CA Test Data Manager

Mainframe Integration

Best Practices Guide

Author : Walter Guerrero

Version: 1.5

Filename: CA-TDM-MF-Best-Practices-GuideV1.5.docx

Date: 1/19/2018

Table of Contents

Overview

CA Test Data Management now provides the ability of processing Cobol/PL1 copybooks that reside in the MF and use the definitions found in these copybook files to create the necessary AFL (Advanced Record Layout) files, which will be consumed by TDM as G-T Excel files for the creation of the corresponding relational table representation for the desired Mainframe objects.

This document will provide with the necessary information in how the mainframe integration works and best practices with a small sample for you to get familiar with this feature.

TDM MF Requirements

The following requirements need to be completed prior to the use of the TDM MF facilities:

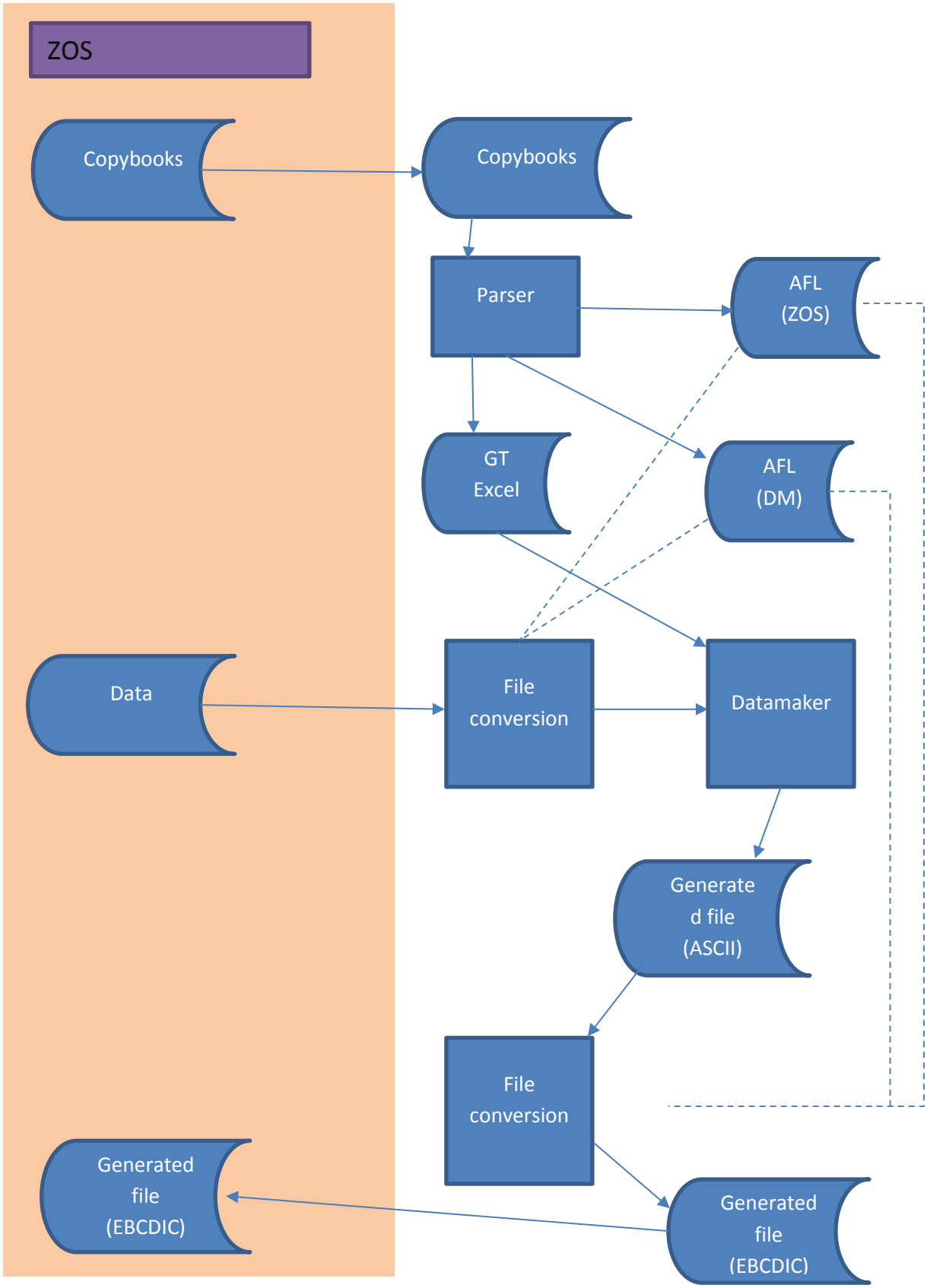
- Mainframe user access
  - TSO access
  - FileMaster Plus access or any type file manager, such as FileAid
  - ISPF editor access
  - DB2 instance access
  - Access to VSAM files
  - IMS instance access
- Mainframe user needs to have ftp capabilities
- QWS3270 or equivalent installed
- TDM 4.x
- DB2 Connect v10.x or better (or DB2 standard with DB2 Connect feature)
-

TDM Mainframe Architecture

The TDM mainframe integration is composed of several components:

- CA TDM
- CA TDM File Definition Manager
- CA TDM File Conversion Utilities

TDM MF Processing



The process flow is as follows:

1. Transfer copybooks to your local Windows system
2. Parse copybooks to produce
    a. GT Excel (registered in DM for generation)
    b. Generated AFL (describes the format of the data published by TDM)
    c. ZOS AFL (describes the format of the data in EBCDIC format)
3. Transfer example data file to Windows (optional, but very useful to give a starting point for data masking or synthetic data generation)
4. Convert example data file to Windows format (uses the two AFLs to define the source and target)
5. Load data into TDM
6. Set up data generation rules
7. Publish to file (FD)
8. Convert published file to ZOS format (uses the two AFLs to define the source and target)
9. Transfer file to ZOS

Copybook Definition

The "copybook" that is part of Cobol and PL/1 contains a series of record layouts that describe the individual fields that compromise each record in a data file. There are very specific rules that need to be followed by COBOL programs.

The copybook layout will specify at least the name of each field, its type, size, and position in the record. The copybook layout may provide you with a detailed description of the use of each field and the values to be found in it. Sometimes this information is not provided, but is available in the data dictionary. Here is another item that you need to in mind, a copybook layout usually pertains to a single disk or tape file, as opposed to a table within a database subsystem.

The contents of a copybook layout file are as follows:

HEADER RECORD

DETAILED RECORD (REPEATED FOR EACH DETAIL RECORD)

TRAILER RECORD

Copybook Fields

The lowest data item level is a field, and this "*field*" is called an *elementary item*. Several fields can be associated to form a group. Together all the fields will form a *record*.

In the copybook layout, there is going to be a line for each field or group. You will include the field name and the "PIC" or picture clause, this clause tells you the data type or data category of the field. The most common datatypes that you will see are:

- "A" for alpha (A-Z, a-z, and space only)
- "9" for numeric (numbers 0-9, no alpha!!)

- "X" for any character (including binary)

Here are some examples of the type of field lines that you will encounter:

- 05 ZIPCODE PIC 99999.
- 05 ZIPCODE PIC 9(5).
- 05 LASTNAME PIC A(15).
- 05 LASTNAME PIC X(15).
- 05 AMOUNT PIC 999.99.
- 05 AMOUNT PIC 9(3).9(2). → Represent same value as above.
- 05 AMOUNT PIC 999V99 → This is an implied decimal.
- 05 BALANCE PIC S9(6)V99 → This is a signed field for +/- entries.

Special Characters

There special formatting characters that we need to keep in mind.

Literal – This special formatting in a field causes that character to appear in that given location, for example:

05 ZIPCODE-PLUS-9 PIC 99999-9999

The above specifies that the field will have five digits initially, a dash, and finally the last four digits. The only item that is variable in this field is the dash. That is the literal character.

Copybook Levels

The copybook layouts have levels from level 01 to level 49 (you are going to encountered usually up to about level 30 with the more complex copybook layouts). The levels are used to tell the COBOL compiler how to associate/group the fields in the record.

It is important that you pay very close attention to "level 01", since this level is reserved for the record level; and this level is the name of the record.

For levels 02 to 49, they are equal, where there is no higher significance for level 02 over level 03.

Here is an example of copybook including level 01:

```
01  MAILING-RECORD.
    05  COMPANY-NAME            PIC X(30).
    05  CONTACTS.
        10  PRESIDENT.
            15  LAST-NAME       PIC X(15).
            15  FIRST-NAME      PIC X(8).
        10  VP-MARKETING.
            15  LAST-NAME       PIC X(15).
            15  FIRST-NAME      PIC X(8).
        10  ALTERNATE-CONTACT.
            15  TITLE           PIC X(10).
            15  LAST-NAME       PIC X(15).
            15  FIRST-NAME      PIC X(8).
    05  ADDRESS                 PIC X(15).
    05  CITY                    PIC X(15).
    05  STATE                   PIC XX.
    05  ZIP                     PIC 9(5).
```

Figure: A sample copybook with a level 01

66 and 88 Levels

The COBOL language has two levels with special meaning.

Level 66 assigns an alternate name to a field or group, this level does not add a new field to a record, thus you are not likely to see this level very often.

Level 88 equates a value with a name, here is an example:

```
05      SEX     PIC X.
        88      MALE        VALUE "M".
        88      FEMALE      VALUE "F".
```

You can also specify multiple values or a range of values, for example:

```
88      ODD-NUMBERS     VALUE       1, 3, 5, 7, 9.
88      PRE-SCHOOL      VALUE       0 THROUGH 4.
```

For all levels, groups, and fields the first 6 characters must be blank on the left margin. There can be additional characters as needed.

Typical copybooks that you might encountered from a very simple copybook to a more complex layout.

Here is a simple copybook with a single level 01

```
01  CUSTOMER-RECORD.
    05  LAST-NAME              PIC X(15).
    05  FIRST-NAME             PIC X(8).
    05  STREET-ADDRESS         PIC X(20).
    05  CITY                   PIC X(17).
    05  STATE                  PIC XX.
    05  ZIP-CODE               PIC 9(5).
    05  FILLER                 PIC X(10).
```

Figure: Simple copybook layout

Now here we have a complex copybook with multiple levels, as well as level 01 fields.

```
01  MAILING-RECORD.
    05  COMPANY-NAME           PIC X(30).
    05  CONTACTS.
        10  PRESIDENT.
            15  LAST-NAME      PIC X(15).
            15  FIRST-NAME     PIC X(8).
        10  VP-MARKETING.
            15  LAST-NAME      PIC X(15).
            15  FIRST-NAME     PIC X(8).
        10  ALTERNATE-CONTACT.
            15  TITLE          PIC X(10).
            15  LAST-NAME      PIC X(15).
            15  FIRST-NAME     PIC X(8).
    05  ADDRESS                PIC X(15).
    05  CITY                   PIC X(15).
    05  STATE                  PIC XX.
    05  ZIP                    PIC 9(5).
```

Figure: More complex copybook

Copybook Field/Group Redefines

A field in the copybook layout can be used for different data by "redefining" the original field. For example:

```
05      STREET-ADDRESS                  PIC X(20).
05      PO-BOX REDEFINES STREET-ADDRESS PIC X(20).
```

In the example, you can see that both fields are using 20 bytes, and both fields occupy the same 20 bytes in the record, this is possible because the second definition redefines the first one.

Now let's say that you need to redefine a group, you will do the following:

```
05  STREET-ADDRESS.
    10  ADDRESS-LINE-1         PIC X(20).
    10  ADDRESS-LINE-2         PIC X(20).
05  RURAL-ADDRESS REDEFINES STREET-ADDRESS.
    10  RURAL-ROUTE            PIC 9(3).
    10  RR-BOX-NUMBER          PIC X(10).
    10  FILLER                 PIC X(27).
```

Figure: A Redefine of a group

Copybook Tables

One of the powerful features of COBOL is the use of “tables”, this is accomplish via the “occurs” and “occurs depending on” clauses.

OCCURS allows you to define a table, instead of having to create multiple fields that will hold monthly sales data for example. This is accomplished via the following:

```
05      MONTHLY-SALES       OCCURS 12 TIMES
                                    PIC S9(5)V99.
```

This clause can also be used for groups as well, for example:

```
05  LINE-ITEMS OCCURS 25 TIMES.
    10  QUANTITY           PIC 9999.
    10  DESCRIPTION        PIC X(30).
    10  UNIT-PRICE         PIC S9(5)V99.
```

Figure: Occurs at the group level

The following clauses are not presently supported in the TDM parser, as this support is added, we are going to an explanation and examples of how this clause is implemented.

- Occurs depending on
- Copy
- Indexed by

TDM Mainframe Support

The Test Data Management Mainframe package is composed of:

- CA TDM File Definition Manager parser
- CA TDM file conversion utilities
- CA TDM mainframe objects (PGMs and JCL procs)

The above packages can be downloaded from the CA Support site as needed, in this example, the version being downloaded is 5.4.12 or greater:

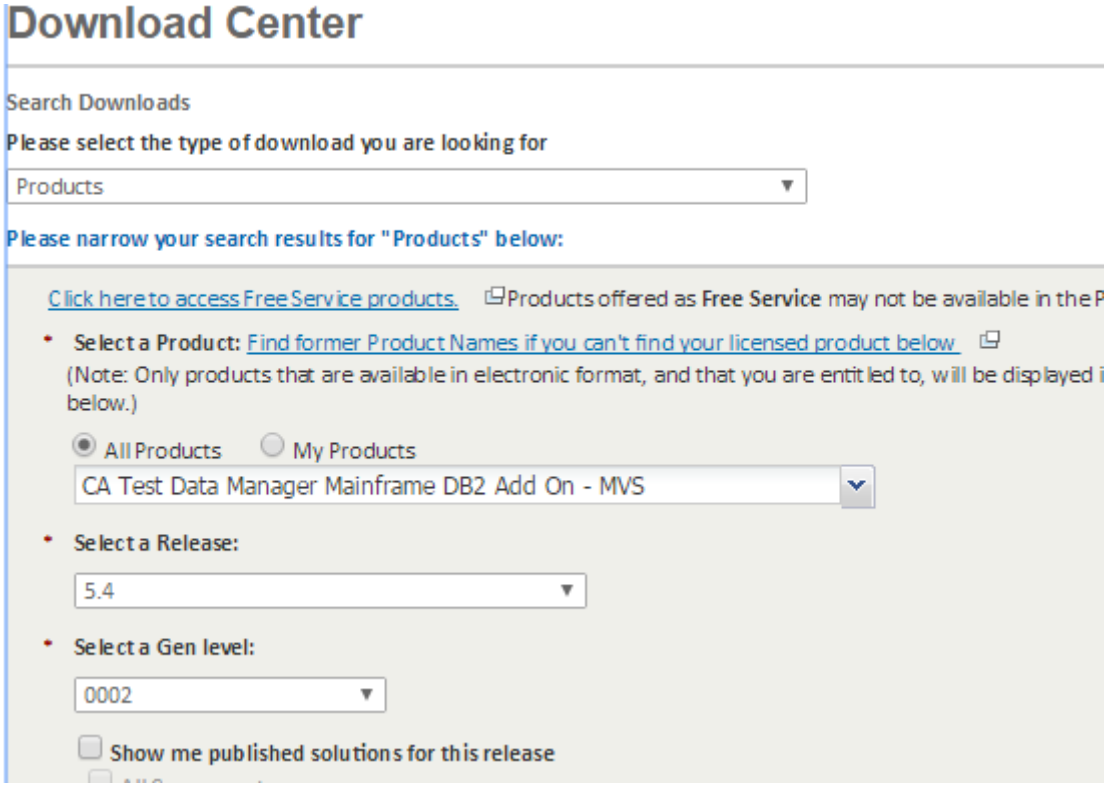


Figure: CA Support with the correct parameters

The available packages that you will be downloaded are:

- Copybook Parser utility
- File Conversion utility
- Mainframe package

Product Components			
CA Test Data Manager Copybook Parser Utility 5.4.8 GEN10143457E.zip	N/A	11/03/2015	7.9MB
CA Test Data Manager File Conversion Utility 5.4.8 GEN10143531E.zip	N/A	11/03/2015	19.85MB
CA Test Data Manager for Mainframe 5.4.8 GEN10143600E.zip	N/A	11/03/2015	36.65MB

Figure: Typical listing of TDM Mainframe components

Copybook Parser

The file definition manager will provide you with the facilities to parse thru a copybook to determine the record layouts that you will need to populate a TDM project.

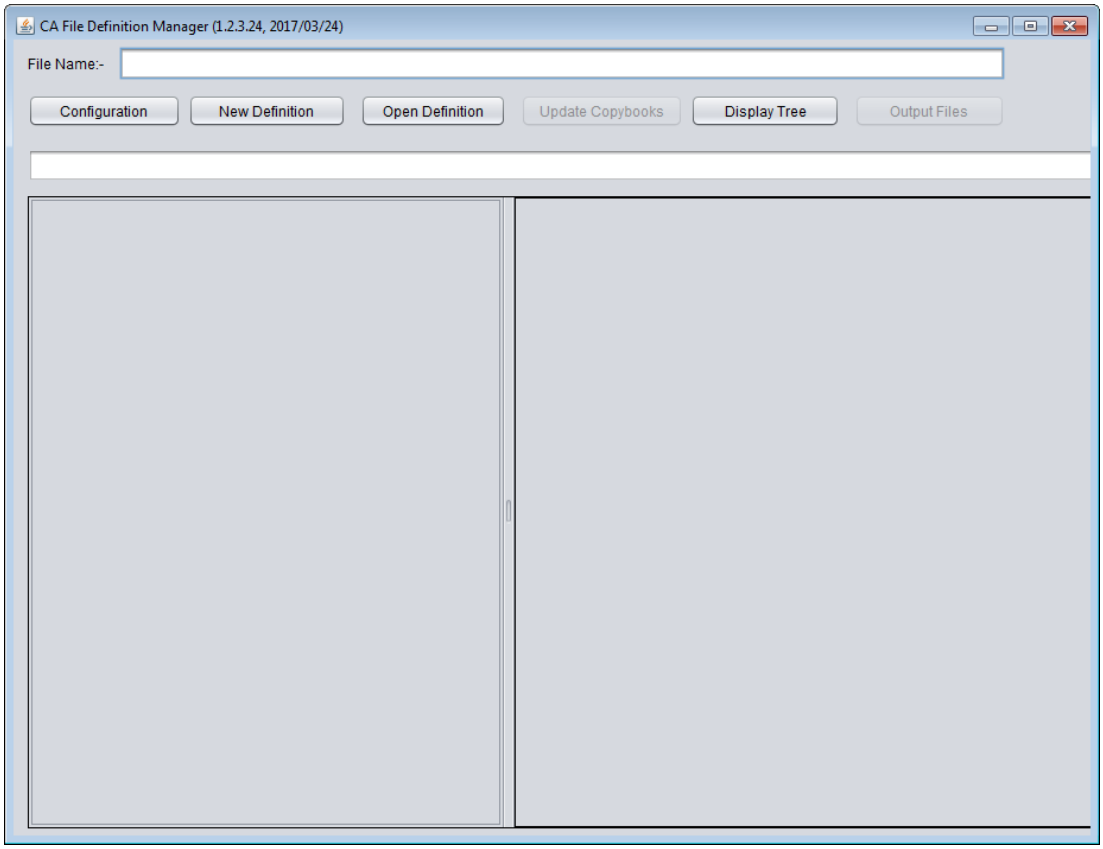


Figure: Default view of Copybook Parser

Before starting the parsing process, you will need to make sure of the following:

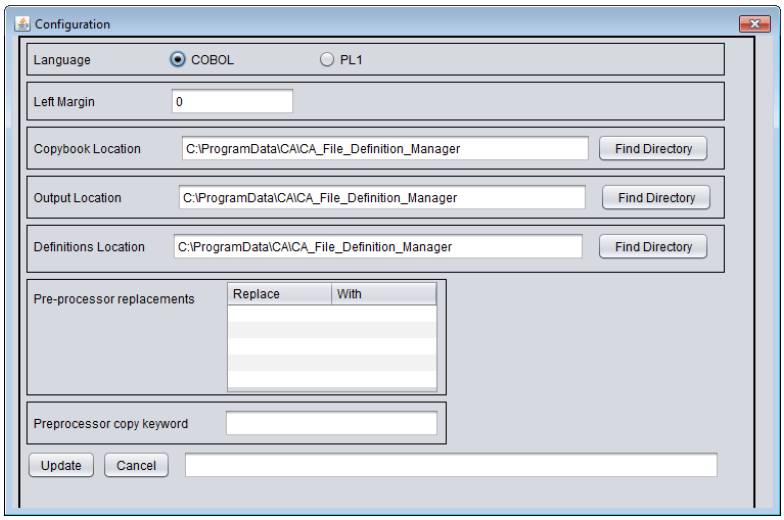


Figure: Example of the copybook parser configuration

- Download the copy book from the mainframe and save it, a good practice is to name the file: <sample_copybook>.cpy, since that is the Cobol naming standard.
- Setup your copybook parser configuration settings following the example above.

After you have setup your configuration settings, you can start the process of parsing a copy book.

Click the “New Definition” button to start the process. In the figure below, you are going to see the areas that you will have to change for a successful parsing session. After the parsing has been completed and the file saved, you can close this window.

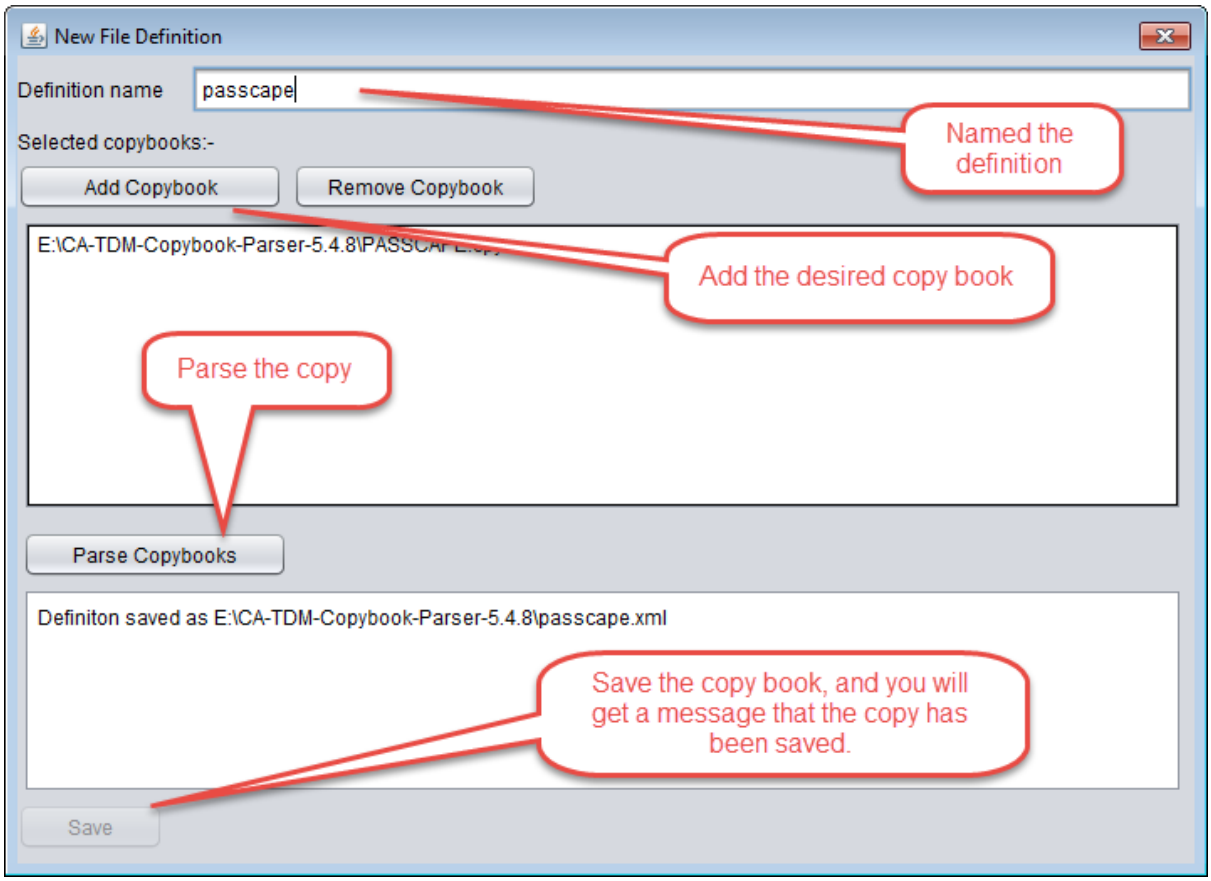


Figure: Copybook parsing window

Now that we are back at the main window, you need to click on the “Display Tree” button to see the copybook records and you can make condition changes

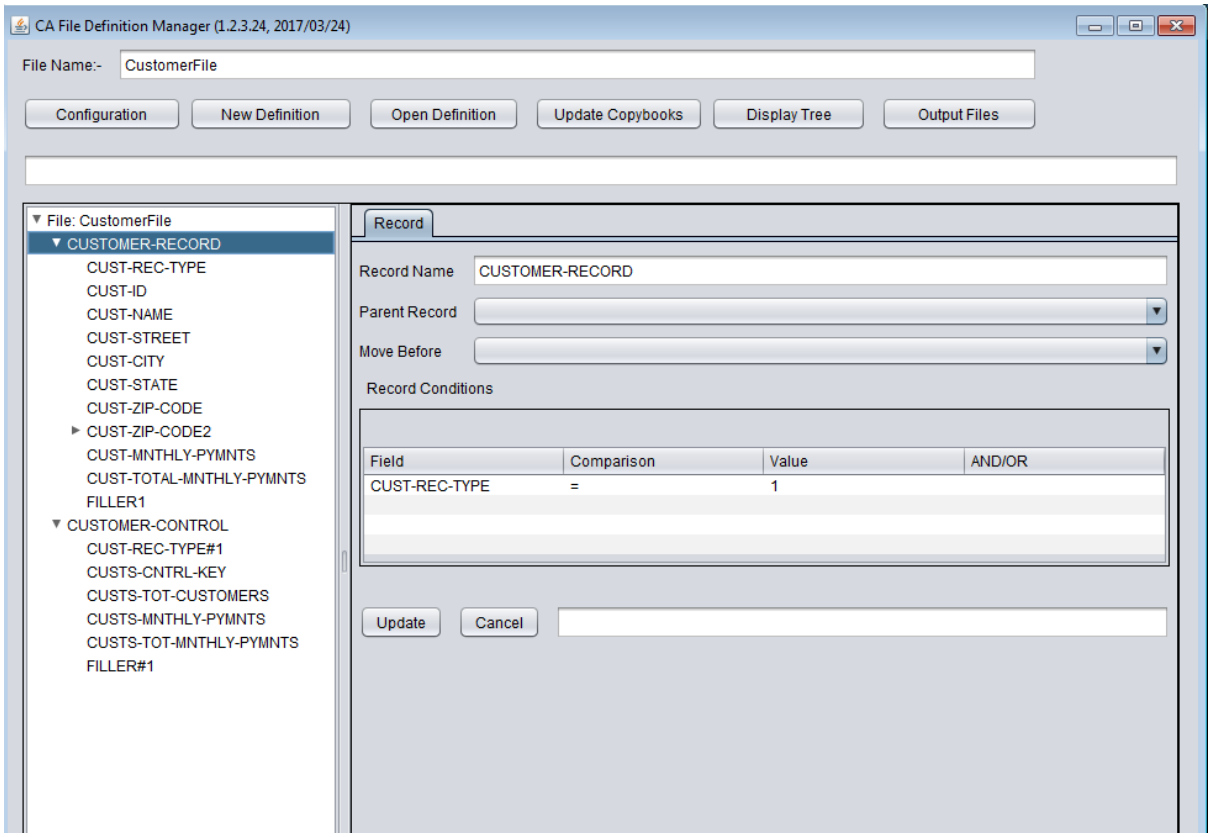


Figure: Copybook records being displayed

If you need to make changes to the records, and you have completed these changes, you can the necessary output files. This is accomplished by clicking the “Output Files” button.

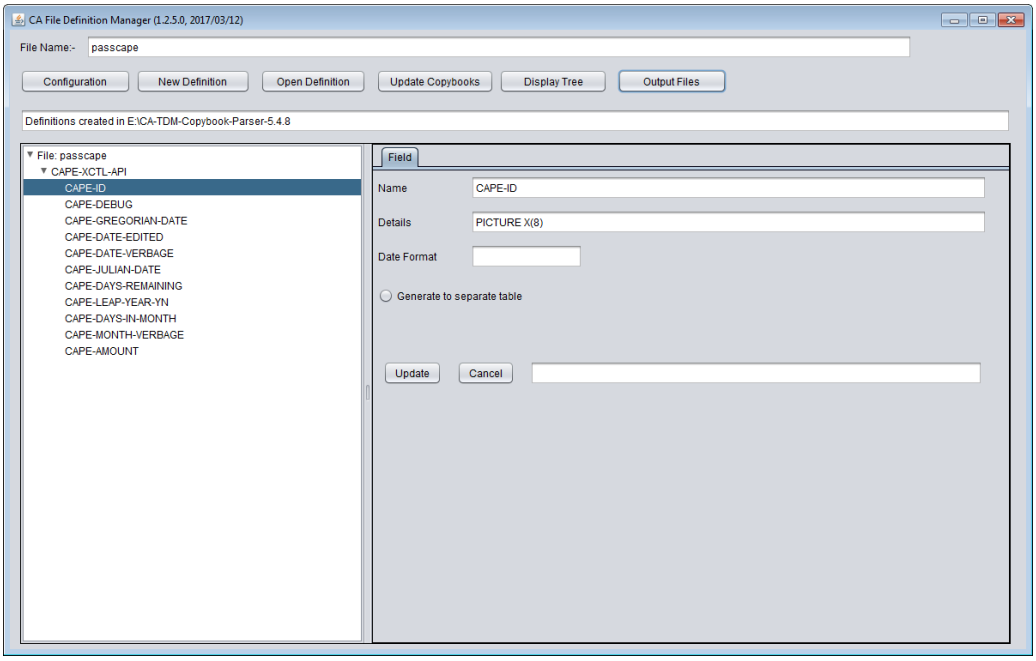


Figure: Generated output file in Copybook parser

Here is an example of the generated Excel Data Definition file.

A	B	C	D	E	F	G	H	I	J	K	L	M
PARAMETERS,HEADER=N,TRAILER=N,STYLE=SIMPLE,RECSEP=<EMPTY>,DELIM=<FIXED>,LINETERM=<CR/LF>,NULLIND=N,ID_OFFSET=0,ID_LENGTH=32												
Field Name	Mandatory	From	To	Length	Format	Column Name	Value Start	Value End	Default			
1	M	1	32	32	a	_RECNAME	CAPE_XCTL_API	CAPE_XCTL_API	CAPE_XCTL_API			
2		33	40	8	a	CAPE_ID						
3		41	48	8	a	CAPE_DEBUG						
4		49	56	8	n	CAPE_GREGORIAN_DATE						
5		57	66	10	a	CAPE_DATE_EDITED						
6		67	82	16	a	CAPE_DATE_VERBAGE						
7		83	89	7	n	CAPE_JULIAN_DATE						
8		90	92	3	n	CAPE_DAYS_REMAINING						
9		93	93	1	a	CAPE_LEAP_YEAR_YN						
10		94	95	2	n	CAPE_DAYS_IN_MONTH						
11		96	105	10	a	CAPE_MONTH_VERBAGE						
12		106	116	11	n	CAPE_AMOUNT						

Figure: Excelr Definition File contents

This file is imported into the desired TDM project. In addition to the XLS file that was generated, the supporting AFL files that will be needed for the data conversion/upload efforts will be generated as well.

Figure: Sample AFL generated

File Conversion

The file conversion utilities will provide you with the ability of printing, viewing the file contents (in ASCII and EBCDIC. The AFL files that were generated by the copy book parser will be used for the conversion process.

The conversion facilities also include a FujitsuNetCobol runtime, this document does not explain how to use said runtime and examples.

The file conversion utilities is composed of the following commands:

- Prt_ASCII.cmd → Print the contents of a file in ASCII
- Prt_EBCDIC.cmd → Pring the contents of a file in EBCDIC
- Convert.cmd → Convert file form ASCII to EBCDIC
- Convert2.cmd → Convert file from EBCDIC to ASCII

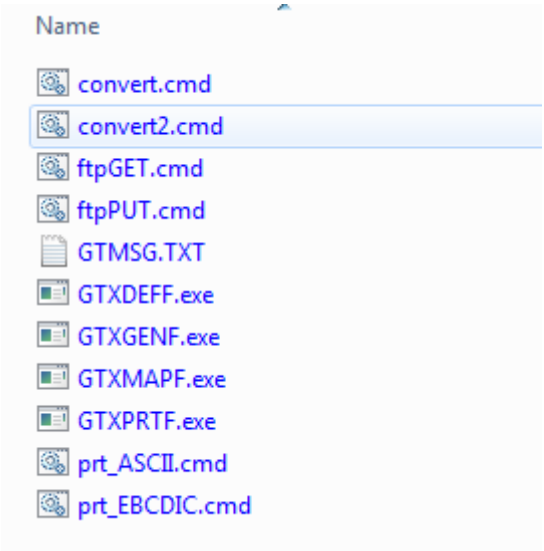


Figure: Typical file conversion listing

For the “convert.cmd”, you will need to update the following values as shown below.

```
@echo off

set EXEDIR=C:\CobolConv\FileConversion\Executables
set WORKDIR=C:\CobolConv\FileConversion\Work
set LOGFILE=C:\CobolConv\FileConversion\Work\Log.log
set ERRORFILE=C:\CobolConv\FileConversion\Work\Error.error

set SOURCEDMTXT=C:\CobolConv\FileConversion\TestCobolGen.DM.txt
set INPUTFILE=C:\CobolConv\FileConversion\TestFileGen.txt
set INPUTMODE=LS
set INPUTRECFM=
set INPUTLRECL=
set INPUTCODESET=ASCII

set TARGETDMTXT=C:\CobolConv\FileConversion\TestCobol.DM.txt
set OUTPUTFILE=c:\CobolConv\FileConversion\TestFile_ZOS.txt
set OUTPUTMODE=B
set OUTPUTRECFM=VB
set OUTPUTLRECL=300
set OUTPUTCODESET=EBCDIC
```

The *_ZOS.AFL.DM.txt will be placed here

The data file generated by TDM will be placed here.

The desired output files will be placed here.

Figure: Sample convert.cmd entries

For the “convert2.cmd”, you will need to change the “SOURCEDMTXT” to point to the “*_DG.AFL.DM.txt”, and the input file will be the file downloaded from the mainframe.

Mainframe Data Masking

The mainframe data masking facilities are design to help with the masking of DB2 or IMS datasets. These facilities provide you with consistent, robust, and repeatable methodologies for securing sensitive data.

Prior to the transfer of the XMI files, it is advisable that you pre-allocate these files in the mainframe based off the following values and defined as partitioned dataset files (PDS).

XMI Name	Source DSN	Member #	Approx space (KB)	Format
libdbrm	GRIDT01.LIB.DBRM	14	72	RECFM=FB,LRECL=80
libdef	GRIDT01.LIB.DEFCSV	2	9	RECFM=FB,LRECL=120
libload	GRIDT01.LOADLIB	14	3,439	RECFM=U,LRECL=80
libmap	GRIDT01.LIB.MAPCSV	2	17	RECFM=FB,LRECL=255
libparm	GRIDT01.LIB.PARM	7	21	RECFM=FB,LRECL=80
libproc	GRIDT01.LIB.PROCLIB	11	78	RECFM=FB,LRECL=80
libjcl	GRIDT01.LIB.RUNJCL	19	104	RECFM=FB,LRECL=80
libspufi	GRIDT01.LIB.SPUFI	9	18	RECFM=FB,LRECL=80

Sequential data files (PS)

XMI Name	Approx Space (KB)	Format
msgdata	112	RECFM=FB,LRECL=140
seedcard	6	RECFM=FB,LRECL=120
seeddat1	133,000	RECFM=FB,LRECL=2329
seeddat2	199,000	RECFM=FB,LRECL=2329
seeddat3	125,000	RECFM=FB,LRECL=2329
seeddat4	159,000	RECFM=FB,LRECL=2329
seeddat5	161,000	RECFM=FB,LRECL=2329
seeddat6	161,000	RECFM=FB,LRECL=2329
seeddat7	252,000	RECFM=FB,LRECL=2329
seeddata	55,204	RECFM=VB,LRECL=16384
testdata	20	RECFM=VB,LRECL=258

We are going to concentrate on the installation of the binaries that reside in the mainframe. These files need to be transfer in the native format with no conversion. Below are the contents of the ftp command needed based on the record type and length.

```
@echo off
echo user guewa01     > ftpcmdsupp.dat
echo <your password>        >> ftpcmdsupp.dat
echo bin             >> ftpcmdsupp.dat
echo quote mode b >> ftpcmdsupp.dat

echo quote site filetype=seq recfm=fb lrecl=80 blksize=3120 rdw >> ftpcmdsupp.dat
echo quote site filetype=seq recfm=fb lrecl=80 blksize=32720 rdw >> ftpcmdsupp.dat

REM SPACETYPE CYLINDER PRIMARY 2 SECONDARY 5
echo put .\XMI_Files\libdbrm.xmi 'guewa01.public.GRIDT543.lib.dbrm' >>
ftpcmdsupp.dat
echo put .\XMI_Files\libparm.xmi 'guewa01.public.GRIDT543.lib.parm' >>
ftpcmdsupp.dat
echo put .\XMI_Files\libproc.xmi 'guewa01.public.GRIDT543.lib.proclib' >>
ftpcmdsupp.dat
echo put .\XMI_Files\libjcl.xmi 'guewa01.public.GRIDT543.lib.runjcl' >>
ftpcmdsupp.dat
echo put .\XMI_Files\libspufi.xmi 'guewa01.public.GRIDT543.lib.spufi' >>
ftpcmdsupp.dat
```

```
rem now adding fb and 120 files
echo quote site recfm=fb lrecl=120 blksize=32640 rdw >> ftpcmdsupp.dat
echo put .\XMI_Files\libdef.xmi 'guewa01.public.GRIDT543.lib.defcsv' >>
ftpcmdsupp.dat
echo put .\XMI_Files\seedcard.xmi 'guewa01.public.GRIDT543.load.cards' >>
ftpcmdsupp.dat

rem now adding fb and 140 files
echo quote site recfm=fb lrecl=140 blksize=32620 rdw >> ftpcmdsupp.dat
echo put .\XMI_Files\msgdata.xmi 'guewa01.public.GRIDT543.msg.source' >>
ftpcmdsupp.dat

rem now adding fb and 255
echo quote site recfm=fb lrecl=255 blksize=32640 rdw >> ftpcmdsupp.dat
echo put .\XMI_Files\libmap.xmi 'guewa01.public.GRIDT543.lib.mapcsv' >>
ftpcmdsupp.dat

rem now adding u and 255
echo quote site recfm=u lrecl=255 rdw >> ftpcmdsupp.dat
echo put .\XMI_Files\libload.xmi 'guewa01.public.GRIDT543.loadlib' >>
ftpcmdsupp.dat

rem now adding vb and 258
echo quote site recfm=vb lrecl=258 rdw >> ftpcmdsupp.dat
echo put .\XMI_Files\testdata.xmi 'guewa01.public.GRIDT543.temp.test' >>
ftpcmdsupp.dat

echo put .\XMI_Files\libjcl.xmi 'public.GRIDT543.lib.runjcl' >> ftpcmdsupp.dat
echo quit            >> ftpcmdsupp.dat
ftp -n -s:ftpcmdsupp.dat ca06.ca.com
rem del ftpcmdsupp.dat
```

After the XMI files have been uploaded, you can start the installation of the DB2 and VSAM files by executing the following commands at the mainframe level:

- DB2 installation
    - Gridt01.lib.runjcl(receive)
- VSAM installation
    - Gridt01.lib.runjcl.(receivev)

The “gridt01.lib.runjcl” JCL procedure needs to be modified to reflect the correct HLQ (High Level Qualifier) of where the XMI files have been installed, in this example that will be “gridt01”

For the DB2 support, if the schema where you have installed the following reference tables:

- “gtsrc_reference_lov1”
- “gtsrc_xref”
- “gtsrc_subset”

The above tables are created by the “gridt01.lib.spufi”, and you might need to update the schema where you would like to have these tables created.

Once you have successfully installed the mainframe data masking facilities, you can use the “Mainframe Datamaker User Guide” for the data generation, transformation, and loading into the DB2 for z/OS installation.

Best Practices

The following best practices will help you in being successful in parsing, converting, and masking the transformed DB2 and VSAM (IMS) datasets.

Planning

It is important that you understand how the copy book records and how these records interact with the different datasets.

Copybooks

Understand the different record layouts and how the different arrays within the given record layout are defined. You need to pay very close attention to dynamic records, which are usually defined towards the end of the copybook, but there are times a dynamic array will be in the middle of the copy book.

Understanding the differences between Signed and computational fields, since these fields translated into signed packed data fields in the corresponding dataset.

Reviewing the generated Excel data definition to make sure that the resulting fields match the copybook record layout.

Conversion facilities

It is advisable that you create copy of the shipped cmds files to reflect the copybook projects that you are working at that point in time. This will reduce the confusion with the different copybook projects that you might be working on. Try to maintain a one to one relationship with the TDM project or data pool.

Test out the ftp commands with small dataset files prior to using the production file that you will use for transferring of the generated dataset.

Mainframe Data Masking

Make sure that you create the partitioned data sets (PDS) and sequential files (PS) prior to the loading of the XMI files. Make that the file allocations have enough storage capacity, this will allow you to upload the XMI files with confidence.

Make sure that you have sufficient rights to the DB2 schemas (read/write/alter authorizations), at the same time make sure that you have setup DB2 connect and tested this connection from the system where TDM is installed. Add an ODBC entry to TDM that points to the DB2 subsystem in the mainframe.

You might to load one table at a time with the corresponding generated dataset.

Useful Links

https://docops.ca.com/ca-test-data-manager/4-0/en/installing/mainframe-installation-and-upgrade/install-mainframe-components-v5-4

https://docops.ca.com/ca-test-data-manager/4-0/en/reference/mainframe-reference/mainframe-file-conversion

http://3480-3590-data-conversion.com/article-reading-cobol-layouts.html

http://3480-3590-data-conversion.com/article-redefined.html

http://www.tutorialspoint.com/cobol/cobol_data_layout.htm