

Appendix  
**B**

**WORKSHOP**

*Get on the  
Fast Track!*



TM

**SYS-ED/  
Computer  
Education  
Techniques, Inc.**

---

## 1 Xpediter - Facilities

**Objectives:**

- Compile and run a program under Xpediter.
  - The program is a working program and does not need to be debugged.

**Description:**

Create a PDS for the following:

- source code
- load module
- Source debugging output - symbolic file
- Script file

Except for the load module, all datasets have a LRECL of 80; the load module should be a PDS, a record length of 0 and a record format of U.

**Walkthrough:**

Use the naming conventions from the home installation to:

- Compile the XPED1 program to support Xpediter features.
  - The instructor will provide the location of the program.
  - Unless instructed otherwise, use the Xpediter compile facility.
- The first time that the compile facility is run, a JOB card must be entered.

Assuming that all the steps have completed successfully:

- Select the Xpediter setup and specify the load module and source debugging output name.
- Enter the debugging facility with XPED1 and run the program.
- When the program terminates due to SYSOUT not being allocated, allocate SYSOUT.
- Run the program to a successful termination.
- Restart the program and run it again.

---

## 2 General Rules and Format

**Objectives:**

- Compile, run, and debug a program under Xpediter.
  - The program contains simple run-time errors.

**Description:**

- Compile the XPED2 program to support Xpediter features.
  - The instructor will provide the location of the program.
- Unless instructed otherwise, use the Xpediter compile facility.
  - The first time that the compile facility is run, a JOB card must be entered.

**Walkthrough:**

Assuming that all steps have completed successfully:

- Allocate all datasets for execution using the TSO ALLOCATE commands.
- Enter the debugging facility with XPED2.
- Set a breakpoint on every paragraph and run the program.

At an abnormal breakpoint, check the content of the variable that is causing the error and modify the variable.

- Single step through part of the program.
- Set a conditional breakpoint; this will stop the program when the variable FLAG has been modified.

Restart the program.

- Remove all breakpoints and trace the program.

Restart the program.

- Display an execution count on every statement.
- Correct the errors and run the program to a successful termination.

Allocate all datasets for execution using the Xpediter File Allocation Utility.

---

### 3 Breakpoints

**Objectives:**

- Compile, run, and debug a program that invokes another procedure under Xpediter.
- Debug the calling and the called program.

**Description:**

- Compile programs XPED3 and XPED4 into the same load module.
- Unless instructed otherwise, use the Xpediter compile facility.

**Walkthrough:**

- Set two breakpoints; one in each procedure.
- Run the program and debug the error.

After the program is debugged, rerun only the subprogram in XPED4.

- Debug the program as a standalone module without a calling program.

---

## 4 Variables

**Objectives:**

- Compile, run, and debug a program that invokes another procedure under Xpediter.
- Debug the calling and the called program.

**Description:**

- Compile programs XPED3 and XPED4 into the same load module.
- Unless instructed otherwise, use the Xpediter compile facility.

**Walkthrough:**

- Display the values in the first three fields in Working-Storage.
  - Reset the display.
- Display the values in all the fields in Working-Storage.
  - Reset the display.
- Keep a variable in the KEEP window.
- Run the program and view the variable.
  - Reset the keep variable.
- Go to the current line using the LOCATE command.
  - Set FLAG in Working-Storage to Y using the Xpediter MOVE statement.

---

## 5 Utility Commands

**Objectives:**

- Compile, run, and debug a program that invokes another procedure under Xpediter.
- Debug the calling and the called program.

**Description:**

- Compile and debug program XPED5.
  - Unless instructed otherwise, use the Xpediter compile facility.

**Walkthrough:**

- Within the Xpediter test environment, browse the program input dataset.
- Display the LOG screen.
- Set the Xpediter PF12 key to SHOW COUNTS.
- Display the DCB characteristics for all input and output datasets.
- Ascertain when this program was compiled.
- Ascertain when this program was linked.
- Ascertain the version of the compiler.
- Display the allocated files.

---

## 6 Test Scripts

**Objectives:**

- Record, code and execute a script.
- Code and run Xpediter Batch.

**Description:**

- Use programs XPED3 and XPED4, turn on the counts and set three breakpoints.

**Walkthrough:**

- Save the script into the Test script dataset.
- Return to test mode and run the script.
- Setup the script to run when test mode is invoked.
- Code the JCL to run Xpediter from a batch job.
- Invoke XPED3 and XPED4 from batch JCL.

---

## 7 DB2 Stored Procedures

**Objectives:**

- Compile, run, and debug a program that invokes a DB2 stored procedure under Xpediter.
- Debug the stored procedure.

**Description:**

- Compile and debug program XPEDCALL and STPROC as a stored procedure.
- Unless instructed otherwise, use the Xpediter compile facility.

**Walkthrough:**

- Run the setup to specify DB2 system names and DSNLOAD datasets.
- Specify the schema name \_\_\_\_\_.
- Specify the stored procedure name \_\_\_\_\_.
- Specify the DB2 subsystem name \_\_\_\_\_.
- Specify the maximum number of tests for the stored procedure that will be debugged \_\_\_\_\_.
- Convert and submit the JCL to run the program invoking the stored procedure.
- Use the RUN command to submit the job.
- How many arguments are passed to the stored procedure\_\_\_\_\_.
- Display the values of each argument passed to the stored procedure.