TDOC Utility
for the Creation of
Source Code Walkthrough Documents

Version 2.1
November 11, 2003

prepared by:

R. G. Zbrog
LEN CONSULTING LLC

1. TDOC UTILITY

This document describes the Technical Source Code Walkthrough Documentation (TDOC) utility, a function of ZZUtils, a set of utilities for Natural developers and administrators.

1.1. TECHNICAL CODE WALKTHROUGH

A technical code walkthrough, sometimes called a peer code review, among other things, is an effective tool in the areas of quality assurance and education. The developer is exposed to alternate methods and processes as the technical lead and database administrator suggest and discuss improvements to the code. The technical lead is assured of an acceptable level of quality and the database administrator is assured of an acceptable level of database performance. The result is better performance of the developer, his programs, and the entire application.

The typical scenario finds a developer inviting the technical lead, a database administrator, and one or more peers to a meeting to review a set of source modules prior to production implementation. Modified code may be indicated after the fact with annotations or a highlighting pen, or within the code itself with comments.

Here are the most common excuses offered for not enforcing code walkthroughs as a shop standard.

- Volumes of data.
  Technical leads and database administrators are unwilling to spend time wading through large Natural and COBOL listings searching for a few simple source changes. Database administrators need to see quickly what database accesses were added or changed.

- Deleted code.
  Deleted code cannot be reviewed, but to leave the code in-line, converted to comments, can render illegible an otherwise well-structured module.

- Distribution/logistics.
  More and more, development teams are dispersed over several geographic locations, making distribution of documentation more difficult. Even when the distribution concerns are resolved, it is difficult to ensure that all eyes are on the same page. When the developer asks everyone to turn to page 12 of module XXAB***DC***99, invariably someone is looking at page 12 of module XXAB***CD***99.

- Manual effort.
  The amount of effort required by the developer is significant to create useful documentation for a technical walkthrough.

These issues are resolved with a technical solution, the TDOC utility (see Section 1.2, *Objective*). Other issues are addressed by offerings from LEN Consulting LLC. Please visit www.ZZUtils.com/walkthru.html for more information.

1.2. OBJECTIVE

The objective of the TDOC utility is to mitigate the problems associated with performing code reviews.

ZZTDOC is an on-line Natural screen which spawns a mainframe job to create a dataset containing a set of source and comparison listings. The dataset is downloaded to a PC where it is converted to a word processing document. A macro creates a title page, table of contents, header, and footer. All source code additions, deletions, and changes are highlighted with shaded text. The result is saved as a Word document for softcopy distribution.

ZZTDOC can create a one-time report for a single module, or the user may create a Natural source member which defines a *set* of up to ten modules. This *set* module may be modified or re-submitted as necessary. The set may include a mixture of Natural, COBOL, JCL, catalogued procedure, parameter data source modules, and sample reports. Each module may be located in a different environment or library/PDS.

For updated source modules, TDOC will create a comparison listing of the old and new versions using IBM's SuperC utility. For new modules a source listing is created. Natural modules may be listed as standard source, formatted, or structured (see Section 2.17, *Listing Type*). For COBOL a compile listing may be requested for the source or comparison listing, and Adabas, CICS, and DB2 pre-compilers may be invoked.

The problems described in the previous section are addressed as follows:

- Volumes of data.
  For an updated module, a *delta* comparison is created. At a glance the reviewer can see whether the module's changes require a cursory review or in-depth investigation. For a Natural module, an Adabas command listing is generated. This brief report includes the line number, file number, and file name of each Adabas command in the module.

- Deleted code.
  By comparing the old version to the new, all changes are identified without negative impact to the source module.

- Distribution/logistics.
  The resulting report is a word processing document, easily distributed via e-mail or posted to a Web site. Although comprised of several mainframe reports, each with its own page numbering, the entire report has a contiguous page number in the footer. This is the page number referenced in the table of contents.

- Manual effort.
  TDOC can reduce the manual effort of documentation by 95%. Developers have been observed to need an average of five hours to create manually a ten-module report with the same pagination and formatting as the TDOC utility. TDOC typically reduces that effort to fifteen minutes.

TDOC eliminates the need for multiple procedures, concurrently processing Natural and COBOL modules, and sample report output. And it enforces a standard report format.

1.3. TDOC COMPONENTS

Several TDOC components are related to the installation and customization of the utility. The ZZUtils administrator should refer to Section 4.3, *Customization Procedure*, for more information. The following components are related to the day-to-day operation of the utility by the end-user, the developer.

1.3.1. ZZTDOC Specification Screen

The ZZTDOC specification screen is the heart of the TDOC utility. It is used to generate batch jobs which create the computer listings that are formatted into walkthrough documents.

ZZTDOC is capable of handling Natural, COBOL, JCL, catalogued procedures, and other data which is needed during a code review. The administrator has the ability to customize ZZTDOC. By setting a flag, all COBOL-related fields are eliminated. Another flag can limit ZZTDOC to Natural-only processing. Eliminating the COBOL and 3GL capability leaves ZZTDOC with a more streamlined look.

Most fields have a default value. The defaults are defined in module ZZTDPROF, and are set by the administrator via the ZZTDPMNT function during installation and customization. Refer to Section 1.3.2, *ZZTDPMNT Profile Maintenance Screen*, below, and Sections 4.3.3, *ZZTDPROF*, and 4.3.4, *ZZTDPMNT*.

Each field on the ZZTDOC screen is documented in Chapter 2, *ZZTDOC Specification Screen*.

ZZTDOC is invoked from the Natural command line or the ZZUtils menu.

The screen is divided into six areas:

- Env — identifies the environments in which the source modules are found
- Set — identifies the location of a set
- Listing — characteristics of the generated reports
- Print — characteristics of the job/spool output
- File — specification of the output dataset
- Job — characteristics of the batch job

Either a single module may be reported (OLD and/or NEW) or a *set* is specified. These two options are mutually exclusive. If a set is specified, it must be verified via PF key *SetV* prior to job submission.

If COBOL processing is allowed, copybook libraries must be specified and verified with PF key *CpyBk*.

1.3.2. ZZTDPMNT Profile Maintenance Screen

ZZTDOC has more than two dozen entry fields, plus twenty copybook entries in pop-up windows. To expedite data entry, each user may create a TDOC profile. All but two of the fields (old member and new member) may be assigned user-customized default values. By default ZZTDPMNT will store the profiles as text members in library ZZTDPROF, each identified by the user's userid.

ZZTDPMNT is invoked from the Natural command line or the ZZUtils menu. Refer to Section 4.3.4, *ZZTDPMNT*, for more information.

1.3.3. TDOC Sets

More often than not, a code walkthrough will involve multiple source modules. The user creates a TDOC set to specify which modules to include in the report, where they are located, and in what sequence they are to be reported. The set is stored as a Natural text member in a library of the user's choosing.

A set contains the same information as found on the ZZTDOC screen in the OLD, NEW, and LISTING areas (see Section 1.3.1, *ZZTDOC Specification Screen*, above). If a set does not contain a complete specification of those three areas, the missing values are taken from the ZZTDOC specification screen on which the set was selected. In any case, values from the PRINT, FILE, and JOB areas always originate from the ZZTDOC specification screen.

A complete set of specifications is advantageous because it is self-documenting. A subset specification is advantageous if the set is to be re-used. For example, consider a group of modules which will move first from development to acceptance, and later from acceptance to the production environment. A set may be created which includes the development and acceptance environment settings. For the subsequent execution, the set would be modified for the new environment specifications. If the environment settings are not included in the set specification, then the set would not require modification for the second execution; the environment specifications could be supplied on the ZZTDOC screen at execution time.

Each entry in a set is comprised of *keywords* and *values*, in the form *keyword=value*. Each keyword corresponds to a field on the ZZTDOC screen. Refer to Appendix B, *Keywords*, for a complete list.

Keyword values are separated by commas, as in *keyword1=val1,keyword2=val2*. Embedded blanks are not allowed; blanks are line delimiters and are followed by comments. A single entry may require multiple lines. End the previous line with a complete keyword value, a comma, and a hyphen, and continue the entry with the next keyword value on the next line. Entries are delimited by a non-continuation line.

There are two types of set entries. An entry which contains a member specification is a member entry. An entry which does not contain a member specification is a default entry. A default entry is used only to set the default values for all subsequent member entries. This is useful in reducing the size and increasing the legibility of contiguous member entries which have multiple keyword values in common. Keyword values in member entries also will set default values for subsequent member entries. The exceptions are the old member name and new member name keywords.

Up to ten member entries may be specified in a set. A set may contain a maximum of sixty (60) source lines, including default entries, comments, and blank lines.

Refer to Appendix C, *Set Examples*.

1.3.4. ZZTDMERG Combining Sets

In practice the ten-module maximum imposed by ZZTDOC is not a limitation; ten medium-to-large Natural or COBOL modules would create a document of unwieldy size. On the other hand there may be sets of more than ten small modules which could be combined into a document of reasonable size. ZZTDMERG was created for this situation.

A series of mainframe datasets, each containing up to ten modules from a ZZTDOC run, are combined into a single dataset prior to download to a PC. Refer to Appendix D, *ZZTDMERG JCL*.

1.4. CREATING A WALKTHROUGH DOCUMENT

To create a formatted document, follow this procedure:

- If multiple modules are to be reported, create a *set* using the Natural editor. See Section 1.3.3, *TDOC Sets*, above, and Appendix C, *Set Examples*. To report a single module, go directly to the next step.
- From an on-line Natural session, invoke the ZZTDOC utility from the command line.
  - Complete the specification screen.
  - To speed the data entry process, create a customized set of default values with the ZZTDPMNT utility, described in Section 1.3.2, *ZZTDPMNT Profile Maintenance Screen*.
  - In the FILE section, set the DISP and NAME parameters to direct the output to a sequential file.
  - Press PF key *Sub* to submit a batch job to create the sequential file.
- Download the sequential file as a text file (.TXT) to your PC. (A TSO/emulator/IND$FILE transfer is recommended over FTP.)
- Open the text file in MS Word.
  - Maximize the MS Word application window.
  - Click *Tools* on the menu bar.
  - From *Macro* select *Macros*.
  - Highlight ZZ*TDOC*.
  - Click *Run*.
- Save the document in MS Word format (.doc).

2. ZZTDOC SPECIFICATION SCREEN

This section is essentially a reproduction of the information available on-line on the ZZTDOC specification screen via the *Help* PF key. Following an overview of the screen are descriptions of the selection fields, in alphabetical order. Following the field name, in parentheses, is the associated keyword, where applicable. Refer to Appendix B, *Keywords*, for a cross-reference of keywords to field names and Section 1.3.3, *TDOC Sets*, for a description of their use.

2.1. ZZTDOC OVERVIEW

This utility will submit a batch job to create a source code walkthrough document.

A single source module may be reported, or a set of up to ten modules may be combined into a single document. Each module may be one of the following source types:

- Natural
- COBOL
- JCL
- Cataloged Procedure
- Sample report output
- Operations documentation
- Utility parameters
- Other

Sample report output must have a record length of 133 bytes, with RECFM=FBA. For all other source types, the record length must be 80 bytes, with RECFM=FB.

For each module, one or more reports may be generated:

- A standard listing of a new module
  - Natural modules may be formatted or structured
  - COBOL modules may be compiled, with Adabas, CICS and DB2 pre-compilers
- A SuperC “delta” listing, showing only the module’s changes
- A SuperC “long” or “context” listing, with each modification highlighted
- A SuperC summary report
- COBOL cross-reference listing
- List of the Adabas commands within the module (Natural only)

The document may be spooled to a SYSOUT hold queue or directly to a printer. The recommended procedure is to create the document as a PDS member, which is downloaded to the PC as a text file. Enhancements, such as a title page, a table of contents, page numbering, and highlighting of changes, are applied automatically by a supplied MS Word macro: ZZTDOC.

Most fields on the selection screen will be populated with default values. Each user may create and maintain a profile of default values with the Profile Maintenance function ZZTDPMNT (see Appendix E.4).

2.2. ADABAS COMMAND LISTING (ADACMD)

Specify whether to generate an Adabas command listing for a Natural module.

2.3. COMPARISON SUMMARY (SUMMARY)

Specify whether to generate a SuperC summary report.

2.4. COMPARISON TYPE (COMPARE)

The comparison listing may be of two types, context or full. The context type displays up to ninety-nine (99) lines of source above and below any changed line (see Section 2.5, *Context Window*). The full listing shows the entire source module with embedded changes.

This specification applies only to comparisons. If a new module is selected, this specification is ignored, and a standard listing is created (see Section 2.17, *Listing Type*).

This field is required.

2.5. CONTEXT WINDOW (CONTEXT)

Specify the number of lines to be displayed in a context comparison (see *Comparison Type*, above), before and after each changed line. Specify a positive number between 1 and 99.

2.6. COPIES

Specify the number of copies of any spooled reports.

This field is required.

2.7. COPYBOOKS (CB-OLDNN, CB-NEWNN)

Up to ten PDSs may be concatenated at compile time into a Copybook library. A different list of PDSs is specified for Old and New members. At least one PDS must be specified for each.

Press ENTER on the TDOC specification screen to populate the first entry of each list with the name of the PDS in which the old or new member is found. Press the CpyBk PF key to further customize the lists.

2.8. CROSS-REFERENCE (XREF)

The inclusion of a cross-reference listing in the document is optional.

This field is required for COBOL modules, but disallowed for non-COBOL.

2.9. DELTA LISTING (DELTA)

Specify whether to generate a SuperC “delta” listing, showing only the module’s changes.

2.10. FILE DISPOSITION

If routing the document to a sequential file, specify the file's disposition:

N	New	- create a new file or PDS member
O	Old	- overwrite an existing file or PDS member
A	Append	- append to an existing file
X	None	- route the document to the printer

Output from the Append option may be submitted to the ZZTDMERG utility.

This field is required.

2.11. FILE NAME

Specify a data set name.

If a member name is specified (in parentheses), the file is presumed to be a PDS and a new member is created or an existing member is overwritten depending upon the specification of the File Disp value. If no dataset is requested (ie File Disp value is "X"), the file name is ignored.

A default value is created by concatenating the userid and the current Julian date. If a named module is currently in Natural's work area, that name is used as a PDS member name.

Press PF Key MEMBR to append the module name to the dataset name as a PDS member. The module name is derived from the first non-blank entry found in the Set Member, New Member, or Old Member fields.

2.12. HOLD OUTPUT

Specify whether to place printer output into the Hold Queue.

This field is required.

2.13. IGNORE NATURAL LINE REFERENCE CHANGES (ILRC)

Specify whether changes to Natural line number references should be considered source changes.

2.14. INFO

Specify a sub-account number for the JOB card.

This field is required.

2.15. JOB ACCOUNT

Specify an account number for the JOB card.

This field is required.

2.16. JOB NAME SUFFIX

Specify a suffix to be appended to your userid to create a JOB name. The value must be alphanumeric.

Specify a different value for each request to allow multiple jobs to run in parallel.

This field is required.

2.17. LISTING TYPE (LIST)

The standard and comparison listings may be of the following types.

For Natural:

Compile	formatted listing via ZZLIST
List	source listing via ZZPUNCH
Struct	formatted listing via Natural's Struct command

For COBOL:

Compile	formatted, with optional pre-compilation
Source	source listing

2.18. MEMBER TYPE (MEM-TYPE)

Both the old and new source members must be of the same type:

N	Natural
C	Cobol
J	JCL
P	catalogued Procedure
R	Report sample – PDS member
Q	report sample – seQuential file
U	Utility parameters
D	operations Documentation
O	Other

2.19. NEW ENVIRONMENT (ENV-NEW)

Specify the environment in which the new source module is found. This field is required for Natural modules.

For example:

D	development
P	production

2.20. NEW LIBRARY (LIB-NEW)

Specify the Natural library in which the new source module is found. This field is required for Natural modules.

The default value is the current Natural library.

2.21. NEW MEMBER (MEM-NEW)

If the modules to be compared have the same name in different libraries, specify an "old" member; the "new" member name will default to the "old" name.

If the modules to be compared have different names in the same or different libraries, specify both the "old" and "new" names.

If the module to be reported is new, specify a "new" member and leave the "old" member blank. A standard listing will be produced in place of a comparison listing.

For Member Type “Q” the New Member field is used as the report name in the document header.

If multiple modules are to be combined into a single document, use Natural’s program editor to create a source member containing the names of up to ten modules. Specify the name of this source module in the "set" member field (see Section 2.31, *Set Member*, below).

2.22. NEW PDS (PDS-NEW)

Specify the PDS in which the new source module is found. This field is required for non-Natural modules.

For Member Type “Q” this field is used as the name of the sequential file which contains the report sample.

2.23. OLD ENVIRONMENT (ENV-OLD)

Specify the environment in which the old source module is found. This field is required for Natural modules.

For example:

D	development
P	production

2.24. OLD LIBRARY (LIB-OLD)

Specify the Natural library in which the old source module is found. This field is required for Natural modules.

2.25. OLD MEMBER (MEM-OLD)

If the modules to be compared have the same name in different libraries, specify an "old" member; the "new" member name will default to the "old" name.

If the modules to be compared have different names in the same or different libraries, specify both the "old" and "new" names.

If the module to be reported is new, specify a "new" member and leave the "old" member blank. A standard listing will be produced in place of a comparison listing.

If multiple modules are to be combined into a single document, use Natural's program editor to create a source member containing the names of up to ten modules. Specify the name of this source module in the "set" member field (see Section 2.31, *Set Member*, below).

The default value for the "old" member is the current Natural work area name.

2.26. OLD PDS (PDS-OLD)

Specify the PDS in which the old source module is found. This field is required for non-Natural modules.

For Member Type "Q" this field is used as the name of the sequential file which contains the report sample.

2.27. OUTPUT CLASS

Specify the printer output class.

This field is required.

2.28. PRE-COMPILE (COMPILE)

Specify which, if any, pre-compilers are required and the sequence in which to invoke them.

A	Adabas SQL
C	CICS
D	DB2

This field applies only to COBOL modules. It may be blank or contain from one to three characters. The order of the specification is the order in which the pre-compilers are invoked.

2.29. PRINTER ID

Specify a JES printer.

This field is required.

2.30. SET LIBRARY

Specify the Natural library containing a Natural module which defines a set of modules to be reported as a package. This field is required if a "set" member name is specified.

There is no default value.

2.31. SET MEMBER

Up to ten modules may be combined into a single document via a *set* construct. A “set” is a text module containing up to ten entries, with each entry defining an individual source module. The text module is a *separate* Natural source member, saved as a program, subprogram, subroutine, helproutine, copycode, or text module. Each set entry is composed of a combination of keywords. The keywords are documented in Appendix B, *Keywords*.

Indeed you may specify more than ten entries in a set, but only the first ten will be processed. A warning message will indicate that the remaining entries have been ignored.

Here is a simple example of a set entry.

```
MEM-OLD=OLDIE,MEM-NEW=GOODY
```

For more information, refer to Section 1.3.3, *TDOC Sets*. For examples, refer to Appendix C, *Set Examples*.

2.32. SYSOUT

Specify whether to include JES listings with the printer output. These JCL and execution-time listings will not be included in the output routed to the sequential output file.

This field is required.

3. TIPS

3.1. INFORMAL VS FORMAL WALKTHROUGHS

For an informal walkthrough, route the output directly to a printer. Set the number of copies appropriately.

For a formal walkthrough, route the output to a sequential file, download it, format it with the supplied macro, and distribute the documentation as softcopy.

3.2. SET MEMBER

On the ZZTDOC control screen if a set member is specified, the Old Member and New Member specification fields are ignored.

3.3. PRINTER VS FILE

Printer and file output are mutually exclusive. The exception is JES output, which is created when requested regardless of the File Disp specification. If a dataset is requested (ie File Disp value is not "X"), the Printer ID, Copies, Class, and Hold fields apply only to the SYSOUT listings.

3.4. BATCH JOB ABENDS

The most common cause of batch job abends is a conflicting File Disp parameter. Invariably the File Disp is set to New when the dataset already exists. For debugging purposes set the SYSOUT parameter to Y and resubmit the job. The cause of the abend may be determined from the system output.

3.5. NO COMPARISON

If the environment, library, and member values are the same for both ‘old’ and ‘new’ (or the PDS names and member names are the same in the case of 3GL modules), then no comparison listings are generated. A standard listing is created rather than compare a module to itself.

3.6. FTP

FTP is faster than an emulator-invoked download. FTP is invoked from a DOS window; an emulator-invoked download requires an active TSO session.

Despite FTP’s advantages, its use is not recommended at this time. Data transfers via IND$FILE have provided more consistent and accurate results.

3.7. MAXIMIZED WINDOW

The ZZTDOC macro appears to run faster if the Word window is maximized.

4. INSTALLATION

4.1. PREREQUISITES

4.1.1. Word Processor

The ZZTDOC macro was created using MS Word 97. It has been tested under Word 97, Word 2000, Word 2002, and Word XP.

4.1.2. Mainframe Operating System

TDOC's on-line routines are written in Natural, so they will run on any platform which supports Natural 3. TDOC submits batch jobs to a mainframe operating system via NATRJE. Currently only MVS JCL is supported.

4.1.3. Comparison Utility

IBM's SuperC utility is used to generate the comparison listings.

4.1.4. Copy Utility

Batch datasets are copied using IBM's DFSort utility. The parameters used are compatible with SyncSort parameters.

4.1.5. Natural User Exits

TDOC uses the following Natural userexits, which must be available in the steplib chain:

- USR0080
- USR1005
- USR1029
- USR1031
- USR1043
- USR1055
- USR1057
- USR1067
- USR2013
- USR2019
- USR2026
- USR2031

4.2. MAINFRAME INSTALLATION

ZZTDOC is a component of ZZUtils. For convenience the basic ZZUtils installation instructions are included here. If ZZUtils is already installed, ignore Section 4.2.

4.2.1. Unzip the Distribution File

Unzip ZZUTIL2.ZIP to a directory of your choice. *ZZUTIL2.ULD* contains the mainframe Natural components. *ZZTDOC Macro.dot* is a template which contains the ZZTDOC macro and the ZZUtils logo.

4.2.2. Load the Natural Components

- Allocate a mainframe flat file with RECFM=FB, LRECL=250, BLKSIZE=250.
- FTP file ZZUTIL2.ULD to the mainframe. The resulting dataset will have the characteristics of RECFM=U, LRECL=0, BLKSIZE=250. The FTP commands are
    - binary
    - put zzutil2.uld *mf_fb_file* fixrecfm 250
- Use standard Natural JCL to verify the flat file's contents with the SCAN option of the NATLOAD utility.
    - point CMWKF01 to the mainframe input file
    - logon sysunld
      natload scan
      fin
- If using Natural Security, create and secure a ZZUTILS library.
- Load the Natural components into the ZZUTILS library.
    - point CMWKF01 to the mainframe input file
    - logon sysunld
      natload
      fin

4.3. MAINFRAME CUSTOMIZATION

4.3.1. ZZCUSTOM

This Natural subprogram is used to customize the ZZUtils environment. Make adjustments as necessary, and STOW in the Natural library specified in the #ZZUTILIB variable (typically the ZZUTILS library).

Refer to Appendix A, *Customization*, for an excerpt of ZZCUSTOM.

- #ZZUTILIB library containing ZZUtils modules
- #STEPLIB whether #ZZUTILIB is within the steplib chain
- #ZZTDLIB library containing custom profiles
- #ZZTDPMNT whether a user may override #ZZTDLIB
- #PF-xxxxx set PF keys to match the shop standard
- #ENVIRON Natural environment definitions
- #3GL whether TDOC will process non-Natural modules
- #COBOL whether #3GL includes COBOL
- #COMPILER COBOL compiler executable program name

4.3.2. ZZTDJMVS

This Natural text module contains MVS JCL segments used to create batch jobs which generate the TDOC reports. Make adjustments as necessary, and SAVE in the library specified in ZZCUSTOM.#ZZUTILIB.

- Natural catalogued procedures are referenced. Ensure that the DDNAME sequences and symbolics match your PROCs.
- Verify all DCB parameters to adhere to shop standards.
- Verify that use of temporary files (&&) adheres to shop standards.
- Ensure that DFSORT/SyncSort parameters are compatible with your sort utility.

4.3.3. ZZTDPROF

This Natural text module contains default values for the ZZTDOC specification screen. Make adjustments via the ZZTDPMNT program – DO NOT EDIT THIS MODULE DIRECTLY.

These default values will be presented to ZZTDOC users who have not created a custom TDOC profile. They also are used as a template when creating a custom profile.

Once adjustments are complete, set ZZCUSTOM.#ZZTDSEC to TRUE to secure the default profile from further changes via ZZTDPMNT. Copy ZZTDPROF from the library specified in ZZCUSTOM.#ZZUTILIB to the library specified in ZZCUSTOM.#ZZTDLIB.

4.3.4. ZZTDPMNT

This Natural screen is used to make adjustments to the default TDOC profile, ZZTDPROF. It also will create individual TDOC user profiles, using the ZZTDPROF profile as a template.

Custom profiles will be stored in the library specified in ZZCUSTOM.#ZZTDLIB, identified by the name found in *USER. If the user is allowed to override the library name, the profile will be named ZZTDPROF.

4.4. WINDOWS CUSTOMIZATION

4.4.1. Load the Word Macro

The macro is attached to the document *ZZTDOC Macro.dot*. This document must be available to each PC from which ZZTDOC will be executed.

- Click *Tools* → *Options* → *File Locations* and note the location values for *User templates* and *Workgroup Templates*. If none is specified, click *Modify* to enter a directory name for *User templates*.
    - If multiple PCs will execute ZZTDOC, copy *ZZTDOC Macro.dot* to the directory specified for *Workgroup Templates*.
    - If ZZTDOC is to be executed on a single PC, copy *ZZTDOC Macro.dot* to the directory specified for *User templates*.

To make the macro available, follow this procedure for each PC:

- Open ZZ*TDOC Macro.dot* in MS Word.
- When prompted, click *Enable Macros*.
- Click *Tools* → *Macro* → *Macros*.
- Ensure that ZZ*TDOC Macro.dot (template)* is selected in the *Macro in:* dropdown.
- Ensure that ZZ*TDOC* is the selected Macro name. The name may vary. (eg *Project.ZZTDOC*).
- Click *Organizer*.
- Ensure that ZZ*TDOC Macro.dot (Template)* is selected in the lower left-hand dropdown.
- Ensure that *Normal.dot (global template)* is selected in the lower right-hand dropdown.
- Highlight the ZZ*TDOC* macro in the upper left-hand selection box.
- Click *COPY*.
- Ensure that the ZZ*TDOC* macro is in the upper right-hand selection box.
- Click *CLOSE* on the bottom right.
- **Close and then open MS Word to make the new macro available.**
- When prompted to save changes to Normal.dot, respond *Yes*.

4.4.2. Create a Macro Button on the Word Toolbar

Add a button to the toolbar and customize the button name:

- Click *Tools* → *Customize ...*
- Select the *Commands* tab.
- Select the *Macros* entry in the *Categories* windows.
- Click and drag an entry from the *Commands* window to a tool bar of your choice.
- Right-click the new button.
- Click *Name*.
- Edit the highlighted button text.
- Close the *Customize* window.

4.4.3. Logo

By default the title page of the TDOC report displays the ZZUtils logo. To change this, replace the logo in *ZZTDOC Macro.dot*. The location of this file is defined in Section 4.4.1, *Load the Word Macro*.

The *ZZTDOC Macro.dot* document is used as a template for TDOC title pages, so use care to replace the default logo with one of the same height.

APPENDIX A – CUSTOMIZATION

The following is an excerpt from the Natural source module ZZCUSTOM, supplied with ZZUtils. Modify these parameters and STOW the module in the Natural library specified in #ZZUTILIB.

```
0010 ************************************************************************
0020 * Subpgm:   ZZCUSTOM                    (c) 2003 LEN Consulting LLC
0030 * System:   ZZUTILs
0040 * Function: Client Customization
0050 * Author:   Ralph G. Zbrog

0170   /*
0180   /* Client definition:
0190   /*
0200   2 #client (a30)  init <'LEN Consulting LLC'>   /* Your organization
0210   2 #zzutilib (a8) init <'ZZUTILS'>    /* ZZUtils Natural library
0220   2 #steplib (a1)  init <'Y'>          /* Y  Individual utilities will
0230                                        /*    be found/executed in the
0240                                        /*    Steplib chain
0250                                        /* N  Not in Steplib; must logon
0260                                        /*    to #zzutilib
0270                                        /*' ' Determined based on current
0280                                        /*    online Steplib; presumes
0290                                        /*    same Steplib in batch.
0300   /*
0310   /*
0320   /* Library specification restrictions:
0330   /*
0340                         /* Restricting a utility enforces Natural
0350                         /* Security; you must be logged onto the
0360                         /* appropriate application library to access
0370                         /* the specified module, and the utility must
0380                         /* be executable from within the Steplib chain.
0390                         /*
0400                         /* Unrestricted - access any member from any
0410                         /* application library:
0420                         /*       logon ZZUTILS
0430                         /*       zzxxx applib,member
0440                         /*
0450                         /* Restricted - you must have access to the
0460                         /* application library to access its modules:
0470                         /*       logon applib
0480                         /*       zzxxx member
0490                         /*

0510   2 #zzadacmd (l)  INIT <false>
0520   2 #zzlist (l)    INIT <false>
0530   2 #zzpunch (l)   INIT <false>
0540   2 #zztdpmnt (l)  init <false>
```

```
0560   /*
0570   /* PF Key definitions:          /* NB Case sensitivity
0580   /*
0590   2 #pf-help  (a4) init <'PF1'>
0600   2 #pf-auth  (a4) init <'PF2'>   /* Author window
0610   2 #pf-exit  (a4) init <'PF3'>
0620   2 #pf-rfrsh (a4) init <'PF4'>   /* Screen refresh
0630   2 #pf-pswd  (a4) init <'PF6'>   /* Specify Natural Security password
0640   2 #pf-up    (a4) init <'PF7'>   /* Page backward
0650   2 #pf-down  (a4) init <'PF8'>   /* Page forward
0660   2 #pf-cpybk (a4) init <'PF9'>   /* TDOC - COBOL copybook libraries
0670   2 #pf-jcl   (a4) init <'PF11'>  /* View JCL
0680   2 #pf-sub   (a4) init <'PF12'>  /* Submit JCL
0690   2 #pf-term  (a4) init <'PF24'>  /* Terminate Natural
0700   /*


2230   /*
2240   /* ZZTDOC parameters:
2250   /*
2260   2 #ZZTDSEC (L)   INIT <TRUE>    /* Secure default profile
2270   2 #ZZTDLIB (A8)  INIT <' '>     /* Profile repository/library
2280                                   /*   eg ZZTDPROF
2290   2 #ENVIRON (A21/40)   INIT      /* Must match sequence of #FUSER
2300     /*  Cls    JCL Proc     NATLIB..     Abbr
2310        <'E' - 'NATDEVL ' - 'APPDEV  ' - 'DEVL'        /* D
2320        ,'E' - 'NATTEST ' - 'APPTST  ' - 'TEST'        /* T
2330        ,'E' - 'NATUSER ' - 'APPUSER ' - 'USER'        /* U
2340        ,'E' - 'NATPROD ' - 'APPPROD ' - 'PROD'        /* P
2380        >
2390   2 #ESYMB (A50/40)     INIT          /* Symbolics for Natural PROC
2400        <'TIME=10,SYSOUT=$TRACE$'      /* D
2410        ,'TIME=10,SYSOUT=$TRACE$'      /* T
2420        ,'TIME=10,SYSOUT=$TRACE$'      /* U
2430        ,'TIME=50,SYSOUT=$TRACE$'      /* P
2470        >
2480   2 #EPARM (A50/40)     INIT          /* Parameters for Natural PROC
2490        <'MADIO=0,MAXCL=0,MT=0'        /* D
2500        ,'MADIO=0,MAXCL=0,MT=0'        /* T
2510        ,'MADIO=0,MAXCL=0,MT=0'        /* U
2520        ,'MADIO=0,MAXCL=0,MT=0'        /* P
2560        >
```

```
2580   2 #3GL (L)       INIT <TRUE>          /* Include other module types
2590   2 #COBOL (L)     INIT <FALSE>         /* Include COBOL (#3GL subset)
2600   2 #COBCMP (A8)   INIT <'IGYCRCTL'>    /* COBOL compiler
2610   /*
2620   /*
2630   /* JCL/Script parameters:
2640   /*
2650   2 #JCLMEM (A8)   INIT <'ZZTDJMVS'>              /* JCL source member
2660   2 #COMPARE (A8)  INIT <'ISRSUPC'>               /* SuperC
2670   2 #SORT (A8)     INIT <'SORT'>                  /* DFSort/SyncSort
2680   2 #DUMMY (A8)    INIT <'IEFBR14'>               /* Dummy executable
2690   2 #HOLDN (A3)    INIT <'NO'>
2700   2 #HOLDY (A3)    INIT <'YES'>
2710   2 #TRACE (A1)    INIT <'2'>                     /* Hold print class
2720   2 #DCLS (A1)     INIT <'N'>                     /* Dummy print class
2730   2 #DSPN (A50)    INIT <'(NEW,CATLG,DELETE)'>    /* New disposition
2740   2 #DSPM (A50)    INIT <'(MOD,CATLG,DELETE)'>    /* Mod disposition
2750   2 #DSPO (A50)    INIT <'(OLD,KEEP,KEEP)'>       /* Old disposition
2760   2 #DSPS (A50)    INIT <'SHR'>                   /* SHR disposition
2770   2 #ALLOC (A10)   INIT <'CYL'>                   /* Space units
2780   2 #SPACES (A25)  INIT <'(15,15),RLSE'>          /* Sequential space
2790   2 #SPACEP (A25)  INIT <'15,15,15'>              /* PDS space
2800   2 #JCLT (A25)    INIT <'//'>                    /* JCL termination
```

APPENDIX B – KEYWORDS

Refer to Section 2, *ZZTDOC Specification Screen*, for complete descriptions of the fields associated with these keywords.

Keyword	Field Name
ADACMD	Adabas Command Listing
CB-NEWnn	New Copybooks
CB-OLDnn	Old Copybooks
COMPARE	Comparison Type
COMPILE	Pre-compile
CONTEXT	Context Window
DELTA	Delta Listing
ENV-NEW	New Environment
ENV-OLD	Old Environment
ILRC	Ignore Natural Line Reference Changes
LIB-NEW	New Library
LIB-OLD	Old Library
LIST	Listing Type
MEM-NEW	New Member
MEM-OLD	Old Member
MEM-TYPE	Member Type
PDS-NEW	New PDS
PDS-OLD	Old PDS
SUMMARY	Comparison Summary
XREF	Cross-reference

APPENDIX C – SET EXAMPLES

C.1. NATURAL SET

```
0010 * ZZTDOCEX Sample TDOC Parameters
0020
0030 MEM-TYPE=N,-                          * Default parameters
0040 ENV-OLD=d,LIB-OLD=DLSTMAIN,-
0050 ENV-NEW=D,LIB-NEW=WKSTRS12,-
0060 COMPARE=C,CONTEXT=10,-
0070 DELTA=N,SUMMARY=N,ADACMD=Y,ILRC=Y,LIST=L
0080
0090 MEM-OLD=ABNDADD                       * 1 Presumes new name = old
0100 MEM-OLD=abndadd,mem-new=rgzdadd       * 2
0110 mem-new=rgz,list=c                    * 3 New only; no comparison
0120 mem-new=rgz,list=s                    * 4 Struct
```

C.2. COBOL SET

```
0010 * COBOL members
0020 MEM-TYPE=C,COMPARE=C,CONTEXT=7,-      Set defaults
0030 LIST=C,XREF=N,COMPILE=AC,-            Adabas/CICS pre-compilers
0040 PDS-OLD=PRODLIB.LIBRIAN.COBOL,-
0050 PDS-NEW=DEVLLIB.SOURCE.COBOL,-
0060 CB-OLD01=PRODLIB.LIBRIAN.COBOL,-
0070 CB-OLD02=PRODLIB.COPYBOOK.PYAPP,-
0080 CB-OLD03=PRODLIB.COPYBOOK.APAPP,-
0090 CB-NEW01=DEVLLIB.SOURCE.COBOL,-
0100 CB-NEW02=PRODLIB.LIBRIAN.COBOL,-
0110 CB-NEW03=PRODLIB.COPYBOOK.PYAPP,-
0120 CB-NEW04=PRODLIB.COPYBOOK.APAPP,-
0130
0140 MEM-OLD=PYP00100,MEM-NEW=PYP00101     Replacement
0150 MEM-OLD=APN00210,XREF=Y               New = old name
0160 MEM-NEW=APP12345,XREF=N               New module
```

C.3. BATCH SET

```
0010 * Batch sequence
0020 COMPARE=F,CONTEXT=10,-                Set defaults
0030 ENV-OLD=P,ENV-NEW=U,-
0040 LIB-OLD=PRODAPP,LIB-NEW=ACPTAPP,-
0050 LIST=C,XREF=N,COMPILE=DAC,-           DB2/Adabas/CICS
0060 PDS-OLD=PRODLIB.LIBRIAN.COBOL,-
0070 PDS-NEW=DEVLLIB.SOURCE.COBOL,-
0080 CB-OLD01=PRODLIB.LIBRIAN.COBOL,-
0090 CB-OLD02=PRODLIB.COPYBOOK.PYAPP,-
0100 CB-OLD03=PRODLIB.COPYBOOK.APAPP,-
0110 CB-NEW01=DEVLLIB.SOURCE.COBOL,-
0120 CB-NEW02=PRODLIB.LIBRIAN.COBOL,-
0130 CB-NEW03=PRODLIB.COPYBOOK.PYAPP,-
0140 CB-NEW04=PRODLIB.COPYBOOK.APAPP,-
0150
0160 MEM-OLD=PYP00100,MEM-TYPE=N           Mainline
0170 MEM-OLD=PYN00210                      Subprogram
0180 MEM-NEW=PYM12345,LIST=S               New selection screen
0190 MEM-OLD=PY098765,MEM-TYPE=C,LIST=C    COBOL subprogram
0200
0210 PDS-NEW=DEVLPYR.SOURCE.COBOL,-        New default PDS
0220 CB-NEW03=PAYROLL.LIBRIAN.COBOL,-      New default copybook
0230 MEM-TYPE=C,MEM-OLD=PYX001223
0240
0250 PDS-NEW=SYSOPS.PROD.JOBS,-
0260 MEM-TYPE=J,MEM-NEW=PPYR070            JCL
0270
0280 PDS-OLD=SYSOPS.PROD.DOC,-             Documentation
0290 PDS-NEW=DEVLLIB.SOURCE.DOC,-
0300 MEM-TYPE=D,MEM-OLD=PPYR070
0310
0320 PDS-OLD=STRGZ.ONE.TWO.THREE.FOUR.FIVE.SIX.SEVE.OLDIE,-
0330 PDS-new=STRGZ.ONE.TWO.THREE.FOUR.FIVE.SIX.SEV.GOODIE,-
0340 MEM-TYPE=Q,compare=f,-                * New defaults
0350 MEM-NEW=GOODIE                        * 5
0360
0370 MEM-TYPE=R,DELTA=Y,SUMMARY=Y,-
0380 PDS-OLD=STRGZ.list,MEM-OLD=OLDIE,-
0390 PDS-NEW=STRGZ.list,MEM-new=goodie     * 6
```

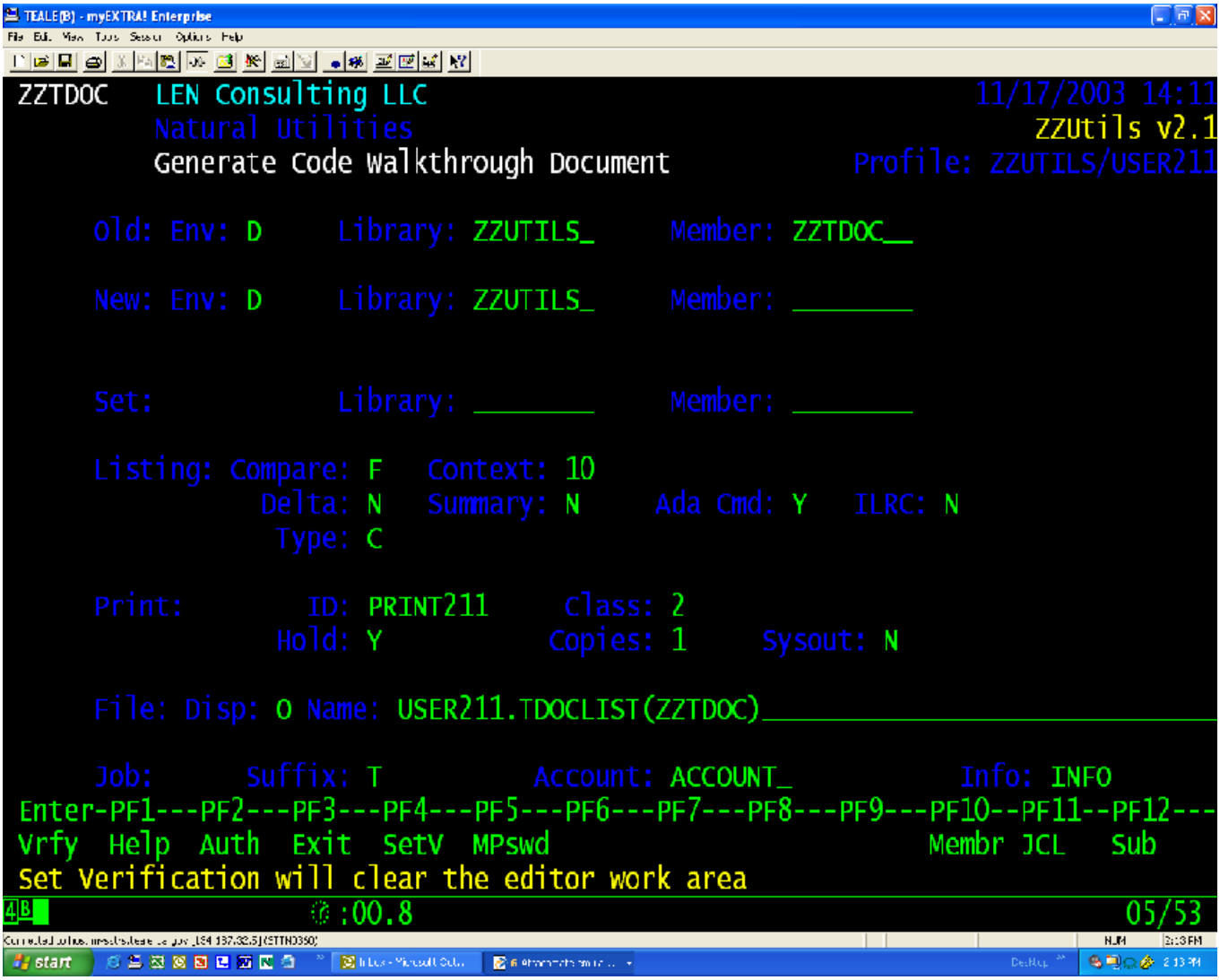
APPENDIX D – ZZTDMERG

D.1. SAMPLE JCL

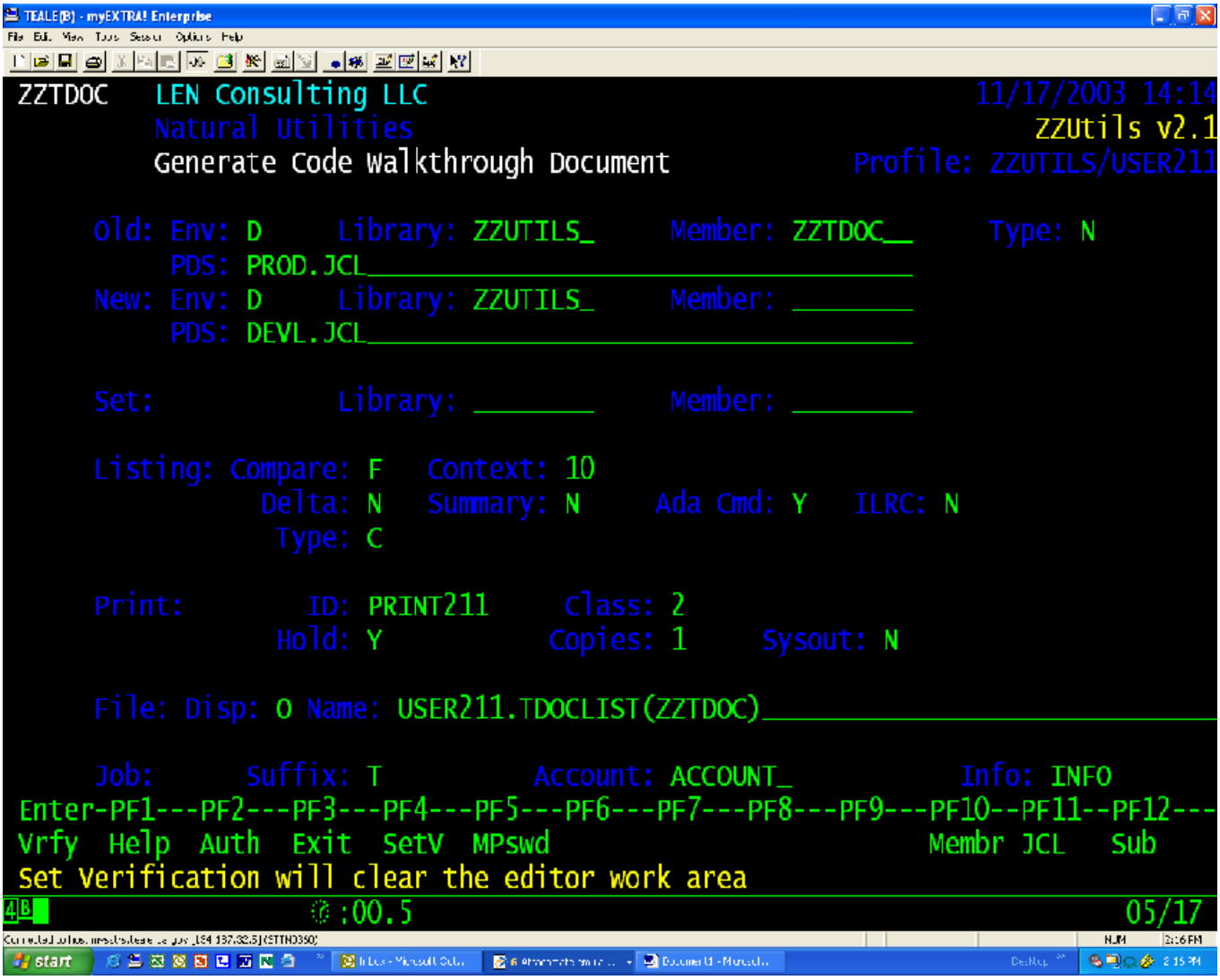
```
//MRG      JOB (account,info,subacct),'ZZTDMERG',
//          MSGCLASS=c,NOTIFY=&SYSUID
//* ***************************************************
//* MERGE TDOC REPORTS
//*
//* CMWKF02: RECFM=FBA,LRECL=133
//* ***************************************************
//SYSTR    EXEC NATBAT,PRM='AUTO=ON,MAXCL=0,MT=0,IM=D'
//CMPRINT   DD SYSOUT=*
//CMWKF01   DD DSN=user.TDOCLIST(set1),DISP=SHR
//          DD DSN=user.TDOCLIST(set2),DISP=SHR
//CMWKF02   DD DSN=user.TDOCLIST(ALL),DISP=SHR
//CMSYNIN   DD *
ZZTDMERG
FIN
/*
//
```

APPENDIX E – SCREEN IMAGES

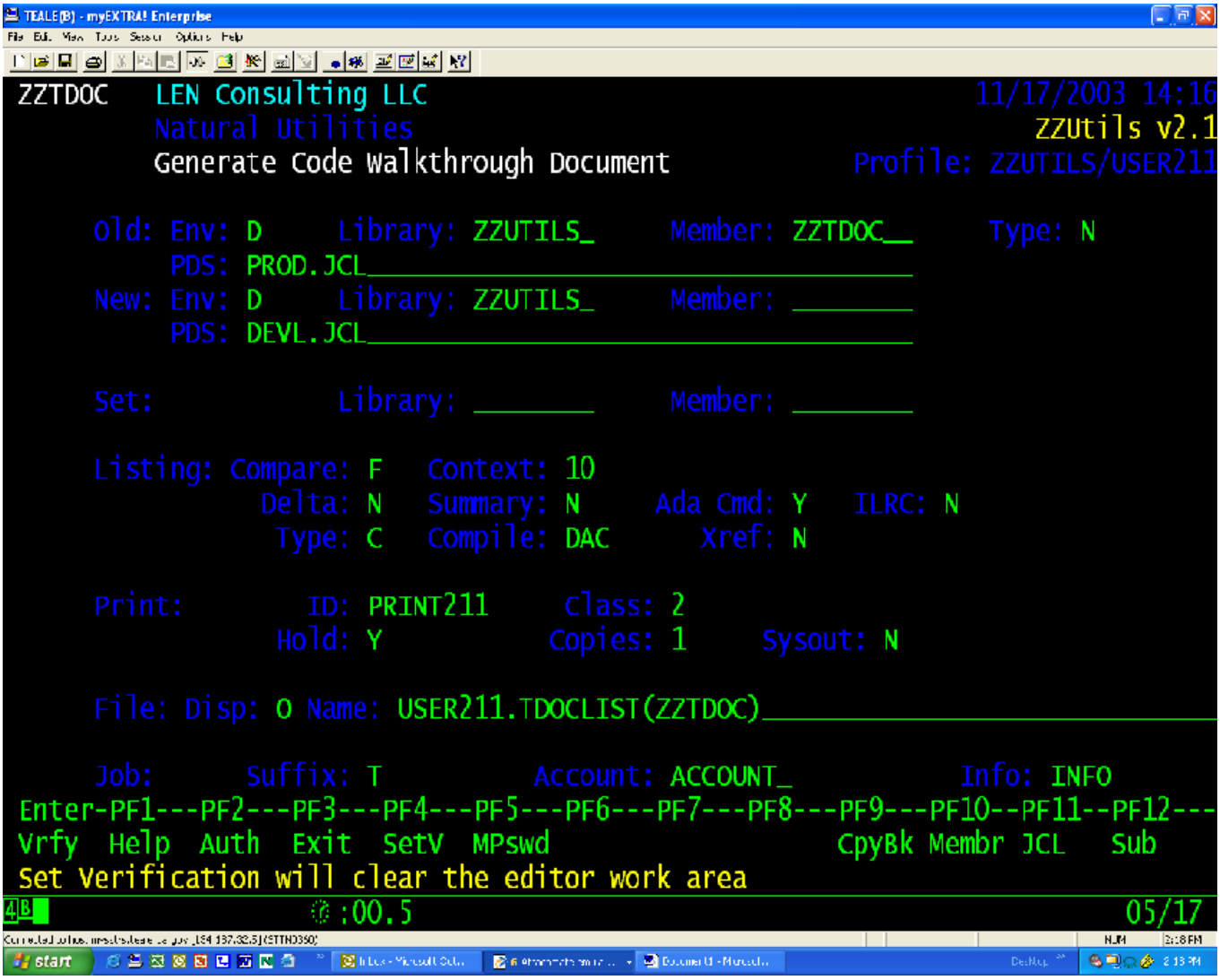
E.1. ZZTDOC – NATURAL-ONLY



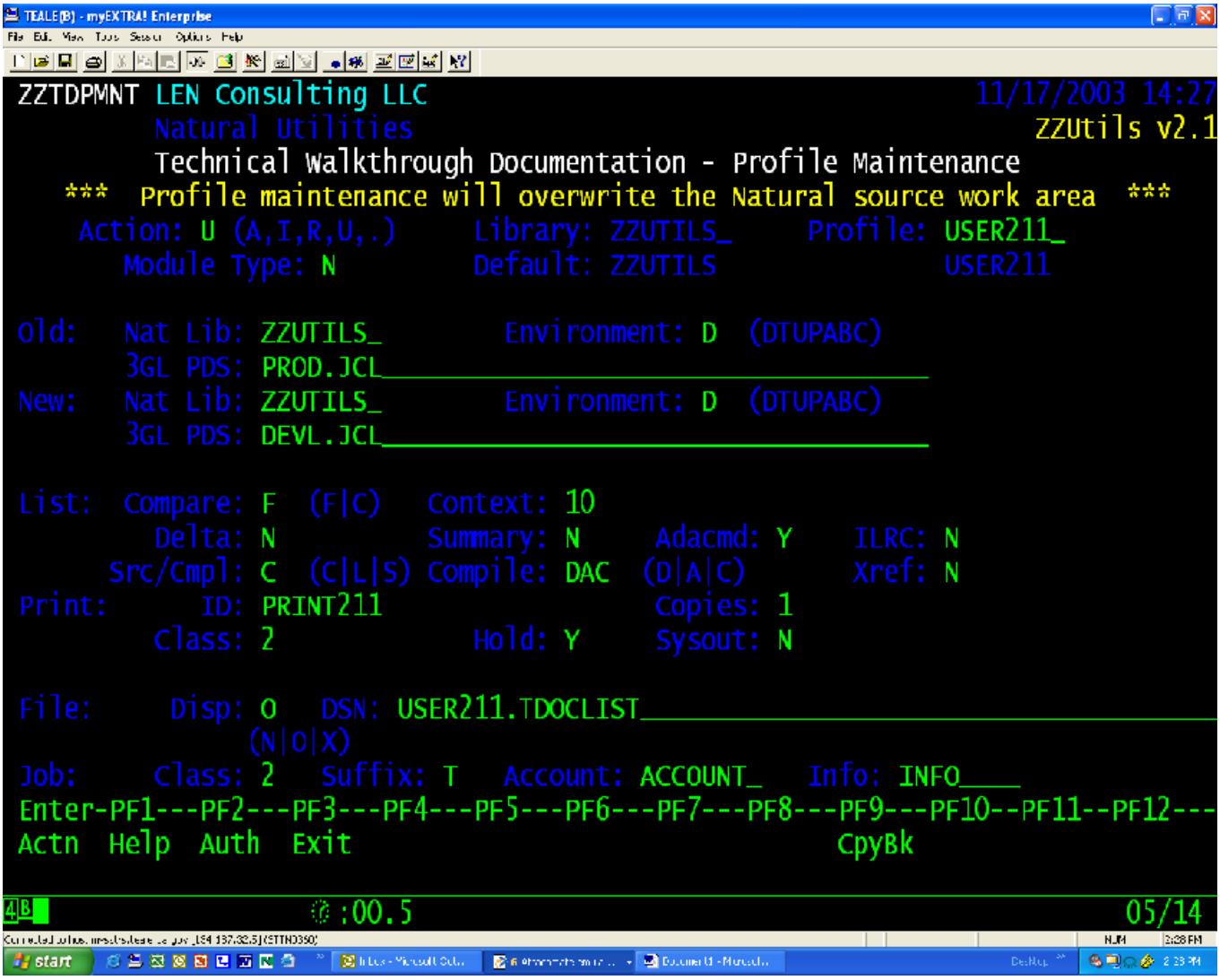
E.2. ZZTDOC – WITH 3GL SUPPORT



E.3. ZZTDOC – WITH COBOL SUPPORT



E.4. ZZTDPMNT


TEALE(B) - myEXTRA! Enterprise
ZZTDPMNT LEN Consulting LLC 11/17/2003 14:27
Natural Utilities ZZUtils v2.1
Technical Walkthrough Documentation - Profile Maintenance
*** Profile maintenance will overwrite the Natural source work area ***
Action: U (A,I,R,U,.) Library: ZZUTILS_ Profile: USER211_
Module Type: N Default: ZZUTILS USER211
Old: Nat Lib: ZZUTILS_ Environment: D (DTUPABC)
3GL PDS: PROD.JCL
New: Nat Lib: ZZUTILS_ Environment: D (DTUPABC)
3GL PDS: DEVL.JCL
List: Compare: F (F|C) Context: 10
Delta: N Summary: N Adacmd: Y ILRC: N
Src/Cmpl: C (C|L|S) Compile: DAC (D|A|C) Xref: N
Print: ID: PRINT211 Copies: 1
Class: 2 Hold: Y Sysout: N
File: Disp: O DSN: USER211.TDOCLIST
(N|O|X)
Job: Class: 2 Suffix: T Account: ACCOUNT_ Info: INFO____
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Actn Help Auth Exit CpyBk
05/14
start