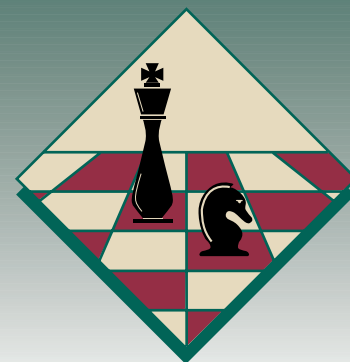


Building an ISPF-Based Corporate Management Information System: Part II



In part I of this series (December 1995) we examined managers' information needs, which are timely data they understand and can manipulate; using currently printed reports held on disk or tape as free sources of information; and TSO as a platform for a corporate management information system. This month's article reveals how this system has been implemented at SaskPower and provides steps to implement it at your company.

The first part of this series described a corporate management information system running under ISPF. This article describes implementation procedures.

The first step in implementing this system is to imagine yourself as a manager trying to use the corporation's management information system. What actions must you perform? Imagine yourself as the programmer. How can you assist the manager in performing these actions? See Figure 1.

Additional facilities that managers need include the ability to:

- download the report;
- select a section for printing; and
- print.

To use these facilities, the manager enters in the COMMAND field above the displayed report:

- DOWNLOAD;
- LINES from to;
- HARDCOPY; and then presses Enter.

The commands are implemented through CLISTs (and TSO command processors). To permit them to be run by typing them above the displayed report, they are invoked through an ISPF command table entry. For example:

```
DOWNLOAD 8 SELECT CMD(DOWNLOAD)
          DOWNLOAD THE REPORT
```

For downloading: (1) the report(s) are copied into a file named DOWNLOAD; and (2) a blank screen is displayed. It is up to the manager to run RECEIVE.

ORGANIZING INFORMATION

What's been proposed so far eliminates distribution costs and time, making it possible to improve the currency of information through more frequent reporting, but it doesn't do anything to improve the utility of the information by reorganizing it. What's a good organization for information? Obviously, it's the organization the manager wants at the moment; a subjective thing that will change with time. The only solution that's going to keep a manager happy is extraction software that can organize (sort), summarize (add), join (merge), print, and save information in a form suitable for download to a PC. The trick is to have a tool that's easy to use and flexible. Our solution is shown in Figure 2.

Figure 2 shows an extract panel on which a manager specifies the information he wants extracted from a report. In this example, the report is the employee roster and the manager is extracting the CLASS, LASTNAME, and FRSTNAME of employees whose CLASS is MGMT or UNION1, and then sorting the extracted employees by their first names. It is not important that the system you design contain a duplicate of this panel, but it is important that it address the concerns addressed by the features of the panel. See Figure 3.

How can the extract panel know the names of the fields in the report? The fields in the extract panel (and their location in the report) are specified in a dictionary created by the DBMS administrator. There is an entry in the dictionary for each report.

Figure 1: Actions to be Performed

Action	Implementation
The manager presses one or two keys to logon to ISPF.	On a terminal, the logon command is recorded so that the manager can play it back. On a PC, a keyboard key is remapped to play the logon command. In addition, other PC keys are remapped to do what they say. For example, PageUp is remapped to scroll up (send F7), PageDown is remapped to scroll down (F8), End sends F15, etc.
A list of all corporate reports is displayed.	Your first thought is probably to create a panel with the report names embedded in it—don't do it. A single panel isn't big enough to hold the names of all your company's reports, which quickly leads to a panel for financial reports, a panel for inventory reports, etc. What happens when there are too many financial reports to fit on a panel? Don't ask, and don't get into this trap. At SaskPower, we display an ISPF table of reports.
From the list of all corporate reports, the manager selects a report. A list of back issues from which to choose is displayed.	The result of a LISTCAT is used to initialize a table which is then displayed. The most recent issue is displayed first.
The manager selects several issues to BROWSE at once.	The selected issues are copied into a temporary file. The temporary file is BROWSEd.
Freezes column headings.	The manager first scrolls the line he wants to freeze to the top of the screen. ZLINES, the record number of the line at the top of the screen, is VPUTted by a modified BROWSE panel and VGETted by a program that then reads that record from the file being BROWSEd. The program VPUTs the record of column headings into a variable. The variable is displayed above the data area on a modified BROWSE panel.

Figure 3: Concerns to be Addressed

Concern	Our Solution
Manager may forget the names of the fields in the report.	Don't depend on managers to remember and correctly type field names. Display a scrollable list of field names.
Manager may forget commands.	Remind manager of commands by explaining them on the panel.
Manager may not understand command syntax.	Simplify, i.e. to apply a command to a field, type it as a single character (e.g. > for sort ascending) before the field's name.
Manager may need to alter the order of columns in the result.	The order of columns in the result is the same as the order of the rows in the extract panel, which can be altered with the Edit-like M (move) B (before) and A (After) commands.
Need to select records.	Test and Values columns.
Need to save extract table.	SAVE and COPY commands may be entered in the COMMAND field.
Managers forget the name under which they saved an extract.	To save an extract, the <i>Title</i> field must be completed. If only COPY is entered in the COMMAND field, the manager is shown a selection list of all saved extracts and their associated titles.
Managers need to exchange copies of extract panels (tables) with one another.	Command SHARETBL issued by a user copies his table into a public area. Command GETTBL issued by another user retrieves the copy.
Managers need to share extract panels. Sharing is not the same as sending a copy. Sending copies creates a maintenance problem if an error is discovered in a panel that has been copied to many accounts.	In addition to being able to save and copy friend's extracts, managers can save and use panels in <i>joint accounts</i> . A single copy of an extract in a joint account is used by many managers throughout the company each day.

Figure 2: A Non-Procedural Solution, the Extract Panel

----- Extract Panel -----

COMMAND ==>
 Title ==>
 Totals Only? ==> N { Y or N }
 +-- > sort ascending C (opy)
 < sort descending M (ove)
 + total A (fter)
 - control break B (efore)
 * extract R (epeat)
 / next line D (elete)
 \ brk+grp+page n insert
 ~ brk+grp
 Field
 C POSITION
 * C CLASS
 * C LASTNAME
 > C FRSTNAME
 * C FRSTNAME

ROW 1 TO 11 OF 1
SCROLL: CS

Edit? ==> Y { Y N or X for no
 = = > < > = < = CONTAINS IN -IN
 ! before a Test tests magnitude
 +-- values separated by commas
 values may be enclosed in
 quotes. ? = wild card
 %begins a comment

Test Values
 = **MGMT, UNION1**

↑
Extract CLASS, LASTNAME,
and FRSTNAME ; then sort
by FRSTNAME

↑ ↑
Select Employees with CLASS =
MGMT or UNION1

This extract panel provides the manager with a fourth-generation (non-procedural) way of extracting information from a report. Managers find it intuitive and error resistant. For flexibility, a procedural language is needed, such as the languages you use to manipulate the system (e.g., assembler, C, etc.), but are designed to manipulate data.

PROCEDURAL DATA MANIPULATION LANGUAGE (PDML)

The corporate management information system uses a procedural data manipulation language (PDML) to extract data. Procedural

language statements are generated from the extract panel, concatenated with the dictionary entry, and then given to an interpreter that performs the extract. Users can:

1. Manipulate the extracted data with their own PDML statements entered in the COMMAND field (shown in Figure 4, the result is shown in Figure 5), or
2. Define new fields by injecting their own PDML declarations before the statements generated from the extract panel.

Figure 4: Procedural Language Statement in COMMAND Field Inserts a Linefeed When FRSTNAME Changes

```
BROWSE - Total Lines 2694----- LINE 00000000 COL 001 080
COMMAND ==> LINEFEED FRSTNAME          SCROLL ==> CSR
95/07/02                                07:09
1992 MAY 24          MGIS.PQNRPT.DEMOR001.G0001V00
CLASS LASTNAME      FRSTNAME
***** TOP OF DATA *****
UNION1 DUCKSBERG     AARON
UNION1 GRANT         AARON
UNION1 NEUMANN       AARON
UNION1 RING          ABE
UNION1 RANKIN        ABE
UNION1 GLENNIE       ADDIE
UNION1 CLINTON       ADELE
MGMT WASHINGTON      ADRIAN
UNION1 JEFFERSON     AIME
MGMT FUCHS           AL
UNION1 HAMILTON      AL
UNION1 PAINE         AL
UNION1 TRUMAN        AL
UNION1 EISENHOWER    AL
UNION1 ROOSEVELT     AL
UNION1 TAFT          AL
UNION1 COOLIDGE      AL
UNION1 WILSON        AL
```

the dictionary, a manager can define new fields on the extract panel, i.e. to concatenate his own PDML declarations with those from the dictionary:

1. He types **COMMANDS** in the **COMMAND** field of the extract panel; Edit is started and displays a blank screen.
2. He types the PDML declarations defining the new fields on the blank screen displayed by Edit. See Figure 6.
3. He presses the End key. The extract panel with the new fields is displayed. See Figure 7.

Figure 6 EQUates a new field, INIT2, as extending from the start of INIT plus 3 (S?INIT+3, the fourth character of INIT) to the start of INIT plus 4 (the fifth character of INIT). The effect of Figure 6 on the extract panel is Figure 7. Figure 8 shows what qualities are necessary in a PDML.

Figure 5: Result

```
BROWSE - Total Lines 2694----- LINE 00000000 COL 001 080
COMMAND ==>                                SCROLL ==> CSR
95/07/02                                07:09
1992 MAY 24          MGIS.PQNRPT.DEMOR001.G0001V00
CLASS LASTNAME      FRSTNAME
***** TOP OF DATA *****
UNION1 DUCKSBERG     AARON
UNION1 GRANT         AARON
UNION1 NEUMANN       AARON

UNION1 RING          ABE
UNION1 RANKIN        ABE

UNION1 GLENNIE       ADDIE

UNION1 CLINTON       ADELE

MGMT WASHINGTON      ADRIAN

UNION1 JEFFERSON     AIME

MGMT FUCHS           AL
UNION1 HAMILTON      AL
UNION1 PAINE         AL
UNION1 TRUMAN        AL
UNION1 EISENHOWER    AL
UNION1 ROOSEVELT     AL
UNION1 TAFT          AL
UNION1 COOLIDGE      AL
UNION1 WILSON        AL
```

THE INTERPRETER

If you had to implement the data manipulation language described previously, how would you? Our first attempt used Edit to manipulate data; our procedural data manipulation language was a collection of Edit macros. On large amounts of data, it was too slow. We tried an experiment to reveal why.

The problem is, the machine you're using is too slow. "You should be using *our* 3090 instead of *your* 4341," was the opinion of a programmer who worked for a service bureau. I put the programs (Edit macros) and the data on a tape and shipped them off to his 3090. It was no faster. Finally, we figured out why.

The Edit macros weren't slow on our machine because of inadequate CPU power (if they were, the 3090 would have been faster); they were slow because of *thrashing*. Paging is an inefficient way of reading a large data file; a sequential file with a large block size and at least two buffers is more efficient. I don't have to tell you why, but paging is exactly what Edit ordered because it holds the entire file in virtual memory. We abandoned Edit, and chose the toy called a SLINKY as a model for our procedural language interpreter.

To understand how our procedural language works, visualize a SLINKY that has been bent into an inverted U. When the PDML starts, two work files are allocated and the data is loaded into the first work file. The two work files correspond to the two ends of the SLINKY. The work file with the data corresponds to the end with the most coils; the other work file is the end that has only a few coils. To process commands, the data is copied from the full work file (the end of the SLINKY with the most

Figure 6: Defining New Field INIT2 as the Fourth and Fifth Characters of INIT

```
EDIT ---- SYS95171.T095539.RA000.TCSMIS.R0000116 --
COMMAND ==>
          SPEDIT Commands
***** TOP OF DATA **
000001 EQU INIT2 S?INIT+3:S?INIT+4
***** BOTTOM OF DATA
```

THE RESULT

By using PDML statements on the extracted data, a manager can reformat, summarize, and reorganize the extracted data. Additionally, by concatenating his own PDML declarations with those from

Figure 7: ReDisplayed Extract Panel. Note the addition of INIT2.

```

----- Extract Panel ----- ROW 1 TO 11 OF 15
COMMAND ==> SCROLL: CSR
Title ==>
Totals Only? ==> N { Y or N } Edit? ==> Y { Y N or X for no, }
+-- > sort ascending C (opy) EQU INIT1 S?INIT:S?INIT THEN
    < sort descending M (ove) EQU INIT2 S?INIT+3:S?INIT+3
    + total A (fter)
    - control break B (efore)
    * extract R (epeat)
    / next line D (elete)
    \ brk+grp+page n insert
    - brk+grp
Field Test Values
C POSITION
C CLASS
C LASTNAME
C FRSTNAME
C INIT
C INIT2

```

It can be considered automatic because by choosing reports as the database, decisions about which raw and summarized data to put in and when to add data have already been made by management. Decisions about whether to store data on disk or tape are made dynamically by HSM.

It can be considered inexpensive because it uses existing resources, of which the most important, most expensive, most difficult to acquire is the understanding by our managers of the meaning of the fields—disks and tapes can be bought and instructions can be found in manuals. However, an understanding of the meaning of fields, whose values are too big or too small, and what to do about them, is only acquired through experience, frequently at cost to the company.

WHAT SHOULD YOU DO NOW?

The beginning of this article noted that this system was developed in stages so that useful results were available immediately. What's the first stage?

1. Save the data. Right now at your company reports are printed, delivered, and discarded. The cost of retaining print files on HSM tape is low, and the payoff of having the data ready in machine-readable form for an audit or historical review is high. Approach your boss and suggest saving the print files for reports on HSM tape. This is the thin edge of the wedge; once the data is there, it makes sense to be able to BROWSE it, etc.

2. Identify the department controlling the data everyone wants; the department that keeps the books (usually the finance department). Try to get them to support you, but be gentle, they're used to saying "no" to user requests. Don't misunderstand them, they haven't been saying no because they enjoy it. They didn't have the tools to say yes. Your job is to convince them that your tools will permit them to say yes. They'll be skeptical.

3. Identify the department that needs the system right now, usually because of an emergency. Offer to help them, then use their departmental muscle to get the data. It's a start.

The amazing thing about the system described in this article is not that it's so brilliant; but that it's so obvious, even technically trivial. But, its impact on your corporation isn't trivial. It's significant, possibly decisive, in the marketplace. It's an indication of the lack of communications within your corporation that anyone could have gotten this system from you for the asking, but no one ever asked. Perhaps you

Figure 8: Necessary Qualities in a PDML

Quality	Implementation
Capable of extracting from reports	Implement the follow commands: IF - to select a record. COPY - to copy a field. OVERLAY - to overlay heading information onto detail records. DROP - to delete a record.
Handle lots of data	See <i>The Interpreter</i> on the next page.
Interactive	Runs under TSO/ISPF
Permit statements to be combined into programs	<i>Edit-macro</i> type of language
Undo	See <i>The Interpreter</i> .
Permit batch as well as interactive operation so that: - Extractions can be scheduled to occur regularly. (Some managers don't want to use the computer; they just want their printed extraction delivered to them monthly.) - Long extractions can be run at night.	Run TSO/ISPF in batch. Ensure that batch diagnostics are adequate, i.e., at least give the: - reason for failure - statement that failed - record number - record contents - computed values
Never gives up on bad data	The result of an operation on <i>bad data</i> (alpha where numeric is expected, etc.) is a 'd'. The manager can FIND 'd' in the result to view the records containing bad data. If the result of an operation is too big to fit, it's replaced by ">".

coils) to the empty one. On the way, the data is modified by executing the commands. After the data has been copied, BROWSE displays the modified data in the second work file. To undo the last command entered, display the first work file.

HOW TO CREATE USER DOCUMENTATION

If you're like most programmers, you may believe that it is impossible to create good user documentation. You're wrong! You can by following the rules shown in Figure 9.

Word for Windows version 6.0 was used to create our textbook. The typical page shown

in Figure 10 is two Word for Windows tables, one above the other. Each table is just one row. The row contains two columns.


Lastly, we found that a textbook wasn't enough to ensure user acceptance of the system. The breakthrough occurred when we gave a morning-long course, went through the textbook line by line, and provided a PC/terminal for each student.

IS THIS A DATA WAREHOUSE?

This is not an example of a traditional data warehouse, but rather an automatic and inexpensive one.

Figure 9: User Documentation Rules

Focus	<p>The focus should be on <i>how to</i>. If you have to put in philosophy, don't put in much and put it in a separate section from the <i>how to</i>.</p> <p>Put little used procedures, such as <i>installation</i>, at the end in appendices.</p>
Writing	<p>Write as little as possible; illustrate as much as possible through screen images. Every action described in our textbook has an adjacent screen image. The description is connected to the screen images with arrows.</p> <p>Text should never be wider than 4.5" — use two-column format if you have a lot of text.</p> <p>Make chapters at most two pages.</p> <p>To let readers find things, use headings liberally. Set them in a bold large sans-serif type.</p> <p>Put the table of contents on the inside of the front cover; if it won't fit on one page, use a two-column format.</p>
Screen Images	<p>To allow room for the explanatory text adjacent to the screen image, print the manual in landscape mode.</p> <p>Don't use bit maps for screen images; they're difficult to read and edit. Capture your screens as characters, insert them in your manual in <i>LinePrinter</i> font with a point size of 8.5.</p> <p>Emphasize the important characters in the screen image by enlarging them and formatting them in a heavy font, but don't depend on bold for emphasis.</p>
Printing	<p>Print on both sides of the page. In an open book, readers see two pages — don't waste one.</p> <p>Don't photocopy; make each page an original.</p>
Binding	<p>To permit the manual to lie flat when open, bind it with a plastic comb binder (Cerlox).</p>

should start to open up better lines of communication within your corporation by letting them know what you can do. Wouldn't providing this system be a start? 

NaSPA member Michael Swetlow has a B.A. in Mathematics from Brooklyn College, and an M.S. in Computer Science from Purdue University. He is a certified information systems auditor, and has more than 30 years of programming experience. He can be reached at (306) 566-2074, or via fax at (306) 566-2330.

©1996 Technical Enterprises, Inc. Reprinted with permission of **Technical Support** magazine. For subscription information, email mbrship@naspa.net or call 414-768-8000, Ext. 116.

Figure 10: User's Textbook: Typical Page

14 September, 1995 **Select a Report**

Nothing you can do can damage the computer. Don't worry -- if you get into trouble, phone Mike Swetlow @ (306) 566-2074.

As an example of displaying a report, we are going to display report DEMOR001.

With the *MGIS PRIMARY OPTION MENU* displayed, type **M** in the *OPTION* field, then press the right-hand Ctrl key. The *MGIS Report Selection Screen*, shown below, should be displayed [remember, when *** appear, press the right-hand Ctrl key].

MGIS PRIMARY OPTION MENU -----

OPTION ==> **M**

D Docs - Documentation
L List - List of Personalized reports
M MGIS - Management Information System: All Reports
X EXIT - Terminate MGIS using log and list defaults

Enter application to see only reports from a particular application Enter Application and Report to view a particular report

Application ==> V Report ==> V

Last Name ==> SWETLOW Printer ==> LJET2

COMMAND ==> ROW 3 FROM 9

MGIS Report Selection Screen
=====

PLACE:
A Before a member name to add it to your private list;
B Before a member name to see a report description; or
I Before a member name to see the report.

Member	S	Report	Contents
CEISR001	%-	PPX166	Standard Costs
CSISR001	%-		In service meters
CSISR002	%-		In stock meter records
CSISR003	%-		Transformer meter records
DEMOR001	%-	LINDEX	with phone numbers
DEMOR002	%-		Meter Records
FMISR006	%-	O M A	Account Detail Report

Displaying All the Issues of a Report

To display the issues of report DEMOR001, use the *Enter* key to move the cursor to the area before DEMOR001. Type **i**, then press the right-hand Ctrl key. The *MGIS Historical Report Selection* screen, shown on the next page, should be displayed.

? Trouble? Phone Mike Swetlow @ (306) 566-2074 to talk about it.