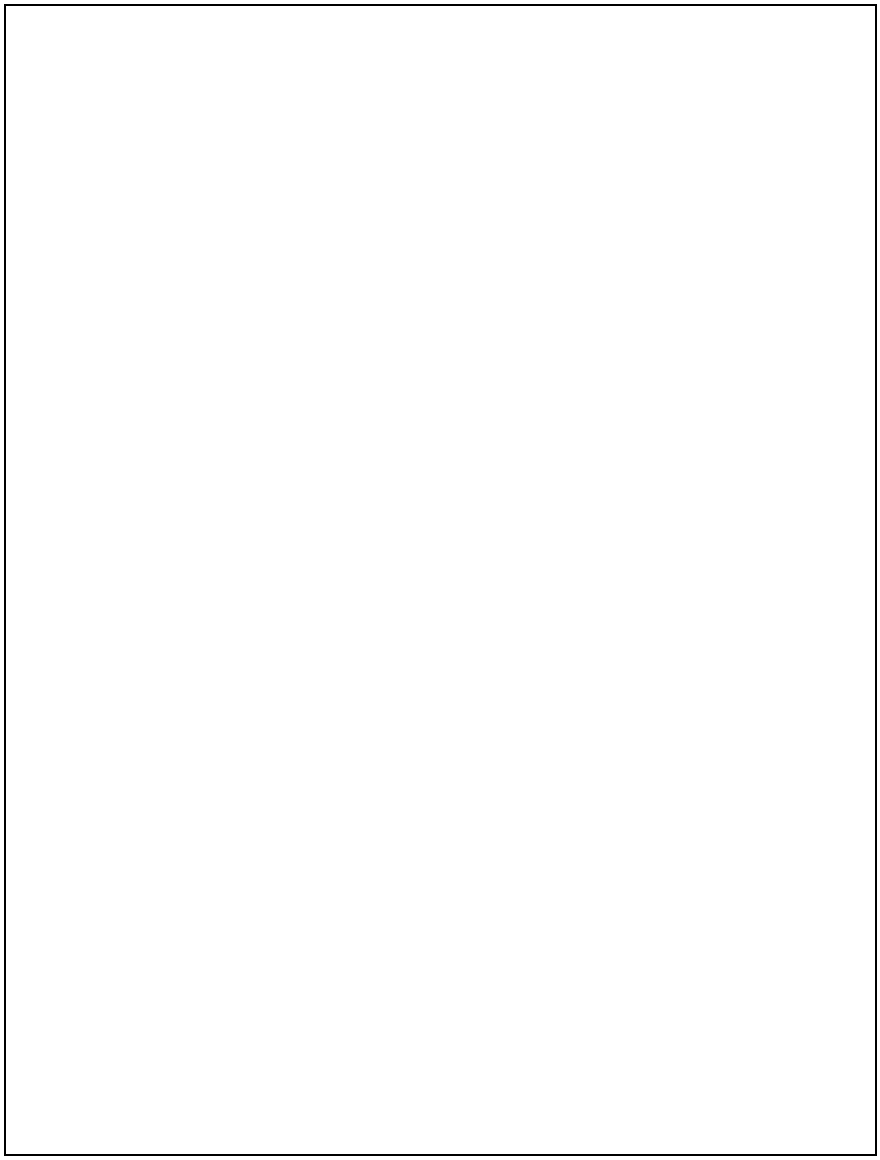
Standard on-line error processing.

System Processing Flow

System Processing Flow Diagram - Maintenance Functions



System Process Flow Diagram - Other Functions



Application Programmer's Guide

Introduction

The Batch Submission subsystem allows users to call for a particular job to be submitted to the reader by executing the Request Submission function directly or from the direct command line.

The normal series of tasks needed to be done before a program can call for JCL to be submitted is as follows:

- The existing JCL Members are examined to see if one already in existence can work for the new job.
- If not, new JCL should be added as a JCL Member.
- The batch transaction that is to be submitted by the new request is added to the Security Subsystem to a transaction group to which all users who will submit the request have access.
- If an input parameter validation subprogram is needed, it is created.
- Based upon the JCL, a standard request is either added or an existing one is chosen to be used.

Setting up a New JCL Member

The JCL Member Name identifies a set of JCL that, with certain substitutions, may be used to execute the report. The contents of the JCL Member are entered using the JCL Member Maintenance function. However, certain parameters cannot be identified in a standard set of JCL; they vary with each request.

The JCL may contain certain variable names, which are substituted by values defined either in Standard Request Maintenance, User Request Maintenance, or User Maintenance in the security subsystem. The values for these variables are determined when a user makes a request, and are automatically substituted by the system. The JCL may include the following variables:

#CMPRINT
: This literal generates a series of DD statements which define CMPRINT and CMPRTNN reports used by Natural in printing.

#COPIES
: The default number of copies is set in Standard Request Maintenance (probably to 1), and may be overridden in User Request Maintenance.

#ISN
: The ISN of the request, allowing update of the request after execution.

#JOBNAME
: The value that substitutes for Job Name is defined in Standard Request Maintenance.

#PARMS
: Entered where input parameters are to be substituted. This variable should be on a line of its own.

#PRINTER
: This literal is substituted by a series of JCL statements which define the printer destination based on the type of printer in the printer file.

#PROGRAM
: The program to execute is substituted by the value defined in Standard Request Maintenance.

#SEQ
: This is substituted with the last 7 digits of the sequence number of the User Request. This variable is useful for creating unique temporary dataset names (e.g., "#USER.S#SEQ.DATA"). Note that for the dataset name to be valid, #SEQ must be preceded by a letter - the IBM operating system does not allow purely numeric names in a dataset name.

#TIME
: If present in the JCL Member, the variable is substituted by the system time, in hours, minutes and seconds, preceded by a "T". This is to insure uniqueness in dataset names.

#USER
: If present in the JCL Member, the variable is substituted with the User ID that submitted the job. This may be used on the JOB statement, as the NOTIFY value, or as the USER parameter on JES3 MAIN card.

#PRINTFIRSTPAGE
: Indicates that the standard first report page is to be produced and that variables specific to that page will follow.

#REPORTFREQUENCY
: Indicates how often the report is run.

#REPORTPURPOSE
Indicates the purpose of the report.

#ACTIONREQUIRED
Indicates action to be performed as a result of the report's being produced.

#REPORTRECIPIENTS
Indicates up to ten (10) recipients of the report.

#FROM-ENV
Indicates which MEDS environment the request came from: Test,UAT,Train, or Prod.

#TO-ENV
Indicates which MEDS environment the request is going to: Test,UAT,or Train.

A set of standard JCL, which updates the user request with a completion status can be defined in the JCL Member. These final steps call a program in the Batch Submission System, passing it the key to the request file.

Input Parameter Validation Subprograms

If the requested program requires any run-time parameters, such as selection criteria, the name of the subprogram used to prompt for and validate the parameters is identified. The subprogram is called from Standard Request Maintenance to display a screen for input of default parameters, and to validate them. It is also called from User Request Maintenance to allow override of the default values with specific values for the request, and to validate the input parameters when the request is added.

The validation routine is passed a parameter to indicate whether they are being called from Standard Request Maintenance. The routine itself may contain different validation for each situation. For example, an input parameter may be optional as a default parameter defined in Standard Request Maintenance, but may be required when a request is submitted through User Request Maintenance.

The validation routine is also passed a parameter which indicates that "validation only" take place. This will avoid the displaying of the associated map that allows parameters to be keyed in.

When the map is displayed, help should be made available on the printer/font combination.

All input parameter validation subprograms should be structured identically. Each should use the parameter data area BSINPP described here for common fields.

```
* * ********************************
* * Title : BSINPP                          /* Used to validate input
* * Desc  : I/O for VALIDATION              /* parameters in batch sub
* * ********************************
  1 #BSINPP-IN
  2 #DO-NOT-INPUT                  L       /* Validate only if true
  2 #FROM-STD-REQ                  A     1 /* Is this called from Std Req?
  1 #BSINPP-IO
  2 PRINTER                        A     8
  2 #BSINPP-OUT
  2 #ACCEPT                        L       /* Accept the parameters
```

#DO-NOT-INPUT
: This should be set to true if the validation subprogram is to only validate the parameters and not display the map for modifying them.

#FROM-STD-REQ
: This is used if the validation subprogram is called from the Standard Request Maintenance program. Some input parameters may be optional as defaults but required on specific circumstances.

#ACCEPT
: Logical indicating that the parameter validation was successful.

PRINTER
: The printer to which the report was routed.

A second program specific PDA should also be included to contain the input validation parameters themselves. This structure should be redefined with the specific parameter definitions. Those fields should be used as sources for the associated map.

```
* * ********************************
* * Title : BSINPP2                         /* Used to validate input
* * Desc  : I/O for VALIDATION              /* parameters only in batch sub
* * ********************************
  1 #BSINPP2-IO
  2 INPUT-PARAMETERS               A    72 (1:10)
```

INPUT-PARAMETERS

These should be populated from the user request prior to calling the validation subprogram.

If the validation subprogram has encountered an error and the #DO-NOT-INPUT input parameter is true, then an error is returned to the calling program in the standard output PDA. The Construct model BS-VALID-SUBP should be used to generate these subprograms.

Building a User Request

When an application program wishes to submit a job to the internal reader (for instance, to print a report based on contents of a screen), the program can designate a PF key to submit the appropriate JCL. However, to actually submit the JCL, a user request must be built. This is done by calling subprogram BSBURN.

To do this the following parameter data area, BSBURP, should be used.

```
* * ********************************
* * Title : BSBURP                          /* Used to build the
* * Desc  : I/O for BSBURN                  /* User Request
* * ********************************
  1 #BSBURN-IN
  2 USER-ID                          A     8
  2 STANDARD-REQ-NAME                A     8
  2 #SUBMIT-IMMEDIATELY              A     1 /* Y If JCL to the reader now
  2 #DO-NOT-INPUT                    A     1 /* true = validate, no window
*
  1 #BSBURN-IO
  2 INPUT-PARAMETERS                 A    72 (1:10)
  2 PRINTER                          A     8
*
  1 #BSBURN-OUT                            /* Returned by calling program
  2 #ACCEPT                          L     /* Accept the parameters
```

USER-ID
: The user submitting the request.

STANDARD-REQUEST-NAME
: The name of the standard request to be used as the source of this user request.

#SUBMIT-IMMEDIATELY
: Set to "Y" if the request is to go directly to the internal reader.

#DO-NOT-INPUT
: Set to true if the validation subprogram is not to display a window to input parameters and is only going to validate parameters already set up on the standard request.

INPUT-PARAMETERS
: Parameters passed by calling program in instances when the parameter validation window will not be displayed. If these are present, they will override default parameters on the standard request.

PRINTER
: Returns the printer used by the request.

#ACCEPT
: Did the user accept the parameters as valid?

Subprogram BSBURN may encounter errors from subprograms it calls. These are: BSUSRN, which retrieves the standard request, CXXANNN, which gets the next sequence number, and whatever input parameter validation subprogram that may be called.

Requirements in Batch Programs

Any batch program that is to be requested by the Batch Submission system must include copycode to accept the User Id of the user submitting the request as input.

The following INCLUDE statement should be inserted at the beginning of each batch program.

```
INCLUDE BBSREPC /* Accept User Id and validate
```

The contents of copycode BSREPC is listed below.

```
*
* Accept the User Id of the user who submitted the batch job as
* the first input parameter and put it in #SCPOPN-IN.#USER-ID
* to be validated by the security system
INPUT(AD=MITL'_' SG=ON IP=OFF ZP=OFF)
        'Please enter your User Id:' #SCPOPN-IN.#USER-ID
INCLUDE SCOPOBC /* Populate security information.
```

The User ID will be automatically populated into the job stream when the user request is generated. This guarantees that a user who does not have access to a specific batch program may not submit that program via the Batch Submission system.

Affect On Security Subsystem

Subprogram SCXFRN recognizes batch transactions. If the transaction is defined as Batch, instead of normal transference, a User Request is built by calling subprogram BSBURN. Based on the successful or unsuccessful completion of the batch submission, subprogram BSSUBW is called to display a message window, advising the user that the request has been successfully (or unsuccessfully) submitted. The system automatically schedules a request for that program to be submitted, pending access verification and possibly parameter validation.

Globals

Globals are used in navigating between transactions. Each major subsystem has its own set of navigation variables. The navigation variables are populated when displaying information for the respective transaction. When users exit MEDS or if they use a function key to go to the TPL system, these navigational variables are lost.

To overcome this problem, the navigational variables are saved to an ADABAS file when the users exists the system (PF4 function key) or if the use a function key to go to the TPL system (PF12 on AUMEM and PF11 on RMemD). When the user returns to MEDS the same day we repopulate the navigational variables from what was saved at the time they exited the system.

See file ME-GLOBALS for the ist of globals that are saved.

The MEDS start up program will retrieve the navigational information from the file and populate globals if the globals were added that same day. The only instance where a user may not have his worker information populated occurs if a worker has multiple location definitions. If a worker has multiple location definitions and this is the first time he has signed on for the day then the worker information will not be populated. If this is a subsequent signon in the same day then worker information is defaulted from the previous signon. Note this only pertains to the worker location, and supervisor information.

The standard quit program (CD-QUIT) includes the copy code MEXGDAC that will save the globals. If a record already exists for the user on the file, it will be modified, otherwise a new record will be added.

The copy code MEXGDAC saves the globals to the Adabas file before leaving MEDS and must be included in module where we want to save the globals.

USER ID AUDIT

```
  UIDAUD                      *** M.E.D. SYSTEM ***               09/02/03
  MECUIDQ                         USER ID AUDIT                   11:06:31
                                                                  MEPROD

TRANSACTION TYPE          LOG ACTIONS | TRANSACTION TYPE          LOG ACTIONS
------------------------- ----------- | ------------------------- -----------
CASE                               34 | BUDGET QUESTION                    49
ELIGIBILITY DETERMINATION          67 | NON-BUDGET MEMBER                  25
APPLICATION                         0 | LASES CASE                          -
CERTIFICATION                      50 | LASES CASE CHILD                    -
SEGMENT                            14 | ABSENT PARENT                       -
PERSON                             56 | RENEWAL                             -
AU MEMBER                          68 | RENEWAL MEMBER                      -
PERSON INCOME                      28 | RENEWAL DETAIL                      -
PERSON EXPENSE                     30 |




*User ID: _________ Start Date: __/__/____ End Date  : __/__/____
*Tran: ________ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit        LogDT
```

Overview

This screen shows the number of log actions for a User ID during a certain period of time. It is broken down into counts of log entries for particular transactions. The User ID as well as a Start Date and End Date are specified through the input fields.

The PF6 key 'LogDT' is used with the cursor selecting a Transaction Type line either on the left or right half of the screen. This key goes to a User ID Audit Detail screen that lists the log actions included in the count selected on the main screen.

Any supervisor may use this transaction, and all users that are not supervisors will not be able to access this screen. (see section on Security Setup for User Audit below)

In order to show the progress in accessing the information, the Log Actions column will display a '-' for each Transaction Type until the Log Action counts are completed. The Log Action counts will appear one after another, replacing each line's '-' with the total as soon as that count is finished.

Field Descriptions

User ID (A8)
: This is the User ID for the person whose logged actions are to be examined for the time period specified.

Start Date (N8)
: This is the date that will be the earliest date for which logged actions will be counted.

End Date (N8)
: This is the date that will be the latest date for which logged actions will be counted.

Non-Standard PF-Keys

PF6 – LogDT

Press this key with the cursor on a transaction line to go to the detail screen for that transaction type.

Helproutines

User ID Help

The User ID Helproutine is invoked from the User ID field using PF1. *Technical Note: use SCUSRH*

File Changes

New keys are required for Person Month Income, Person Month Expense, and Case Month Income that allow the log files to be read in User ID order.

Transaction Type Line Descriptions

Case

This shows the actions performed on Cases for the User ID during the time frame specified. *Technical Note: File ME-CASE-LOG*

Eligibility Determination

This shows the actions performed on Eligibility Determinations for the User ID during the time frame specified. *Technical Note: File ME-ELIG-DETER-LOG*

Application

This shows the actions performed on Applications for the User ID during the time frame specified. *Technical Note: File ME-APPLICATION-LOG*

Certification

This shows the actions performed on Certifications for the User ID during the time frame specified. Actions performed in the “in-progress” area are not included in this count. *Technical Note: File ME-CERT-PERIOD-LOG*

Segment

This shows the actions performed on Segments for the User ID during the time frame specified. Actions performed in the “in-progress” area are not included in this count. *Technical Note: File ME-ELIG-DETER-LOG*

Person

This shows the actions performed on Persons for the User ID during the time frame specified. *Technical Note: File ME-PERSON-LOG*

AU Member

This shows the actions performed on AU Members for the User ID during the time frame specified. Actions performed in the “in-progress” area are not included in this count. *Technical Note: File ME-AU-MEMBER-LOG*

Person Income

This shows the actions performed on Person Monthly Income records for the User ID during the time frame specified. *Technical Note: File ME-PERSON-MONTH-INCOME-LOG*

Person Expense

This shows the actions performed on Person Monthly Expense records for the User ID during the time frame specified. *Technical Note: File ME-PERSON-MONTH-EXPENSE-LOG*

Budget Question

This shows the actions performed on Case Budget Information (Budget Questions) for the User ID during the time frame specified. *Technical Note: File ME-CASE-BUDGET-INFO-LOG*

Non-Budget Member

This shows the actions performed on Case Monthly Income for the User ID during the time frame specified. *Technical Note: File ME-CASE-MONTHLY-INCOME-LOG*

LASES Case

This shows the actions performed on LASES Cases for the User ID during the time frame specified. Actions performed in the “in-progress” area are not included in this count. *Technical Note: File ME-LASES-CASE-LOG*

LASES Case Child

This shows the actions performed on LASES Case Children for the User ID during the time frame specified. Actions performed in the “in-progress” area are not included in this count. *Technical Note: File ME-LASES-CASE-CHILD-LOG*

Absent Parent

This shows the actions performed on Absent Parents for the User ID during the time frame specified. *Technical Note: File ME-ABSENT-PARENT-LOG*

Renewal

This shows the actions performed on Renewals for the User ID during the time frame specified. *Technical Note: File ME-RENEWAL-LOG*

Renewal Member

This shows the actions performed on Renewal Members for the User ID during the time frame specified. *Technical Note: File ME-RENEWAL-MEMBER-LOG*

Renewal Detail

This shows the actions performed on Renewal Details for the User ID during the time frame specified. *Technical Note: File ME-RENEWAL-DETAIL-LOG*

Case Detail (example for all screens)

```
                          *** M.E.D. SYSTEM ***                  10/15/03
     XXXXXXX                    CASE DETAIL                      14:41:14

     Case
    Number       Action     Date       Time     Program
-------------- -------- ---------- --------- --------
9999999999999 Modified 2003-03-01 08:55:118 MECYICN
9999999999999 Modified 2003-03-01 08:57:341 MECPERF
9999999999999 Modified 2003-03-01 09:02:019 MECCASV
              Modified 2003-03-01 09:02:114 MECPERF
9999999999999 Modified 2003-03-01 09:04:456 MECCASV
9999999999999 Modified 2003-03-01 09:05:323 MECDETF
              Modified 2003-03-01 09:09:374 MECYICN
              Modified 2003-03-01 09:09:383 MECYICN
              Modified 2003-03-01 09:10:257 MECAPLV
              Modified 2003-03-01 09:11:450 MECDETF
              Modified 2003-03-01 09:14:198 MECCRTN
9999999999999 Modified 2003-03-01 09:23:249 MECDETF
              Modified 2003-03-01 09:25:047 MECYICN
User ID: ________ Start Date: __/__/____ End Date: __/__/____
*Tran: _______ Act: _ Key: ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Main  Retn  Quit              Back  Fwrd
```

Case Detail (items unique to this transaction type in yellow)

These detail screens will be accessed from the main screen using the PF6 key 'LogDT' and selecting a line with the cursor (either on the left or right half of the screen). The detail screens will not have their own transactions, but will be pop-up subprograms called from the main screen. The detail screens will display different information based on which Transaction Type was selected with the cursor on the main screen. All screens show the Action, Date, Time, and Program from the audit trails.

Field Descriptions

User ID (A8)

This is the User ID for the person whose logged actions are to be examined for the time period specified. This field is copied from the main screen and is display-only here.

Start Date (N8)

This is the date that will be the earliest date for which logged actions will be counted. This field is copied from the main screen and is display-only here.

End Date (N8)

This is the date that will be the latest date for which logged actions will be counted. This field is copied from the main screen and is display-only here.

Log Fields specific to each Transaction Screen

In addition to the Action, Date, Time, and Program from the audit trails, the screens contain other fields as follows:

Case Detail Screen

Case Number

Eligibility Determination Detail Screen

Case Number, Eligibility Determination Number

Application Detail Screen
Case Number, Application Number

Certification Detail Screen
Case Number, Certification Period Number, (Number of Budgets Linked to the Cert)

Segment Detail Screen
Case Number, Certification Period Number, Cert Period Type Case Number, Segment Number

Person Detail Screen
Person Number

AU Member Detail Screen
Case Number, Certification Period Number, AU Member Number

Person Income Detail Screen
Person Number, Month

Person Expense Detail Screen
Person Number, Month

Budget Question Detail Screen
Case Number, Month

Non-Budget Member Detail Screen
Case Number, Month

LASES Case Detail Screen
Absent Parent Number, Case Number, Certification Period Number

LASES Case Child Detail Screen
Absent Parent Number, Case Number, Certification Period Number, AU Member Number

Absent Parent Detail Screen
Absent Parent Number

Renewal Screen
Renewal Number

Renewal Member Screen
Renewal Number, Renewal Member Number

Renewal Detail Screen
Renewal Number, Renewal Detail Number

Security Setup for User Audit

The User ID Audit screen will only be available to Managers and Supervisors, the MEDS Unit and RedMane. Currently all Managers and Supervisors are part of the BHSFSUPV User Group, the MEDS Unit is in the MEDS User Group and RedMane is in the SPLUG User Group.

Summary of Changes

Add a new AUDIT Transaction Group, and add the UIDAUD Transaction to be in this new Transaction Group and give access to the BHSFSUPV, MEDS and SPLUG User Groups.

User Groups with access to AUDIT

The following table shows the User Groups that have access to the AUDIT Transaction Group with the allowable actions.

BHSFSUPV	**BHSF Managers-Supervisors**	**All**
MEDS	**MEDS User Group (Maintenance)**	**All**
SPLUG	**SPL User Group**	**All**

Snapshot Number Processing

This section of the document has been prepared as a guide to understanding the Snapshot Number and the procedures that need to be followed to implement its use. Several common modules have been created for use by both batch and on-line processes which require snapshot number functionality. An overview of the Snapshot Number along with a description of the common modules are detailed below.

Overview

The snapshot number can be thought of as a bookmark within the daily database update activity. All applications which update MEDS files will obtain the current snapshot number from the system parameter file through a common subprogram and then put that value into the SNAPSHOT-NO field of the record being maintained. When it is time to take a "snapshot" of the database, another common module will be invoked which will decrement the snapshot number in the system parameter file. This effectively puts a bookmark in the updates taking place as it allows us to retrieve all data which has a snapshot number within a certain range and determine what the state of our data was at the moment we took the snapshot. Note that this approach relies on the existence of LOG or history files to allow us to view records not as they appear at this moment, but as they appeared at the time of the snapshot. This is accomplished by having any program which updates a file insert the current snapshot number into the new version of the record.

Common Modules

Snapshot PDA (MEXSNPP)

This PDA contains the fields common to the subprograms described below.

```
#MEXSNPP-IN
  #NULL (A1)
#MEXSNPP-OUT
  #SNAPSHOT-NO(N8)
#MEXSNPP-IO
  #NULL(A1)
```

Return Snapshot Number (MEXSNPN1)

Return Snapshot Number is a common subprogram that returns to the calling module the current value of the snapshot stored in the system parameter file. All object subprograms will use this routine immediately prior to updating their records.

Decrement Snapshot Number (MEXSNPN2)

Decrement Snapshot Number is a common subprogram that causes the snapshot number stored in the system parameter file to be updated with a value of one less than the current value. Typical users of this routine will be batch processes that need to know the condition of the database at a point in time (e.g. balancing programs).

Run Control Processing

This section of the document has been prepared as a guide to understanding batch process controls and using the common modules which have been created for the purpose of facilitating their implementation. This guide will explain the common modules and define the procedures that must be followed for those modules to work as advertised.

Overview

The "prime-key" to the run control file is made up of a batch process id and a run number. A batch process id is a unique identifier for a batch process. Each batch process will have its own batch process id. A run number is an eight digit number that uniquely identifies each run of a process. The first run has a run number of 1. Each subsequent run increments this run number by one. Each run control record also has a status. There are three possible values for status:

"R" – Running.

"C" – Complete.

"A" – Abandoned.

The following are examples that show what happens during the run of a batch process that uses batch run control logic.

Example 1

PROCESS1 runs with no abends.

```
TIME  --+----------+---------+----------+----------+-------->
        |          |         |          |          |
      START        ET        ET         ET        END
```

- The processing begins at START with a unique process id (PROCESS1) and a unique run number (99999999). The status is set to 'R' for Running (see Fig. 1) and any fields on the control record that need updating are updated.
- After a certain number of updates and/or reads (counts accumulated in program), the restart data and any fields on the control record that need updating are updated and an end transaction (ET) is performed (see Fig. 2).
- The processing ends normally at END and status is set to 'C' for Complete (see Fig. 3). The restart data is reset and any appropriate fields on the run control record are updated.

PROCESS-ID	RUN-NO	STATUS	RESTART DATA				
PROCESS1	99999999	R					...

Fig. 1 - Run control record for PROCESS1 at START.

PROCESS-ID	RUN-NO	STATUS	RESTART DATA				
PROCESS1	99999999	R	KEY-VALUE				...

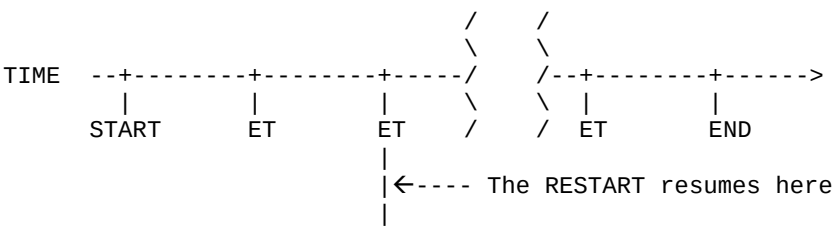
Fig. 2 - Run control record for PROCESS1 at ET.

PROCESS-ID	RUN-NO	STATUS	RESTART DATA				
PROCESS1	99999999	C					...

Fig. 3 - Run control record for PROCESS1 at END.

Example 2

PROCESS1 is run again; but, this time it abends after the second ET.



- The processing begins at START with the unique process id (PROCESS1) and a unique run number (99999998). The status is set to 'R' for Running (see Fig. 4) and any fields on the run control record that need updating are updated.
- After a certain number of updates and/or reads (counts accumulated in program), the restart data and any fields on the run control record that need updating are updated and an end transaction (ET) is performed (see Fig. 5).
- Some time after the second ET, the process abends. Any updates or stores since the last ET will be automatically backed out by the DBMS.
- After the reason for the abend is fixed, the process is restarted using the restart data. The restart data contains the key value of the record that was obtained (but not processed) just prior to the second ET. When the process is restarted, it skips all the records previous to this key value and resumes processing with the next record (see Fig. 6).
- The processing ends normally at END and status is set to 'C' for Complete (see Fig. 7). The restart data is reset and any appropriate fields on the run control record are updated.

PROCESS-ID	RUN-NO	STATUS	RESTART DATA				
PROCESS1	99999998	R					...

Fig. 4 - Run control record for PROCESS1 at START.

PROCESS-ID	RUN-NO	STATUS	RESTART DATA				
PROCESS1	99999998	R	KEY-VALUE				...

Fig. 5 - Run control record for PROCESS1 at ET.

PROCESS-ID	RUN-NO	STATUS	RESTART DATA				
PROCESS1	99999998	R	KEY-VALUE				...

Fig. 6 - Run control record for PROCESS1 at RESTART.

PROCESS-ID	RUN-NO	STATUS	RESTART DATA				
PROCESS1	99999998	C					...

Fig. 7 - Run control record for PROCESS1 at END.

Common Modules

Run Control Object Subprogram (MEXRUNU)

The Run Control Object Subprogram was created to access the run control file.

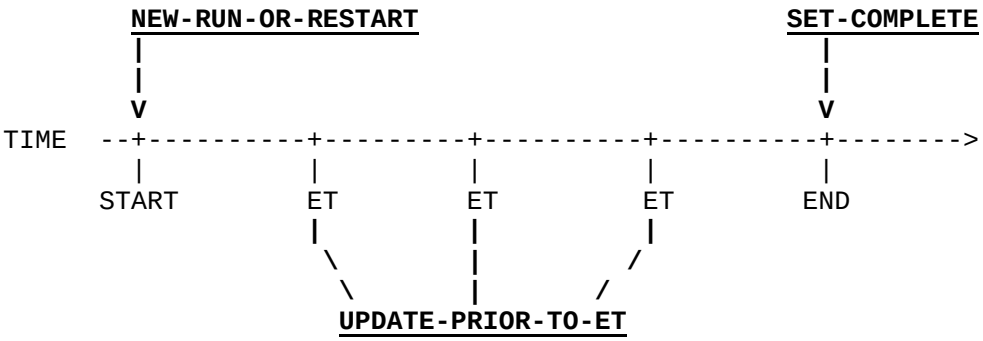
When a batch program needs to access the run control file, it does so by sending a valid message to the Run Control Object Subprogram. These messages are kept in a LDA.

Following is a list of the messages and an explanation of what the Run Control Object Subprogram does when it receives them.

The three most often used messages by a batch process are listed below. Refer to Example 3 for an explanation of where in the batch process they are used.

- NEW RUN OR RESTART - This message begins a new run if there is not an existing run control record or if the existing run control record has a status of 'C' for Complete or 'A' for Abandoned. If the status of the existing run is 'R' for Running, the process is restarted.
- UPDATE PRIOR TO ET - This message updates the RESTART-DATA and PGM-SPECIFIC-DATA fields on the run control record with the most recent data passed from the calling program. This message should be issued immediately prior to each periodic ET.
- SET COMPLETE - This message marks the current run control record with a status of 'C' for Complete. If the run control record does not exist, or the status is not 'R' for Running, a fatal error is logged.

Example 3 - Where the above three messages are used.



The rest of the messages are less often used. However, they are available when processing needs warrant their use.

- ABANDON MOST RECENT - This message marks the most recent run control record for the specified process id as status 'A' for Abandoned. A fatal error is logged if the process does not exist or if the most recent run is not status 'R' for running.
- NEXT RUN NUMBER - This message returns the run number that will be used for the next new run of the specified process. This function returns the process “SNAPSHOT” number. For more information, refer to the Snapshot technical document.
- PUT RECORD ON HOLD - This message issues a GET 'with hold' to put the run control record with a specific ISN on hold. This message assumes that normal run control initialization has been performed, and hence, the process id, run number, and timestamp are already known. This message should be issued immediately following each periodic ET.

Run Control PDA (MEXRUNP)

This PDA is made up of three groups:

INPUT - contains:

Process id (unique id for process)
Function (used to pass the messages described above)

OUTPUT - contains:

1) Status ('A', 'C' or 'R')
2) Run control ISN (ISN of current run control record)
3) Log timestamp (timestamp of last update)
4) Logical result (logical field used to flag existence)

I/O - contains:

1) Run number (unique number of run)
2) Restart data (used for knowing where to restart)
3) Program specific data (data kept for a specific process. For example – totals, counts, etc.)

Call to Run Control Object Subprogram (MEXRUNC0)

Copycode to be included in any program which uses run control logic. This copycode contains the subroutine SEND-MSG-TO-RUN-CONTROL which executes a CALLNAT to the run control object subprogram.

```
DEFINE SUBROUTINE SEND-MSG-TO-RUN-CONTROL
*
CALLNAT 'MEXRUNU'
        #MEXRUNP-IN (AD=O)
        #MEXRUNP-OUT (AD=M)
        #MEXRUNP-IO (AD=M)
        #MEXERRP (AD=M)  /* Error Handler PDA
*
END-SUBROUTINE /* SEND-MSG-TO-RUN-CONTROL
```

Periodic ET (MEXRUNC1)

Copycode to be included in any program which uses run control logic. This copycode contains logic for issuing an end transaction in a batch program. It should be included where the run control record is normally updated and an end transaction is issued.

```
DEFINE SUBROUTINE ISSUE-PERIODIC-ET
*
ASSIGN #MEXRUNP-IN.#FUNCTION = #MEXRUNL.#UPDATE-PRIOR-TO-ET
PERFORM SEND-MSG-TO-RUN-CONTROL
END TRANSACTION
ASSIGN #MEXRUNP-IN.#FUNCTION = #MEXRUNL.#PUT-RECORD-ON-HOLD
PERFORM SEND-MSG-TO-RUN-CONTROL
*
END-SUBROUTINE /* ISSUE-PERIODIC-ET
```

Procedures

Program Requirements

To make use of the Batch Process Control logic, a batch process should contain the following:

- Include the PDA **MEXRUNP** in all batch modules that are going to use the Run Control records. This PDA contains all the variables needed to interact with the Run Control subprogram.

  ```
  INCLUDE MEXRUNP /* Parameters to Run Control module
  ```

- Include the LDA **MEXRUNL** in all batch modules that are going to use the Run Control records. This LDA contains all the valid messages that can be passed to the Run Control subprogram.

  ```
  INCLUDE MEXRUNL /* Valid Run Control messages
  ```

- Include the copycode **MEXRUNC0** as the second to the last executable line in all batch modules that are going to use the Run Control records. This copycode contains the subroutine SEND-MSG-TO-RUN-CONTROL.

  ```
  INCLUDE MEXRUNC0 /* Subroutine SEND-MSG-TO-RUN-CONTROL
  ```

- Include the copycode **MEXRUNC1** as the last executable line in all batch modules that are going to use the Run Control records. This copycode contains the subroutine ISSUE-PERIODIC-ET. This subroutine contains the logic for issuing an END TRANSACTION command in the batch jobs. Perform ISSUE-PERIODIC-ET where you would normally issue an End Transaction.

  ```
  INCLUDE MEXRUNC0 /* Subroutine ISSUE-PERIODIC-ET
  ```

- Define the following local variables (these fields are stored on the control record):

  ```
  #RESTART-DATA (A250)
  ```

  This field should be redefined so that it can be used to find the point at which a restart is to begin. For example, redefine it into a restart key that matches the "prime-key" that the process is using.

  ```
  #PGM-SPECIFIC-DATA (A250)
  ```

  This field should be redefined into accumulated totals to be maintained by the program. Its use is specific to each program. It is strongly recommended that a separate LDA be created to define this local variable for each program using run control. The LDA should contain the redefinitions of Program Specific Data that are used by the program. In this way any program that needs to access the Program Specific Data of another program can include the appropriate LDA so that it has the correct redefinitions of this field.

- At the start of any batch process, load the #PROCESS-ID, assign #FUNCTION = #MEXRUNL.#NEW-RUN-OR-RESTART and perform SEND-MSG-TO-RUN-CONTROL to test for a new run or restart. In the case of a restart, move the RESTART-KEY and PGM-SPECIF-DATA fields from the run control file to their local variable counterparts (described above) and start processing based on the local redefinition of #RESTART-DATA.

- Before issuing an End Transaction in any batch module by performing ISSUE-PERIODIC-ET, load the RESTART-DATA and PGM-SPECIF-DATA fields on the run control record from their local counterparts.

- At the end of any batch process, assign #FUNCTION = #MEXRUNL.#SET-COMPLETE and perform SEND-MSG-TO-RUN-CONTROL to close the Run Control record and issue the End Transaction command.

Example Program

```
PROGRAM:  SAMPLE
**
DEFINE DATA
LOCAL USING MEXERRL /* Local Error Handler variables
LOCAL USING MEXERRP /* Error Handler PDA
LOCAL USING MEXRUNP /* Parameters to Run Control module
LOCAL USING MEXRUNL /* Valid Run Control messages
LOCAL
01 #RUN-CONTROL /* Local redefinitions of RESTART-DATA and PGM-SPECIF-DATA
  02 #RESTART-DATA (A250)
  02 REDEFINE #RESTART-DATA
    03 #RESTART-KEY (A11)
*
  02 #PGM-SPECIF-DATA (A250)
  02 REDEFINE #PGM-SPECIF-DATA
    03 #TOTAL (N10)
      /*
01 #LOCAL
  02 #ET-READ-COUNT (P5)
  02 #ET-UPDATE-COUNT (P5)
    /*
01 #CONSTANTS
  02 #ET-READ-LIMIT (P5) CONST<500> /* Maximum number of reads
  02 #ET-UPDATE-LIMIT (P5) CONST<150> /* Maximum number of updates
    /*
01 #LOGICALS
  02 #CAN-UPDATE (L)
END-DEFINE
**
**
**
INCLUDE MEXERRC1 /* Loads the #MAIN-PROGRAM
INCLUDE MEXERRC2 /* Loads the #PGM-TRACE
  /*
  /* Check for new start, or restart of an aborted job
  /*
ASSIGN #MEXRUNP-IN.#PROCESS-ID = 'PROCESS1'
ASSIGN #MEXRUNP-IN.#FUNCTION = #MEXRUNL.#NEW-RUN-OR-RESTART
PERFORM SEND-MSG-TO-RUN-CONTROL
MOVE BY NAME #MEXRUNP-IO TO #RUN-CONTROL /* Move from run control to local
  /*
  /* If #RESTART-KEY has a value in it, the process is restarting.
  /* Read starting from the #RESTART-KEY.
  /*
READ EXAMPLE-FILE-VIEW BY EXAMPLE-KEY FROM #RESTART-KEY
  ADD 1 TO #ET-READ-COUNT
  /*
  /* Check to see if an end transaction should be performed.
  /* If so, load the restart and program specific information
  /* before performing the end transaction.
  /*
  IF #ET-READ-COUNT GE #ET-READ-LIMIT OR
     #ET-UPDATE-COUNT GE #ET-UPDATE-LIMIT
    ASSIGN #RESTART-KEY = EXAMPLE-FILE-VIEW.EXAMPLE-KEY
       MOVE BY NAME #RUN-CONTROL TO #MEXRUNP-IO /* Move from local to run control
       PERFORM ISSUE-PERIODIC-ET
  END-IF
**
```

```
**  Now process the record.  For example:
**
...
...
  ASSIGN #CAN-UPDATE = FALSE
**
  IF EXAMPLE-FILE-VIEW.FIELD1 = ' '
    ASSIGN #CAN-UPDATE = TRUE
  END-IF
**
  IF #CAN-UPDATE
    (GET-RLOOP1.)
    GET EXAMPLE-FILE-VIEW *ISN(READ-LOOP1.)
    UPDATE (GET-RLOOP1.)
    ADD 1 TO #ET-UPDATE-COUNT
    ADD 1 TO #TOTAL
  END-IF
...
...
END-READ
...
  /*
  /* Close the run control record as the process has completed successfully
  /*
ASSIGN #MEXRUNP-IN.#FUNCTION = #MEXRUNL.#SET-COMPLETE
MOVE BY NAME #RUN-CONTROL TO #MEXRUNP-IO /* Move from local to run control
PERFORM SEND-MSG-TO-RUN-CONTROL
END TRANSACTION /* Issue the final ET command
...
...
INCLUDE MEXERRC0 /* Subroutine LOCAL-ERROR-HANDLER
INCLUDE MEXRUNC0 /* Subroutine SEND-MSG-TO-RUN-CONTROL
INCLUDE MEXRUNC1 /* Subroutine ISSUE-PERIODIC-ET
END
```

Run Control (sample DDM)

```
   1  Z1  SP-PROCESS-ID-RUN-NO            A   16  N S  * PRIME KEY
          HD=PROCESS ID/RUN NUMBER
 *        -------- SOURCE FIELD(S) -------
 *        PROCESS-ID(1-8)
 *        RUN-NO(1-8)
   1  AG  PROCESS-ID                      A    8  N    * ID OF BATCH PROCES
   1  AI  RUN-NO                          N  8.0  N    * IDENTIFIES A PARTI
   1  AJ  RUN-STATUS-CODE                 A    1  N    * STATUS OF THE RUN
   1  AK  RUN-START-DATE                  N  8.0  N    * DATE RUN STARTED
   1  AL  RUN-START-TIME                  N  7.0  N    * TIME RUN STARTED
   1  AM  RUN-END-DATE                    N  8.0  N    * DATE RUN ENDED IN
   1  AN  RUN-END-TIME                    N  7.0  N    * TIME RUN ENDED STO
   1  AO  INTERIM-UPD-FAILURE-CNT         P  5.0  N    * NUMBER OF INTERIM
   1  AP  RESTART-DATA                    A  250  N    * RESTART POSITIONIN
   1  AQ  RESTART-CNT                     P  5.0  N    * NUMBER OF RESTARTS
 P 1  AR  RESTART-DURATION-PE                           * 1 OCCURS PER RESTA
   2  AS  RESTART-DATE                    N  8.0  N    * DATE RESTARTED IN
   2  AT  RESTART-TIME                    N  7.0  N    * TIME WHEN RESTARTE
   2  AU  LAST-ET-DATE                    N  8.0  N    * DATE OF THE LAST E
   2  AV  LAST-ET-TIME                    N  7.0  N    * TIME OF THE LAST E
   1  AW  PGM-SPECIF-DATA                 A  250  N    * PROGRAM SPECIFIC D
 G 1  AX  LOG-FIELDS                                    * GROUP OF LOG FIELD
   2  AY  LOG-ACTION                      A    1  N    * LAST DATABASE ACTI
   2  AZ  LOG-TIMESTMP                    B    8  N    * *TIMESTMP OF LAST
   2  BA  MAIN-PGM                        A    8  N    * MAIN CALLING PROGR
   2  BB  SUB-MODULE                      A    8  N    * UPDATING MODULE
   1  BC  CO-NO                           N  2.0       * UNIQUE COMPANY NUM
   1  BD  DIV-NO                          N  2.0       * DIVISION NUMBER
 G 1  BE  LAST-FM-STAMP                                 * LAST FILE MAINTENA
   2  BF  LAST-FM-DATE                    N  8.0       * LAST FILE MAINTENA
   2  BG  LAST-FM-TIME                    N  6.0       * LAST FILE MAINTENA
   2  BH  LAST-FM-USER                    A    8       * NATURAL USERID OF
 G 1  BI  CREATION-STAMP                                * RECORD CREATION ST
   2  BJ  CREATION-DATE                   N  8.0       * DATE RECORD ADDED
   2  BK  CREATION-TIME                   N  6.0       * TIME RECORD ADDED
   2  BL  CREATION-USER                   A    8       * NATURAL USERID OF
```

Run Control/Snapshot Processing

This section of the document has been prepared as a guide to understanding the Run Control Snapshot process controls and using the common modules that have been created for the purpose of facilitating their implementation.

Overview

Run Control Snapshot processing operates under the same principles as Snapshot Number Processing (for more information refer to the Snapshot Number Processing technical document), however, instead of using a single system wide “snapshot” number, Run Control Snapshot processing utilizes the next run control number for a specific batch process. All applications that update records relating to the specific batch process must obtain the next run number from the run control file and “stamp” it onto the record being maintained. Normal run control processing (at the start of the next batch process) will update the run number and in effect take a "snapshot" of the database records. Note that this approach relies on the existence of batch process specific run numbers on all records (including LOG or history files). This technique is useful when trying to retrieve all database records that have been modified since the last batch process.

Common Modules

Run Control Object Subprogram (MEXRUNU)

When a program needs to access a next run control number (for a specific batch process), it does so by sending the "NEXT RUN NUMBER" message to the Run Control Object Subprogram.

For more information about the Run Control Object Subprogram refer to the Batch Control Processing technical document.

Run Control PDA (MEXRUNP)

For information about the Run Control PDA refer to the Batch Control Processing technical document.

Call to Run Control Object (MEXRUNC0)

Copycode to be included in any program which uses run control logic. This copycode contains the subroutine SEND-MSG-TO-RUN-CONTROL that executes a CALLNAT to the run control object.

For information about the Run Control PDA refer to the Batch Control Processing technical document.

Procedures

Database Update Program (Object Subprogram) Requirements

To make use of Run Control Snapshot processing, all application modules that update records relating to the specific batch process must contain the following:

- Include the PDA **MEXRUNP.** This PDA contains all the variables needed to interact with the Run Control subprogram.

```
INCLUDE MEXRUNP /* Parameters to Run Control module
```

- Include the LDA **MEXRUNL.** This LDA contains all the valid messages that can be passed to the Run Control subprogram.

```
INCLUDE MEXRUNL /* Valid Run Control messages
```

- Include the copycode **MEXRUNC0.** This copycode contains the subroutine SEND-MSG-TO-RUN-CONTROL.

```
INCLUDE MEXRUNC0 /* Subroutine SEND-MSG-TO-RUN-CONTROL
```

- Prior to modifying the object record, load the #PROCESS-ID, assign #FUNCTION = #MEXRUNL.# NEXT-RUN-NUMBER and perform SEND-MSG-TO-RUN-CONTROL. Move the run control number returned into the object record and proceed to the object UPDATE.

Program Requirements For Using Run Control Snapshot

To make use of Run Control Snapshot processing, a batch process is required to contain all of the same program requirements as any other program using Batch Process Control (for more information, refer to the Batch Process Control technical document). After normal batch control startup processing, the current run control number is compared to the run control number “stamped” on the database record to determine the records “eligibility” for processing. Typically run numbers “stamped” on records are also included in at least one ADABAS super-descriptor to facilitate quick retrieval.

Example Program

```
PROGRAM:  SAMPLE
**
DEFINE DATA
LOCAL USING MEXERRL /* Local Error Handler variables
LOCAL USING MEXERRP /* Error Handler PDA
LOCAL USING MEXRUNP /* Parameters to Run Control module
LOCAL USING MEXRUNL /* Valid Run Control messages
LOCAL
01 #RUN-CONTROL /* Local redefinitions of RESTART-DATA and PGM-SPECIF-DATA
  02 #RESTART-DATA (A250)
  02 REDEFINE #RESTART-DATA
    03 #RESTART-KEY (A20)
      04 #RUN-NBR (N8)
      04 #CUST-NO (A12)
*
  02 #PGM-SPECIF-DATA (A250)
  02 REDEFINE #PGM-SPECIF-DATA
    03 #TOTAL (N10)
     /*
01 #LOCAL
  02 #ET-READ-COUNT (P5)
  02 #ET-UPDATE-COUNT (P5)
   /*
01 #CONSTANTS
  02 #ET-READ-LIMIT (P5) CONST<500> /* Maximum number of reads
  02 #ET-UPDATE-LIMIT (P5) CONST<150> /* Maximum number of updates
   /*
01 #LOGICALS
  02 #CAN-UPDATE (L)
END-DEFINE
**
**
**
INCLUDE MEXERRC1 /* Loads the #MAIN-PROGRAM
INCLUDE MEXERRC2 /* Loads the #PGM-TRACE
  /*
  /* Check for new start, or restart of an aborted job
  /*
ASSIGN #MEXRUNP-IN.#PROCESS-ID = 'PROCESS1'
ASSIGN #MEXRUNP-IN.#FUNCTION = #MEXRUNL.#NEW-RUN-OR-RESTART
PERFORM SEND-MSG-TO-RUN-CONTROL
MOVE BY NAME #MEXRUNP-IO TO #RUN-CONTROL /* Move from run control to local
  /*
  /* If #RESTART-KEY has a value in it, the process is restarting.
  /* Otherwise, build starting key with current run-nbr
  /*
IF #RUN-CONTROL.#RESTART-KEY = ""
  ASSIGN #RUN-CONTROL.#RUN-NBR = #MMXRUNP-IO.#RUN-NBR
END-IF
/*
/* Read starting from the #RESTART-KEY.
/*
READ-EXMPL.
READ EXAMPLE-FILE-VIEW BY EXAMPLE-KEY FROM #RESTART-KEY
  /*
  /* If we have gone beyond our current run number, exit
  /*
  IF EXAMPLE-FILE-VIEW.RUN-NBR > #RUN-CONTROL.#RUN-NBR
```

```
    ESCAPE BOTTOM (READ-EXMPL.)
  END-IF
  /*
  ADD 1 TO #ET-READ-COUNT
  /*
  /* Check to see if an end transaction should be performed.
  /* If so, load the restart and program specific information
  /* before performing the end transaction.
  /*
  IF #ET-READ-COUNT GE #ET-READ-LIMIT OR
     #ET-UPDATE-COUNT GE #ET-UPDATE-LIMIT
    ASSIGN #RESTART-KEY = EXAMPLE-FILE-VIEW.EXAMPLE-KEY
      MOVE BY NAME #RUN-CONTROL TO #MEXRUNP-IO /* Move from local to run control
      PERFORM ISSUE-PERIODIC-ET
  END-IF
**
**  Now process the record.  For example:
**
...
...
  ASSIGN #CAN-UPDATE = FALSE
**
  IF EXAMPLE-FILE-VIEW.FIELD1 = ' '
    ASSIGN #CAN-UPDATE = TRUE
  END-IF
**
  IF #CAN-UPDATE
    (GET-RLOOP1.)
    GET EXAMPLE-FILE-VIEW *ISN(READ-EXMPL.)
    UPDATE (GET-RLOOP1.)
    ADD 1 TO #ET-UPDATE-COUNT
    ADD 1 TO #TOTAL
  END-IF
...
...
END-READ /* (READ-EXMPL.)
...
  /*
  /* Close the run control record as the process has completed successfully
  /*
ASSIGN #MEXERRP-IN.#FUNCTION = #MEXRUNL.#SET-COMPLETE
MOVE BY NAME #RUN-CONTROL TO #MEXRUNP-IO /* Move from local to run control
PERFORM SEND-MSG-TO-RUN-CONTROL
END TRANSACTION /* Issue the final ET command
...
...
INCLUDE MEXERRC0 /* Subroutine LOCAL-ERROR-HANDLER
INCLUDE MEXRUNC0 /* Subroutine SEND-MSG-TO-RUN-CONTROL
INCLUDE MEXRUNC1 /* Subroutine ISSUE-PERIODIC-ET
END
```

Next Number Processing

Many processes, both online and batch, depend upon the system (not the user) to provide primary keys for records which are to be added. The *Next Available Number* modules have been developed as an efficient means by which to do this. For example, when a purchase order is created, either by a batch or online process, the next available number modules may be invoked to determine its purchase order number. This example will be used throughout this document to explain details.

Overview

The next available number scheme is primarily comprised of an ADABAS file and a subroutine that accesses it. Secondarily, there is also a program that may occasionally be used to maintain the file. On the next available number file, a code is associated with each primary key for which next available number processing is performed. For example, purchase order number may be referred to by the next available number processes as 'PONUMBER'. When a module needs to add a new record and is using next available number processing, it calls a subroutine with the primary key's next available number code. This subroutine then reads the next available number file to determine the next value to use as a key for that code.

There are two different ways in which the subroutine can be instructed to reference the next available number file (both of which are explained in greater detail in the subroutine's module discussion that follows).

1) The calling module may need a single available number. In this case, this next number is retrieved and returned to the calling module. The next available number for the key code is incremented by one to reflect the fact that it is the next available value.

2) The calling module may need a series of available numbers. In this case, the next available number subroutine will need to be called twice. It is first invoked in order to retrieve the next available number, in a similar manner to 1) above. However, the calling module may continue to assign key numbers starting from this retrieved value and incrementing by one each time without calling the subprogram for each key value. Instead, the calling module may wait until it has finished assigning key values, and then invoke the subprogram. The subprogram will update the next available number file in such a way that, if the last number used by the calling module was N, the next available number will be N+1.

Occasionally, there may exist the need to instate a new primary key code in the next available process, to purge a key code that is no longer in use, or to simply view the key codes and the current values of their next available numbers. These tasks are accomplished by the next available number maintenance screen program, the module discussion for which appears later in this document.

Structure of the Next Number File

For a particular next available number code (e.g., 'PONUMBER'), a range of values for the key to take is established (for example, 1 to 1000). However, there will often exist multiple records on file corresponding to this code. Each of these records corresponds to a sub-range of values for the key to assume, and is referred to as a sequence. For example, PONUMBER may be allowed to take values from 1 to 1000, and may have 10 sequences allocated for it. In this example, the records on the next available number file corresponding to the key code PONUMBER would have the following information on them:

KEY CODE	SEQUENCE	VALUE RANGE
PONUMBER	1	1-100
PONUMBER	2	101-200
PONUMBER	3	201-300
PONUMBER	4	301-400
PONUMBER	5	401-500
PONUMBER	6	501-600
PONUMBER	7	601-700
PONUMBER	8	701-800
PONUMBER	9	801-900
PONUMBER	10	901-1000

There exist multiple sequences for a single key code for the important reason of enabling multiple users to simultaneously access next available number information for the key code - each can access a different sequence. For example, if a batch process is creating purchase orders, it will be referencing and updating one of the sequences of values for PONUMBER. If there is only one such sequence, then an online user attempting to add a purchase order would be unable to do so for as long as the batch process was running. With multiple sequences, the two users can access next available number information without interfering with one another. This implies that there is not a unique next available number for a particular key code. Instead, each sequence has a next available number associated with it (lying within the sequence's range of values). When a process inquires or updates the next available number, it is actually updating the next available number for a particular sequence. Next available numbers are used sequentially within a sequence. In other words, if a process uses the next available number 104 in sequence 2 in the example above, then it would set the next available number of the sequence to be 105. Similarly, if a process has determined that the next available number in sequence 3 is 207, and the process needs to assign 10 numbers, it should assign the numbers 207 through 216 and set the next available number in sequence 3 to be 217. (These processes are described in greater detail in the subroutine's MEXNXNU valid message processing discussion.) The particular sequence that a calling module will reference is explained below in the subroutine MEXNXNU processing. After a process has updated a next available number sequence, the user ID of that process will be stored on the sequence record. So at a particular instant in time, the next available number information for PO number might look like this (notice that some of the sequences have not yet been used):

KEY CODE	SEQUENCE	VALUE RANGE	NEXT AVAIL NUMBER	USER ID (LAST USE)
PONUMBER	1	1-100	113	
PONUMBER	2	101-200	104	SCOTT
PONUMBER	3	201-300	207	CAROL
PONUMBER	4	301-400	301	NAT
PONUMBER	5	401-500	401	
PONUMBER	6	501-600	572	MIKE
PONUMBER	7	601-700	601	
PONUMBER	8	701-800	730	JOCELYN
PONUMBER	9	801-900	830	DUNCAN
PONUMBER	10	901-1000	922	GILL

Common Modules

Next Number PDA (MEXNXNP)

This PDA is made up of three groups:

- INPUT – contains:

  Next Available Number Code
  Last Number Used
  Function

- OUTPUT – contains:

  Next Available Number
  Highest Number In Sequence

- I/O – contains:

  Record ISN
  Update Timestamp

Next Number Object (MEXNXNU)

MEXNXNU is the subroutine described in the overview. It is called by modules in the system that need to determine and update the next available number for a specific key.

Processing

When a process needs to access the next available number file, it does so by sending a valid message to **MEXNXNU**. These messages are kept in LDA **MEXNXNL**, and include #GET-NEXT-NUMBER, #GET-FIRST-IN-SERIES, and #LAST-USED-IN-SERIES, and the function of each is described below.

1) Message #GET-NEXT-NUMBER - This message should be passed when the calling module needs a single next available number for the key code. For example, the online purchase order maintenance screen will be adding a single purchase order at a time, and so will pass this message to this subroutine. This subroutine will read the next available number file to determine, for the passed key code, if there exists a sequence bearing the current user's ID in the user ID field. If there is such a sequence, and there are still numbers available for assignment in it (i.e., the sequence range is not all used up), the subroutine will attempt to use this sequence (see sequence processing below). If there is no such sequence, the subroutine randomly generates a sequence number and will use it (regardless of the user id stored on the record), unless all the numbers in the range have been used. In this last instance, the subroutine will use the next larger sequence number, unless all its numbers have been used, and so forth. In the extreme case in which all the sequences have been all used up, it is not possible to assign a next available number, and the subroutine will return a message to this effect to the calling module.

Sequence processing: Once the subroutine has determined a sequence from which to access a next available number, it simply passes the next available number for the sequence back to the calling module. It also increments the next available number by one and moves the user ID of the calling module to the sequence user ID.

While the next available number record is updated with this information, an END TRANSACTION must be issued by the calling module in order to write the change to the database.

It may happen that the sequence that the subroutine is attempting to process is currently in use by another user. For example, it may happen, in the above diagram, that user REX calls the subroutine to obtain a next available number for PONUMBER. Since REX's user ID is not on any of the sequences, the subroutine randomly generates a sequence for him to reference -- say, sequence 3. It may happen that CAROL currently has that sequence on hold. The subroutine has special processing included to obviate the sending of an error message to REX's module. Instead, REX's call of the subroutine will ignore the sequence, and attempt to use the next larger sequence. If *it* is currently being used (or has been exhausted), the routine will try to use the next higher sequence, and so forth. Note that if REX can use sequence 4, it will overwrite NAT's user ID. User ID's simply provide a means by which the chance of a sequence conflict is greatly decreased. They do not permanently associate a sequence with a user. Coupled together, the special "on hold" processing and the concept of multiple key code sequences enable multiple users to use the next available number processing for the same key code without conflict.

2) Message #GET-FIRST-IN-SERIES - This message should be passed when the calling module plans to need multiple values for the same key. For example, a batch process may need to create a large number of purchase orders. This message causes the subroutine to look up a next available number using sequence processing in the exact same manner as 1) above. However, this message does *not* update the sequence selected. The record is simply placed on hold, and the next available number read is passed back to the calling module, along with the sequence's ISN, its LOG-TIMESTMP, and its highest value. The calling module may use this next available number as a starting point from which to assign as many values (sequentially) as it needs. When it is finished, it should call the routine with the message #LAST-USED-IN-SERIES.

It is the calling module's responsibility, as it assigns value, not to exceed the highest value of the next available number sequence it is referencing.

3) #LAST-USED-IN-SERIES - This message should be passed after the calling module has referenced a next available number via a call with the message #GET-FIRST-IN-SERIES as described above. The calling module must pass in the ISN of the next available number record that was referenced in the above call, and the LOG-TIMESTMP of the record as it was when the sequence was first referenced. This record is retrieved and the record is updated to have the next available number that was passed in by the calling module and to have the calling user's ID.

Next Number Maintenance (MEXNXNF)

The Next Number Maintenance program will be used on the (rare) occasions that next available number sequences need to added, modified, or purged. Furthermore, the browse subprogram associated with this screen can be used to view the next available number sequences and the next available number information for each.

Processing

The maintenance actions refer to a next available number code and all of its sequences, not to a specific sequence. The action of (A)dd, for example, is not used to add a new sequence for an existing next available number code, but to add an entirely new set of sequences for a code for which there is none. The information displayed on the screen pertains to the code and all its sequences. Specific information for the individual sequences (such as next available number within a sequence) can be viewed with the (B)rowse action. This information cannot be modified through this screen. Some of this sequence specific information is set up when the sequences are first created by the (A)dd action. Other information (Next Available Number and User ID) is updated only by the next available number object subprogram when it is invoked by a calling module. (P)urge will delete all the sequences for the specified next available number code. This can only be performed if the sequences have not yet been accessed by the next available number object subprogram. (M)odify will restructure the sequences according to the stipulations on the screen. For example, the number of sequences can be changed from 5 to 10. A key code sequence structure can only be modified if none of the sequences have been accessed by the next available number object subprogram. The (M)odify processing first checks to see that the sequence stipulations made on the screen are valid. If they are, the existing sequences are deleted and then re-added according to these stipulations.

Next Number Browse (MEXNXNB)

As described above, information pertaining to a next number code can be viewed/maintained; sequence specific information (for example, next available number or user ID of last use) cannot. However, the (B)rowse action will enable the user to browse the actual key code sequences and all of the information pertaining to a code's sequence.

Procedures

Program Requirements

To make use of the Next Number Allocation process, a program should contain the following:

- Include the Next Number PDA in the define data area of the calling program.

```
LOCAL USING MEXNXNP /* Next Number object pda
```

- Include the valid message LDA in the define data area of the calling program.

```
LOCAL USING MEXNXNL /* Valid messages pass to Next Avail Number
```

If a module needs a single key value:

- Move the key code to Next Number PDA #NEXT-NUM-CODE
- Move “GET-NEXT-NUMBER” to Next Number PDA #FUNCTION
- Perform the CALLNAT to the Next Number Object.
- Upon return, the Next Number PDA #NEXT-NUMBER is the next available number to be used.
- An END TRANSACTION statement must be issued to update the next available number file. Usually this will be the END TRANSACTION that adds the record for which the key value was retrieved. If it is determined that the record is not to be added, a BACKOUT TRANSACTION could be issued in order to release the next available number record from hold.

If the module will be adding a series of records and needs to know the first available number from which to start:

- Move the key code to Next Number PDA #NEXT-NUM-CODE
- Move “GET-FIRST-IN-SERIES” to Next Number PDA #FUNCTION
- Perform the CALLNAT to the Next Number Object.
- Upon return, the number from which to begin assigning key values will be Next Number PDA #NEXT-NUMBER.
- A second call to the subprogram will need to be made in any of several circumstances (described below). The call should be made as follows:
- Move the key code to Next Number PDA #NEXT-NUM-CODE
- Move the last number assigned to Next Number PDA #LAST-NUM-USED
- Move “LAST-USED-IN-SERIES” to Next Number PDA #FUNCTION
- Make sure that the Next Number PDA #RECORD-ISN and #UPDATE-TIMESTMP are the same for this second call.
- Perform the CALLNAT to the Next Number Object.
- An END TRANSACTION must be issued to update the next available number record.

The second call described above needs to made when any of the following occur:

1. The calling module has finished assigning key values.
2. The calling module is about to issue an END TRANSACTION and has assigned at least one new key value based on the first call to the subprogram.
3. The calling module, in its key assignment process, has assigned the maximum possible (Next Number PDA #HIGHEST-NUM-IN-REC). In this instance, the second call must be made. If the calling module still has more key values it needs to assign, it can then repeat the process by calling again with #GET-FIRST-IN-SERIES and continuing as described above.

Example Program

```
**
** PROGRAM:   SAMPLE ONLINE PROGRAM WHICH NEEDS A SINGLE KEY VALUE.
**            THIS WAS GENERATED USING THE CONSTRUCT MAINT MODEL.
**
DEFINE DATA
...
LOCAL USING MEXNXNP /* Next Number object pda
LOCAL USING MEXNXNL /* Valid messages to pass to Next Avail Number
LOCAL
01 #KEY-CODE (A10) CONST<'PONUMBER'>
...
END-DEFINE
...
...
**SAG DEFINE EXIT AFTER-INPUT
IF #ACTION = #ADD(*)
  IF PO-HEADER-VIEW.PO-NUMBER NE ' '
    ASSIGN #ERROR = TRUE
    ASSIGN #ERROR-NR = 2
    ASSIGN MSG-INFO.##MSG =
      'PO NUMBER WILL BE SYSTEM ASSIGNED AND SO SHOULD NOT BE SPECIFIED'
    PERFORM NEW-SCREEN
  END-IF
  MOVE #KEY-CODE TO #MEXNXNP-IN.#NEXT-NUM-CODE
  CALLNAT 'MEXNXNU'
          #MEXERRP (AD=M)
          #MEXNXNP-IN (AD=O)
          #MEXNXNP-OUT (AD=M)
          #MEXNXNP-IO (AD=M)
          CDPDA-M.MSG-INFO (AD=M)
  IF NOT #MEXNXN-OUT.#NUMBER-FOUND
    ASSIGN #ERROR = TRUE
    ASSIGN #ERROR-NR = 2
    ASSIGN MSG-INFO.##MSG =
      'NEXT AVAILABLE NUMBER COULD NOT BE DETERMINED; CONTACT ADMINISTRATOR'
    BACKOUT TRANSACTION
    PERFORM NEW-SCREEN
  END-IF
  MOVE #MEXNXNP-OUT.#NEXT-NUMBER TO PO-HEADER-VIEW.PO-NUMBER
END-IF
...
**SAG END-EXIT
...
...
**SAG DEFINE EXIT UPDATE-EDITS
...
DECIDE FOR FIRST CONDITION /* determine if there are data errors on the screen
  WHEN PO-HEADER-VIEW.VENDOR-NO = 0
    ASSIGN MSG-INFO.##MSG = 'VENDOR MUST BE SPECIFIED'
  WHEN PO-HEADER-VIEW.DATE-NEEDED NE MASK(MMDDYY)
    ASSIGN MSG-INFO.##MSG = 'A VALID DATE NEEDED (MM/DD/YY) MUST BE SPECIFIED'
...
  WHEN ANY
    IF #ACTION = #ADD(*)
      /*
      /* release the next available number record on hold - it has not been used.
      BACKOUT TRANSACTION
    END-IF
```

```
    ASSIGN #ERROR = TRUE
    PERFORM NEW-SCREEN
END-DECIDE
...
```

```
**
** PROGRAM:   SAMPLE BATCH PROGRAM WHICH WILL USE A SERIES OF KEY VALUES.
**
DEFINE DATA
LOCAL USING MEXNXNP /* Next Available Number object pda
LOCAL USING MEXNXNL /* Valid messages to pass to Next Avail Number
LOCAL
01 #KEY-CODE (A10) CONST<'PONUMBER'>
...
END-DEFINE
...
...
READ DRIVER-FILE BY DRIVER-KEY
...
...
  /*
  /*  WE ARE NOW READY TO STORE THE PO HEADER VIEW,
  /*  EXCEPT WE NEED THE PO NUMBER
  /*
  PERFORM GET-NUMBER-TO-USE
  MOVE #NUMBER-TO-USE TO PO-HEADER.PO-NUMBER
  STORE PO-HEADER
  ADD 1 TO #ET-UPDATE COUNT
  ADD 1 TO #ET-READ-COUNT
  PERFORM CHECK-IF-NEED-TO-ET
...
...
END-READ /* (of driver file)
...
/*
DEFINE SUBROUTINE GET-NUMBER-TO-USE
  /*
  IF #NUMBER-TO-USE = 0  /*  Need to retrieve next available number
    PERFORM GET-FIRST-IN-SERIES
    ESCAPE ROUTINE
  END-IF
  IF #NUMBER-TO-USE = #NUMBER-MAX-TO-USE
    PERFORM LAST-USED-IN-SERIES
    PERFORM GET-FIRST-IN-SERIES
    ESCAPE ROUTINE
  END-IF
  ADD 1 TO #NUMBER-TO-USE
END-SUBROUTINE
/*
DEFINE SUBROUTINE GET-FIRST-IN-SERIES
  MOVE #KEY-CODE TO #MEXNXNP-IN.#NEXT-NUM-CODE
  MOVE #MEXNXNL.#GET-FIRST-IN-SERIES TO #MEXNXNP-IN.#FUNCTION
  CALLNAT 'MEXNXNU'
          #MEXERRP (AD=M)
          #MEXNXNP-IN (AD=O)
          #MEXNXNP-OUT (AD=M)
          #MEXNXNP-IO (AD=M)
          CDPDA-M.MSG-INFO (AD=M)
  ADD 1 TO #ET-UPDATE-COUNT
  ADD 1 TO #ET-READ-COUNT
  ASSIGN #NEXT-AVAIL-NUM-ON-HOLD = TRUE
  MOVE #MEXNXNP-OUT.#NEXT-NUMBER TO #NUMBER-TO-USE
  MOVE #MEXNXNP-IO.#RECORD-ISN TO #NEXT-AVAIL-NUM-SEQ-ISN
  MOVE #MEXNXNP-IO.#UPDATE-TIMESTMP TO #NEXT-AVAIL-NUM-SEQ-TIMESTMP
  MOVE #MEXNXNP-I0.#HIGHEST-NUM-IN-REC TO #NUMBER-MAX-TO-USE
  PERFORM CHECK-IF-NEED-TO-ET
END-SUBROUTINE
/*
```

```
DEFINE SUBROUTINE LAST-USED-IN-SERIES
  MOVE #KEY-CODE TO #MEXNXNP-IN.#NEXT-NUM-CODE
  MOVE #NUMBER-TO-USE TO #MEXNXNP-IN.#LAST-NUM-USED
  MOVE #NEXT-AVAIL-NUM-SEQ-ISN TO #MEXNXNP-IO.#RECORD-ISN
  MOVE #NEXT-AVAIL-NUM-SEQ-TIMESTMP TO #MEXNXNP-IO.#UPDATE-TIMESTMP
  MOVE #MEXNXNL.#LAST-USED-IN-SERIES TO #MEXNXNP-IN.#FUNCTION
  CALLNAT 'MEXNXNU'
          #MEXERRP (AD=M)
          #MEXNXNP-IN (AD=O)
          #MEXNXNP-OUT (AD=M)
          #MEXNXNP-IO (AD=M)
          CDPDA-M.MSG-INFO (AD=M)
  ADD 1 TO #ET-UPDATE-COUNT
  ADD 1 TO #ET-READ-COUNT
  RESET #NEXT-AVAIL-NUM-ON-HOLD
  RESET #NUMBER-TO-USE #NUMBER-MAX-TO-USE
    #NEXT-AVAIL-NUM-SEQ-ISN #NEXT-AVAIL-NUM-SEQ-TIMESTMP
  PERFORM CHECK-IF-NEED-TO-ET
END-SUBROUTINE
/*
DEFINE SUBROUTINE CHECK-IF-NEED-TO-ET
  /*
  IF #ET-READ-COUNT GE #ET-READ-LIMIT OR
      #ET-UPDATE-COUNT GE #ET-UPDATE-LIMIT
    IF #NEXT-AVAIL-NUM-ON-HOLD
      PERFORM LAST-USED-IN-SERIES
    END-IF
    ASSIGN #RESTART-KEY = DRIVER-FILE.DRIVER-KEY
    MOVE BY NAME #RUN-CONTROL TO #MEXRUNP-IO
    PERFORM ISSUE-PERIODIC-ET
  END-IF
END-SUBROUTINE
```

Next Number (sample DDM)

```
  1  AR  AVAIL-NO-CODE                   A   10  N    * NUMBER TYPE CODE F
  1  AS  AVAIL-NO-SEQ                    N  4.0  N    * SEQUENCE OF RECORD
  1  AT  USERID-LAST-USE                 A    8  N    * WHO GOT LAST NUMBE
  1  AU  KEY-RANGE-LOW                   N 10.0  N    * LOWEST NUMBER IN R
  1  AV  KEY-RANGE-HIGH                  N 10.0  N    * HIGHEST NUMBER AVA
  1  AW  REC-RANGE-LOW                   N 10.0  N    * LOWEST NUMBER FOR
  1  AX  REC-RANGE-HIGH                  N 10.0  N    * HIGHEST NUMBER FOR
  1  AY  NEXT-AVAIL-NO                   N 10.0  N    * NEXT AVAILABLE NUM
  1  AZ  RANGE-USED-FLAG                 A    1  N    * "N" = NOT ALL USED
  1  A0  NO-OF-RANGES                    N  4.0  N    * COUNT OF RECORDS F
G 1  A1  LOG-FIELDS                                   * LOG INFORMATION FI
  2  A2  LOG-ACTION                      A    1  N    *  LAST DATABASE ACT
  2  A3  LOG-TIMESTMP                    B    8  N    * *TIMESTMP
  2  A4  MAIN-PGM                        A    8  N    * MAIN CALLING PROGR
  2  A5  SUB-MODULE                      A    8  N    * LAST SUBMODULE WHI
  2  A6  LOG-SNAPSHOT-NO                 N  8.0  N    * SYSTEM SNAPSHOT NU
G 1  A9  LAST-FM-STAMP                                * LAST FILE MAINTENA
  2  BA  LAST-FM-DATE                    N  8.0  N    * LAST FILE MAINTENA
  2  BB  LAST-FM-TIME                    N  6.0  N    * LAST FILE MAINTENA
  2  BC  LAST-FM-USER                    A    8  N    * NATURAL USERID OF
G 1  BD  CREATION-STAMP                               * RECORD CREATION ST
  2  BE  CREATION-DATE                   N  8.0  N    * DATE RECORD ADDED
  2  BF  CREATION-TIME                   N  6.0  N    * TIME RECORD ADDED
  2  BG  CREATION-USER                   A    8  N    * NATURAL USERID OF
  1  AO  SP-AVAIL-NO-SEQ                 A   14  N S  * UNIQUE PRIME KEY F
         HD=AVAILABLE/NUMBER/SEQUENCE
*        -------- SOURCE FIELD(S) -------
*        AVAIL-NO-CODE(1-10)
*        AVAIL-NO-SEQ(1-4)
  1  AP  SP-AVAIL-NO-USERID-RANGE-USED   A   19  N S  * AVAILABLE NUMBER U
         HD=AVAILABLE NO/USERID RANGE/USED
*        -------- SOURCE FIELD(S) -------
*        AVAIL-NO-CODE(1-10)
*        USERID-LAST-USE(1-8)
*        RANGE-USED-FLAG(1-1)
  1  AQ  SP-AVAIL-NO-CODE-SEQ-RANGE-USED A   15  N S  * AVAILABLE NUMBER C
         HD=AVAILABLE NO/CODE SEQUENCE/RANGE USED
*        -------- SOURCE FIELD(S) -------
*        AVAIL-NO-CODE(1-10)
*        AVAIL-NO-SEQ(1-4)
*        RANGE-USED-FLAG(1-1)
```

Random Number Generator

This section of the document has been prepared as a guide to understanding the Random Number Generator and the procedures that need to be followed to implement its use. A common module has been created for use by both batch and on-line processes that require a system assigned number. An overview of the Random Number Generator along with a description of the common modules is detailed below.

Overview

Random numbers are generated to provide uniformly distributed unique identifiers to objects in the MEDS system. When an object subprogram is adding a new record to the MEDS system, it simply requests a new number from the Random Number Generator common subprogram (no input parameter is required). The responsibility of verifying that the generated number is unique for the object is still the responsibility of the calling program. In the rare case that the generated number has already been used, the Random Number Generator subprogram should be called again.

Common Modules

Random Number PDA (MEXRNDP)

This PDA contains the fields required by the random number generator common subprogram.

```
#MEXRND-IN
  #NULL (A1)
#MEXRNDP-OUT
  #RANDOM-NBR (N11) /* N7, N8, N9, N10, OR N11 available
#MEXRNDP-IO
  #NULL(A1)
```

Generate Random Number (MEXRNDN)

Generate Random Number is a common subprogram that returns to the calling module a randomly generated number. Object subprograms will use this routine to generate system assigned numbers.

Example Program

```
...
...
*
GET-KEY-VALUE.
REPEAT
  /*
  CALLNAT “MEXRNDN”  /* Random Number Generator
          #MEXERRP (AD=M)
          #MEXRNDP-IN (AD=O)
          #MEXRNDP-OUT (AD=M)
          #MEXRND-IO (AD=M)
  IF #MEXERRP-OUT.#ERROR
    PERFORM ESCAPE-PROG
  END-IF
  /*
  ASSIGN #RECORD.KEY = #MEXRNDP-OUT.#RANDOM-NBR
  /*
  PERFORM UNIQUE-CHECK
  IF #ERROR
    ESCAPE TOP
  END-IF
  /*
END-REPEAT /* (GET-KEY-VALUE)
*
...
...
```

Random Number Generator (sample algorithm)

```
0130 DEFINE DATA
0140 PARAMETER USING MEXRNDP
0150 *
0160 INDEPENDENT
0170 01 +NUMBER (N22.7)
0180 *
0190 LOCAL
0200 01 #LOCAL
0210   02 #NUMBER (N22.7)
0220   02 REDEFINE #NUMBER
0230     03 #X1 (N9 )
0240     03 #X2 (N11)
0250   02 REDEFINE #NUMBER
0260     03 #P1 (N1)
0270     03 #P2 (N3)
0280     03 #P3 (N4)
0290     03 #P4 (N2)
0300     03 #P5 (N6)
0310     03 #P6 (N3)
0320     03 #P7 (N4)
0330     03 #P8 (N4)
0340     03 #P9 (N2)
0350   02 #M (N22.7)
0360   02 #N (N22.7)
0370   02 #NUM-N4 (N4)
0380   02 #NUM-N3 (N3)
0390 01 #TIMESTAMP (B8)
0400 01 REDEFINE #TIMESTAMP
0410   02 #FILLER1 (A4)
0420   02 #VALUE (I4)
0430 01 #TEMP-VALUE (N10)
0440 01 REDEFINE #TEMP-VALUE
0450   02 #FILLER2 (N3)
0460   02 #SEED (N7)
0470 END-DEFINE
0480 *
0490 IF +NUMBER = 0
0500   ASSIGN #TIMESTAMP = *TIMESTMP
0510   IF #VALUE < 0
0520     ASSIGN #TEMP-VALUE = #VALUE * -1
0530   ELSE
0540     ASSIGN #TEMP-VALUE = #VALUE
0550   END-IF
0560   ASSIGN +NUMBER = #SEED
0570   COMPUTE +NUMBER = +NUMBER * 987987987987987
0580 END-IF
0590 *
0600 ASSIGN #M = 1.0000001
0610 ASSIGN #N = 0.9999999
0620 *
0630 ASSIGN #NUMBER = +NUMBER
0640 COMPUTE #NUMBER = #NUMBER * #N
0650 ASSIGN #NUM-N4 = #P3
0660 ASSIGN #P3 = #P8
0670 ASSIGN #P8 = #NUM-N4
0680 *
0690 COMPUTE #NUMBER = #NUMBER * #M
```

```
0700 ASSIGN #NUM-N3 = #P2
0710 ASSIGN #P2 = #P6
0720 ASSIGN #P6 = #NUM-N3
0730 *
0740 ASSIGN #NUMBER = +NUMBER
0750 COMPUTE #NUMBER = #NUMBER * 0.9998999
0760 ASSIGN #NUM-N4 = #P3
0770 ASSIGN #P3 = #P8
0780 ASSIGN #P8 = #NUM-N4
0790 *
0800 COMPUTE #NUMBER = #NUMBER * 1.0123451
0810 ASSIGN #NUM-N3 = #P2
0820 ASSIGN #P2 = #P6
0830 ASSIGN #P6 = #NUM-N3
0840 *
0850 ASSIGN #P1 = 0
0860 ASSIGN +NUMBER = #NUMBER
0870 ASSIGN #CXXANNN1-OUT.RANDOM_NBR = #X2
0880 *
0890 END
```

Error Handler

This document has been prepared as a guide to understanding the Error Handler and the procedures that need to be followed to implement its use. Several common modules have been created for use by both batch and on-line processes that require error-handling capabilities. An overview of the Error Handler along with an explanation of the common modules and how to use them are detailed below.

Overview

NATURAL provides two distinct facilities to build an Error Handler that can be used in both on-line and batch programs. Either one can be utilized to automatically call the Error Handler routine (XXX0000) whenever an error occurs. They are:

1. The ON ERROR concept
2. The *ERROR-TA concept.

MEDS uses the *ERROR-TA concept. To use this concept, the start-up program of the system must assign the system variable *ERROR-TA the name of the Error Handler program (XXX0000). When a NATURAL error occurs, the program contained in *ERROR-TA is invoked.

Common Modules

Error Handler Object (MEXERRZ)

MEXERRZ was created to store NATURAL errors, program detected fatal errors, and program detected warnings on the error log file (ERROR-LOG). Except for two conditions, whenever the Error Handler is invoked, a record describing the error is logged on the Error Log file.

One of the times an error is not logged is when it is a NATURAL 1701 error and it occurs in an on-line process. A NATURAL 1701 error occurs when a "non-activity time limit is exceeded". The other time an error is not logged is when it is a NATURAL 3009 error. A NATURAL 3009 occurs when an ADABAS time limit is exceeded.

When an on-line or batch process encounters an error that should cause the process to stop or should cause a warning to be logged, it passes control and information about the error to the Error Handler. The Error Handler uses the information passed to determine what must be done. The "rules" it uses are as follows:

- Any NATURAL error is fatal.
- A program-detected error is fatal when the #FATAL-ERROR flag is "true". Otherwise, it is a warning.

When a warning is encountered, the following five steps occur:

1. The logical transaction that was being processed is NOT backed out of the database.
2. The error information is stored in the Error Log file (no ET is issued).
3. #RECORD-HOLD-COUNT is incremented by 1.
4. #WARNINGS-LOGGED is set to "true".
5. Control is passed back to the object where the warning occurred.

When a NATURAL or fatal error is encountered, the following four steps occur:

1. The logical transaction currently being processed is backed out of the database.
2. The error information is stored in the Error Log file and an End Transaction is issued.
3. An error screen explaining what caused the error is either popped-up (on-line) or printed (batch).
4. Control is passed back to the main menu (on-line) or to the system ABEND generator program (batch)

Below are examples of what the error screens look like.

Error Screen – On-line

```
+-------------------------------------------------------------------------+
|  THE FOLLOWING SYSTEM ERROR HAS OCCURRED:                               |
|  ERROR    :                                                             |
|   MORE THAN ONE RECORDS EXISTS FOR THE INPUT KEY (CUSTOMER-NO)          |
|  AT LINE  : 2160  IN MODULE: XXXXXXXX                                   |
|  AT LEVEL : 2     TYPE     : PGM (PGM=PROGRAM, NAT=NATURAL)             |
|  PGM TRACE: XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX      |
|             XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX      |
|             XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX      |
|                  ----+----1----+----2----+----3----+----4----+----5 |
|  RECORD KEY(0-50):                                                      |
|           (51-100):                                                     |
|          (101-150):                                                     |
|          (151-200):                                                     |
|          (201-250):                                                     |
|  USER DATA (0-50):                                                      |
|           (51-100):                                                     |
|          (101-150):                                                     |
|          (151-200):                                                     |
|          (201-250):                                                     |
+-------------------------------------------------------------------------+
```

Error Form – Batch

```
THE FOLLOWING SYSTEM ERROR HAS OCCURRED:
ERROR    :
 MORE THAN ONE RECORDS EXISTS FOR THE INPUT KEY (CUSTOMER-NO)
AT LINE  : 2160  IN MODULE: XXXXXXXX
AT LEVEL : 2     TYPE     : PGM (PGM=PROGRAM, NAT=NATURAL)
PGM TRACE: XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
           XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
           XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
                ----+----1----+----2----+----3----+----4----+----5
RECORD KEY(0-50):
        (51-100):
       (101-150):
       (151-200):
       (201-250):
USER DATA (0-50):
        (51-100):
       (101-150):
       (151-200):
       (201-250):
```

Error Handler PDA (MEXERRP)

This PDA contains four parameters used by the Error Handler:

1. Main Program
2. Warnings Logged (true/false)
3. Record Hold Count
4. Program Trace (last 20 modules)

Local Error Handler LDA (MEXERRL)

This LDA contains constants representing program detected errors - "PGM", and NATURAL errors - "NAT". It also contains variables which are used to hold the error information that is passed to the Error Handler.

Local Error Handler Copycode (MEXERRC0)

Copycode to be included as one of the last lines in all programs and subprograms. This copycode contains the subroutine LOCAL-ERROR-HANDLER that stacks the error information *(on the NATURAL STACK)* and then passes control to *(FETCH RETURN)* the Error Handler Object.

Error Handler Level 1 Startup Copycode (MEXERRC1)

Copycode to be included as the first executable line in all level one programs. This copycode loads the variable #MAIN-PROGRAM to be used later in case of an error.

Error Handler Startup Copycode (MEXERRC2)

Copycode to be included as the first executable line in all programs (except Level 1 programs in which case it would be the second executable line). This copycode loads the variable *PROGRAM into the #PROGRAM-TRACE array to be used later in case of an error.

Procedures

Program Requirements

To make use of the Error Handler logic, a process should contain the following:

1. In the start-up program for the subsystem, assign system variable *ERROR-TA the name of the Error Handler program – **MEXERRZ**. This is the name of the NATURAL program that receives control in case of an error.

```
ASSIGN *ERROR-TA = 'MEXERRZ'
```

2. Include the copycode **MEXERRC0** as one of the last executable lines in all programs and subprograms. This copycode contains the subroutine LOCAL-ERROR-HANDLER, which stacks the required error information and passes control to the Error Handler.

```
INCLUDE MEXERRC0 /* Subroutine LOCAL-ERROR-HANDLER
```

3. Include the copycode **MEXERRC1** as the first executable line in all level one programs. This copycode loads the variable #MAIN-PROGRAM to be used later in case of an error.

```
INCLUDE MEXERRC1 /* Loads #MAIN-PROGRAM
```

4. Include the copycode **MEXERRC2** as the second executable line in all level one programs and the first executable line in all non-level one programs. This copycode loads the variable *PROGRAM into #PROGRAM-TRACE array to be used later in case of an error.

```
INCLUDE MEXERRC2 /* Loads #MAIN-PROGRAM
```

5. Include the LDA **MEXERRL** in all modules. This LDA contains all the variables that need to be loaded in case of program detected errors or warnings.

```
LOCAL USING MEXERRL /* Error Handler variables
```

6. Include the PDA **MEXERRP** as a LDA/PDA in all calling modules and as a PDA in all called modules. Make sure that CALLNAT statements and subprograms **in all systems that use the Error Handler** include this PDA as the first parameter.

7. When program detected errors or warnings are encountered, load the error variables and perform the error handler. The variables to be assigned are:

```
#MEXERRL.#LOCAL-ERROR-LINE /* Line number where the error occurred (e.g.(0330))
#MEXERRL.#ERROR-MESSAGE    /* Error message to be shown to the user.
#MEXERRL.#RECORD-KEY       /* Key info of the record when the error occurred.
#MEXERRL.#USER-DATA        /* Additional information to be shown to the user.
#MEXERRL.#FATAL-ERROR      /* Logical field to distinguishes between warnings and
errors.
```

Example Programs

Program X CALLNAT's Subprogram Y that contains code to log a fatal error if state code is not a valid code. If a NATURAL error occurs in either one, **MEXERRZ** will be invoked via the *ERROR-TA system variable.

```
PROGRAM X
DEFINE DATA
LOCAL USING MEXERRL /* Local Error Handler Variables
LOCAL USING MEXERRP /* Error Handler Object PDA
LOCAL
……
END-DEFINE
INCLUDE MEXERRC1 /* Build #MAIN-PROGRAM
INCLUDE MEXERRC2 /* Build #PGM-TRACE
……
……
CALLNAT “SUBPROGRAM Y”
        #MEXERRP
……
……
INCLUDE MEXERRC0 /* LOCAL-ERROR-HANDLER subroutine
END

SUBPROGRAM Y

DEFINE DATA
PARAMETER USING MEXERRP /* Error Handler Object PDA
……
……
LOCAL
LOCAL USING MEXERRL /* Local Error Handler Variables
……
……
END-DEFINE
INCLUDE MEXERRC2 /* Build #PGM-TRACE
……
……
IF CUST-FILE.STATE-CODE is not valid
  ASSIGN #MMXERRL.#LOCAL-ERROR-LINE = (0400)
  ASSIGN #MMXERRL.#ERROR-MSG = “INVALID STATE”
  COMPRESS “CUST#:” CUST-FILE.CUST-NO
      INTO #MMXERRL.#RECORD-KEY
  ASSIGN #MMXERRL.#USER-DATA = CUST-FILE.STATE-CODE
  ASSIGN #MMXERRL.#FATAL-ERROR = TRUE
  PERFORM LOCAL-ERROR-HANDLER
END-IF
……
INCLUDE MEXERRC0 /* LOCAL-ERROR-HANDLER subroutine
END
```

Error Log (sample DDM)

```
   1  AA  SP-USER-INVERTED-DATE-TIME       A   23  N S  SUPER DESCRIPTOR OF
          HD=USER/INVERTED/DATE-TIME
 *        -------- SOURCE FIELD(S) -------
 *        LOGON-ID(1-8)
 *        ERROR-INVERTED-DATE(1-8)
 *        ERROR-INVERTED-TIME(1-7)
   1  AB  ERROR-NO                         N  7.0  N    NATURAL ERROR NUMBER
   1  AC  ERROR-MESSAGE                    A   65  N    PROGRAM DETECTED ERR
   1  AD  ERROR-LINE                       N  4.0  N    NATURAL LINE NUMBER
   1  AE  APPL-ID                          A    8  N    NATURAL LIBRARY ID
   1  AF  MAIN-PGM                         A    8  N    MAIN CALLING PROGRAM
   1  AG  SUB-MODULE                       A    8  N    MODULE IN ERROR
   1  AH  ERROR-CALL-LVL                   N  2.0  N    NATURAL CALL LEVEL
   1  AI  ERROR-INVERTED-DATE              N  8.0  N    DATE, 9'S COMPLEMENT
   1  AJ  ERROR-INVERTED-TIME              N  7.0  N    TIME, 9'S COMPLEMENT
   1  AK  LOGON-ID                         A    8  N    USERID
   1  AL  RECORD-IN-PROCESS-KEY            A  250  N    IDENTIFIER OF THE RE
   1  AM  USER-DATA                        A  250  N    MISCELLANEOUS DATA
   1  AN  LOG-TIMESTMP                     B    8  N    *TIMESTMP
   1  AO  ERROR-TYPE                       A    3  N    ERROR TYPE CODE:
   1  AP  ERROR-SEVERITY-CODE              A    1  N    FATAL OR WARNING SEV
   1  AQ  CO-NO                            N  2.0       * UNIQUE COMPANY NUM
   1  AR  DIV-NO                           N  2.0       * DIVISION NUMBER
 G 1  AS  CREATION-STAMP                                 * RECORD CREATION ST
   2  AT  CREATION-DATE                    N  8.0       * DATE RECORD ADDED
   2  AU  CREATION-TIME                    N  6.0       * TIME RECORD ADDED
   2  AV  CREATION-USER                    A    8       * NATURAL USERID OF
 G 1  AW  LAST-FM-STAMP                                  * LAST FILE MAINTENA
   2  AX  LAST-FM-DATE                     N  8.0       * LAST FILE MAINTENA
   2  AY  LAST-FM-TIME                     N  6.0       * LAST FILE MAINTENA
   2  AZ  LAST-FM-USER                     A    8       * NATURAL USERID OF
```

Code Translator

This section of the document describes common routines to translate codes from one system to another.

Overview

The codes used to describe gender, race, and social security verification in the MEDS system may not be the same as the codes in other systems MEDS interfaces with. The code translators have as input the system form which the code is to be translated, the system into which the code is to be translated, and the code. As an output, the code translator returns the tranlated code.

Common Modules

Translator PDA (MEXORCP)

This PDA contains the fields required by the translating common subprogram.

```
01 #METORCP-IN
  02 #SYSTEM-IN  (A8)
  02 #SYSTEM-OUT (A8)
  02 #CODE-ALPHA (A10)
  02 #CODE-NUMERIC (N10)
*
01 #METORCP-OUT
  02 #CODE-ALPHA (A10)
  02 #CODE-NUMERIC (N10)
```

Sex Code Translator (MEXSXCN)

The Sex Code Translator is a common subprogram to tranlate sex codes. Interface programs will use this subprogram.

Race Code Translator (MEXRCCN)

The Race Code Translator is a common subprogram to tranlate race codes. Interface programs will use this subprogram.

Social Security Verification Code Translator (MEXSVCN)

The Social Security Verification Code Translator is a common subprogram to tranlate Social Security Verification codes. Interface programs will use this subprogram.

Example Program

```
...
...
*
LOCAL USING MEXORIL  /* Common LDA for Originating System Indicator for any process
LOCAL USING MEXORCP  /* PDA for Common Code Translator
LOCAL USING CXXGNLP  /* Common Error Messaging
*

REPEAT
  /*
  RESET #MEXORCP-IN
  ASSIGN #MEXORCP-IN.#SYSTEM-IN = #MEXORIL.#MEDS
  ASSIGN #MEXORCP-IN.#SYSTEM-OUT = #MEXORIL.#LASES
  ASSIGN #MEXORCP-IN.#CODE-NUMERIC = #METRCEL.#BLACK
  /*
  CALLNAT 'MEXRCCN'  /* Race Translator
          #MEXORCP-IN
          #MEXORCP-OUT
          #CXXGNLP-OUT
  IF #CXXGNLP-OUT.#ERROR
    WRITE #CXXGNLP-OUT.#ERROR-MSG
    PERFORM ESCAPE-PROG
  END-IF
  /*
  ASSIGN #RECORD.#LASES-RACE = #MEXORCP0-OUT.#CODE-ALPHA
  /*
END-REPEAT /* (GET-KEY-VALUE)
*
...
...
```

Race Code Translator (sample algorithm)

```
0140 DEFINE DATA
0150 PARAMETER USING MEXORCP  /* Program parameters
0160 PARAMETER USING CXXGNLP  /* Common error PDA
0170 *
0180 LOCAL USING METRCEL  /* MEDS Race Code Table
0190 LOCAL USING MEISESL3  /* LASES Race Code Table
0200 LOCAL USING MEXORIL  /* Originating System Code Table
0230 *
0240 END-DEFINE
0250 *
0260 *
0270 **********************
0280 * Main Program Logic *
0290 **********************
0300 *
0310 PROG.
0320 REPEAT
0330   /*
0340   PERFORM INTIALIZE-VARIABLES
0350   /*
0360   /* First decide what type of input we have, then
0370   /* decide what type of output we will return.
0380   /*
0390   DECIDE ON FIRST #MEXORCP-IN.#SYSTEM-IN
0400     /*
0410     VALUE #MEXORIL.#MEDS /* incoming MEDS
0420       /*
0430       DECIDE ON FIRST #MEXORCP-IN.#SYSTEM-OUT
0440         VALUE #MEXORIL.#LASES /* outgoing LASES
0450           PERFORM CONVERT-MEDS-TO-LASES
0460         NONE
0470           PERFORM ESCAPE-WITH-ERROR
0480       END-DECIDE /* #MEXORCP-IN.#SYSTEM-OUT
0490       /*
0500       /*
0510     VALUE #MEXORIL.#LASES /* incoming MEDS
0520       /*
0530       DECIDE ON FIRST #MEXORCP-IN.#SYSTEM-OUT
0540         VALUE #MEXORIL.#MEDS /* outgoing MEDS
0550           PERFORM CONVERT-LASES-TO-MEDS
0560         NONE
0570           PERFORM ESCAPE-WITH-ERROR
0580       END-DECIDE /* #MEXORCP-IN.#SYSTEM-OUT
0590       /*
0600       /*
0610     NONE
0620       PERFORM ESCAPE-WITH-ERROR
0630   END-DECIDE /* #MEXORCP-IN.SYSTEM-IN
0640   /*
0650   PERFORM ESCAPE-PROGRAM
0660   DEFINE SUBROUTINE ESCAPE-PROGRAM
0670     ESCAPE BOTTOM (PROG.)
0680   END-SUBROUTINE /* ESCAPE-PROGRAM
0690 END-REPEAT /* (PROG.)
0700 *
0710 ************************************************************************
0720 DEFINE SUBROUTINE INTIALIZE-VARIABLES /* IV
0730 ************************************************************************
```

```
0740 RESET #CXXGNLP-OUT
0750 RESET #MEXORCP-OUT
0760 END-SUBROUTINE /* INTIALIZE-VARIABLES
0770 *
0780 ************************************************************************
0790 DEFINE SUBROUTINE ESCAPE-WITH-ERROR /* EWE
0800 ************************************************************************
0810 ASSIGN #CXXGNLP-OUT.#ERROR = TRUE
0820 COMPRESS #MEXORCP-IN.#SYSTEM-IN 'to'
0830          #MEXORCP-IN.#SYSTEM-OUT 'translation undefined in '
0840          *PROGRAM
0850     INTO #CXXGNLP-OUT.#ERROR-MSG
0860 PERFORM ESCAPE-PROGRAM
0870 END-SUBROUTINE /* ESCAPE-WITH-ERROR
0880 *
0890 ************************************************************************
0900 DEFINE SUBROUTINE CONVERT-MEDS-TO-LASES /* CMTL
0910 ************************************************************************
0920 DECIDE ON FIRST #MEXORCP-IN.#CODE-NUMERIC
0930   VALUE #METRCEL.#BLACK
0940     ASSIGN #MEXORCP-OUT.#CODE-ALPHA = #METSRCL.#BLACK
0950   VALUE #METRCEL.#WHITE
0960     ASSIGN #MEXORCP-OUT.#CODE-ALPHA = #METSRCL.#WHITE
0970   VALUE #METRCEL.#OTHER
0980     ASSIGN #MEXORCP-OUT.#CODE-ALPHA = #METSRCL.#OTHER
0990   VALUE #METRCEL.#UNKNOWN
1000     ASSIGN #MEXORCP-OUT.#CODE-ALPHA = #METSRCL.#OTHER
1010   NONE
1020     IGNORE
1030 END-DECIDE /* #MEXORCP-IN.#CODE-NUMERIC
1040 END-SUBROUTINE /* CONVERT-MEDS-TO-LASES
1050 *
1060 ************************************************************************
1070 DEFINE SUBROUTINE CONVERT-LASES-TO-MEDS /* CLTM
1080 ************************************************************************
1090 DECIDE ON FIRST #MEXORCP-IN.#CODE-ALPHA
1100   VALUE #METSRCL.#BLACK
1110     ASSIGN #MEXORCP-OUT.#CODE-NUMERIC = #METRCEL.#BLACK
1120   VALUE #METSRCL.#WHITE
1130     ASSIGN #MEXORCP-OUT.#CODE-NUMERIC = #METRCEL.#WHITE
1140   NONE
1150     ASSIGN #MEXORCP-OUT.#CODE-NUMERIC = #METRCEL.#UNKNOWN
1160 END-DECIDE /* #MEXORCP-IN.#CODE-ALPHA
1170 END-SUBROUTINE /* CONVERT-LASES-TO-MEDS
1180 *
1190 ************************************************************************
1220 *
1230 END
```