

ISPF Application Technique with REXX

Frank Clarke
The Nielsen Company

Introduction

- We will build a fresh table and show how rows can be added, changed, and deleted.
- The table will hold information about other ISPF tables for use by other applications.
- It's a simple table – 1 key field and 5 non-key fields.

What the screens should look like

- The main display:

```
----- AAMSTR Table Selection ----- Row 1 to 17 of 17
COMMAND ===>
                                                    SCROLL ===> CSR
  /-   B = Browse, E,U = Change, I = Insert (new)
  /
V  ID   Tbl Name      Description
AA   AAMSTR      Master Table
CF   CONFIG      Configuration Management Elements
CI   CFGINC      Configuration INCLUDEs
CP   COMPARM     Compile Parameters
FE   $FETCH      Fetchable DSNames
JT   JOBTRK      Track submitted jobs
      (etc.)
```

What the screens should look like

- Browse/Edit:

```
----- AAMSTR Table Update -----  
COMMAND ==>  
  
Table ID ==> PM          (xx)  
Table Name ==> PGMSTR    (xxxxxxxx)  
Description ==> Program Master  
  
Key Fields ==> PMKEY  
  
Name Fields ==> PMROOT PMVER PMTYPE PMDESC PMADDDT PMADDID PMCHGDT  
PMCHGID PMACQDT PMACQID PMLOCKED PMSTYLE PMARVER  
  
Sort Sequence ==> PMKEY,C,A
```

What the screens should look like

- Insert:

```
----- AAMSTR Table Update -----  
COMMAND ==>  
  
Table ID ==> (xx)  
Table Name ==> (xxxxxxxx)  
Description ==>  
  
Key Fields ==>  
  
Name Fields ==>  
  
Sort Sequence ==>  
  
SCROLL ==> CSR
```

Basic Outline

- Open the table
 - What?? It doesn't exist yet? But....!
- Display the table
 - Process rows selected for add, change, delete
- Save the table

Services

- LIBDEF
 - Dynamic modification of the search order for ISPF assets.
- TBSTATS
 - Part of Table Services.
 - Provides information about a table.
- TBOPEN
 - Makes the table ready-for-use.
- TBTOP
 - Points to row #1.

Opening the table...

```
/*
  Open the table; initialize as necessary.
. ----- */
BA_OPEN:          /*@          */
  address ISPEXEC

  "LIBDEF ISPTLIB DATASET ID("isptlib") STACK"
  "TBSTATS" $tn$ "STATUS1(s1) STATUS2(s2)"
  if s1 > 1 then do          /* table not found          */
    call BAA_INIT_MSTR      /* Build a new AAMSTR table -*/
  end; else,
  if s2 = 1 then do
    "TBOPEN " $tn$ "WRITE"
  end
  else "TBTOP" $tn$
  "LIBDEF ISPTLIB"

return          /*@ BA_OPEN          */
```


What does TBSTATS tell us?

- **status1-name** (STATUS1)
 - Specifies the name of a variable where the status of the table in the **table input library chain** is to be stored. Values that can be stored and their meanings are:
 - 1 -- table exists in the table input library chain
 - 2 -- table does not exist in the table input library chain
 - 3 -- table input library is not allocated
- **status2-name** (STATUS2)
 - Specifies the name of a variable where the status of the table in this **logical screen** is to be stored. Values that can be stored and their meanings are:
 - 1 -- table is not open in this logical screen
 - 2 -- table is open in NOWRITE mode in this logical screen
 - 3 -- table is open in WRITE mode in this logical screen
 - 4 -- table is open in SHARED NOWRITE mode in this logical screen
 - 5 -- table is open in SHARED WRITE mode in this logical screen
- **status3-name** (STATUS3)
 - Specifies the name of a variable where the availability of the table to be used in **WRITE mode** is to be stored. Values that can be stored and their meanings are:
 - 1 -- table is available for WRITE mode
 - 2 -- table is not available for WRITE mode

Does the table exist ?

```
if s1 > 1 then do                                /* table not found      */
  call BAA_INIT_MSTR                             /* Build a new AAMSTR table -*/
end;
```

If STATUS1 indicates the table does not (yet) exist, we'll need to build the initial table....

Is it open ?

```
... else,  
  if s2 = 1 then do  
    "TBOPEN" $tn$ "WRITE"  
  end
```

The table exists, but STATUS2 tells us it isn't yet open...

It's already open ?

```
else "TBTOP" $tn$
```

If the table is already open when we start, force the cursor to the first row...

More table services...

- TBCREATE
 - Generates an *empty* table in storage
- TBADD
 - Adds a new row to the table in storage

TBCREATE

```
"TBCREATE" $tn$ "KEYS(AATBLID)",  
                "NAMES(AATBLNM AAKEYS AANAMES AASORT AADESC)"
```

After loading, the first row of the table will look (logically) like this:

```
AATBLID   AATBLNM  AAKEYS   AANAMES _____.  
  
AA        AAMSTR   AATBLID   AATBLNM AAKEYS AANAMES AASORT AADESC  
  
... AASORT   AADESC   .  
    AATBLID,C,A   Master Table
```

That is: *this* row describes its own table.

Displaying the table...

The panel code for the scrollable display:

```
-----  
)ATTR  
  % TYPE(TEXT)      INTENS(HIGH) SKIP(ON)  
  + TYPE(TEXT)      INTENS(LOW)  SKIP(ON)  
  _ TYPE(INPUT)     INTENS(HIGH) CAPS(ON)  
  ! TYPE(OUTPUT)    INTENS(HIGH) SKIP(ON)  
  @ TYPE(OUTPUT)    INTENS(LOW)  SKIP(ON)  
)BODY EXPAND(||)  
%|-| AAMSTR Table Selection +|-|  
%COMMAND ==>_ZCMD  
  
% /-      B = Browse, E,U = Change, I = Insert (new)      ==>_ZAMT+  
% /  
%V +ID   +Tbl Name+      Description  
)MODEL  
_Z+!Z    !AATBLNM      !AADESC  
)INIT  
  .ZVARS = '(ACTION AATBLID) '  
  .HELP  = NOHELP  
)END  
-----
```


More table services...

- **TBDISPL**
 - Displays a table according to the display format specified in the)MODEL line(s).

Displaying the table...

...and the code which uses that panel:

```
/*
  Main table processing: display table, handle updates.
. ----- */
BD_DISPLAY:          /*@          */
  address ISPEXEC

  do forever
    "TBDISPL" $tn$ "PANEL("pnl.select")" /* show selection panel */
    if rc > 4 then leave          /* PF3 ?          */
    /* panel processing goes here (ZTDSELS)          */
    action = ''                  /* clear for re-display */
  end                             /* forever          */

return          /*@ BD_DISPLAY          */
```

This is just the outline. There's more code that needs to be added. TBDISPL returns the number of rows to process in variable **ZTDSELS**.

The scrollable panel:

```
----- AAMSTR Table Selection ----- Row 1 to 17 of 17
COMMAND ==>
                                                    SCROLL ==> CSR
  /-   B = Browse, E,U = Change, I = Insert (new)
  /
V ID   Tbl Name      Description
AA  AAMSTR          Master Table
CF  CONFIG          Configuration Management Elements
b  CI  CFGINC        Configuration INCLUDEs
CP  COMPARM         Compile Parameters
FE  $FETCH          Fetchable DSNames
e  JT  JOBTRK        Track submitted jobs
      (etc.)
```

Processing the table...

Whatever the value of ztdsels, that's how many rows we will process. The initial **TBDISPL** delivers the first selected row. Each subsequent "**TBDISPL \$tn\$**" delivers another row for handling, decrementing ztdsels as it does.

```
do ztdsels
  "CONTROL DISPLAY SAVE"
  select
    /* processing for each selected row... */
  end /* Select */
  "CONTROL DISPLAY RESTORE"
  if ztdsels = 1 then, /* no more rows to do */
    ztdsels = 0
  else "TBDISPL" $tn$ /* next row */
end /* ztdsels */
```

The value of 'ztdsels' is the number of rows remaining including the current row.

Processing the table...

Since 'processing' may involve other display actions, before doing any of those we should snapshot the existing image so it can be restored later:

```
"CONTROL DISPLAY SAVE"
```

```
...
```

```
"CONTROL DISPLAY RESTORE"
```

Processing the table...

Inside the 'ztdsels' loop:

```
when Wordpos(action,"B") > 0 then do
    call BDB_BROWSE          /*          -*/
end
when Wordpos(action,"E U") > 0 then do
    call BDC_CHANGE         /*          -*/
end
when Wordpos(action,"D") > 0 then do
    call BDD_DELETE        /*          -*/
end
when Wordpos(action,"I") > 0 then do
    call BDI_INSERT        /*          -*/
end
otherwise nop
```

The data-entry panel

```
-----  
)ATTR  
  % TYPE(TEXT)      INTENS(HIGH)          SKIP(ON)  
  @ TYPE(TEXT)      INTENS(HIGH) COLOR(YELLOW) SKIP(ON)  
  + TYPE(TEXT)      INTENS(LOW)  SKIP(ON)    SKIP(ON)  
  _ TYPE(INPUT)     INTENS(HIGH) CAPS(ON)  
  ! TYPE(INPUT)     INTENS(HIGH) CAPS(OFF)  
  $ TYPE(&IO)       INTENS(HIGH) CAPS(ON)  
)BODY EXPAND(||)  
@|-| % AAMSTR Table Update @|-|  
%COMMAND ==>_ZCMD  
  
%SCROLL ==>_ZAMT+  
  
+  
+   Table ID ==>$Z @           (xx)  
+   Table Name ==>_AATBLNM @   (xxxxxxxx)  
+   Description ==>!AADESC  
+  
+   Key Fields ==>_AAKEYS  
  
+   Name Fields ==>_AANAMES  
  
+Sort Sequence ==>_AASORT  
)INIT  
  .ZVARS = '(AATBLID)'  
)END  
-----
```

More Services...

- DISPLAY
 - Displays a data-entry or selection panel.
 - Can be used to display the contents of a single table-row.
 - Can show/collect information or changes.

More Table Services...

- TBMOD
 - Overlays the row pointed to by the CRP (current row pointer).
- TBDELETE
 - Deletes the row pointed to by the CRP.
- TBADD
 - (we've seen this one before...)

The data-entry phase

INSERT and CHANGE are almost exactly alike...

```
/*
  Display a blank panel for adding a new entry.
  . ----- */
BDI_INSERT:                               /*@                               */
  address ISPEXEC

  io      = "INPUT"                        /* attribute for AATBLID          */
  parse value "" with AATBLID,
                  AATBLNM AAKEYS AANAMES AASORT AADESC

  do forever                               /* until PF3                      */
    "DISPLAY PANEL("pnl.datent")"
    if rc > 0 then leave
  end                                       /* forever                        */

  if rc = 8 then "TBADD" $tn$             /* insert changes                 */
  else do                                  /* DISPLAY failed ?              */
    ...
```

The INSERT display:

```
----- AAMSTR Table Update -----  
COMMAND ==>  
  
Table ID ==> (xx)  
Table Name ==> (xxxxxxxx)  
Description ==>  
  
Key Fields ==>  
  
Name Fields ==>  
  
Sort Sequence ==>  
  
SCROLL ==> CSR
```

The BROWSE+EDIT display:

```
----- AAMSTR Table Update -----  
COMMAND ==>  
  
Table ID ==> PM          (xx)  
Table Name ==> PGMSTR    (xxxxxxxx)  
Description ==> Program Master  
  
Key Fields ==> PMKEY  
  
Name Fields ==> PMROOT PMVER PMTYPE PMDESC PMADDDT PMADDID PMCHGDT  
PMCHGID PMACQDT PMACQID PMLOCKED PMSTYLE PMARVER  
  
Sort Sequence ==> PMKEY,C,A
```

More table services...

- TBSORT
 - Sorts the contents of the table
- TBCLOSE
 - Writes the contents of the table to DASD
- TBEND
 - Purges the table without writing.

We're done ...

Let's save the data to DASD and wrap it up:

```
/*
  Close table.  If the data has changed, TBCLOSE; otherwise TBEND.
. ----- */
BZ_CLOSE:                /*@                */
  address ISPEXEC

  if sw.0table_changed then do
    "TBSORT " $tn$ "FIELDS(AATBLID,C,A)"
    "LIBDEF ISPTABL DATASET ID("isptabl") STACK"
    "TBCLOSE" $tn$                /* write to ISPTABL          */
    "LIBDEF ISPTABL"
  end
  else,
    "TBEND" $tn$                /* purge                */
return                /*@ BZ_CLOSE          */
```

ISPTABL is the **output**-side of table processing. TBCLOSE always involves ISPTABL(‡).

Left as an exercise...

The full-version of this code (~500 lines) can be found on my REXX Utilities website:

<http://web.tampabay.rr.com/mvsrex/REXX/>

That version includes the panel-text as a comment at the back of the code, and uses an internal subroutine, DEIMBED, to extract those panels and load them to a temporary ISPPLIB. It is, therefore, almost completely self-contained and will run virtually as-is (after the missing pieces of the REXXSKEL base are added back).