DATABASE CONVERTER PLUS USER GUIDE

DOCUMENT VERSION: 1.0

JANUARY 26, 2005

CONTENTS

1. INTRODUCTION 3

1.1 FEATURES 3

2. BUILD THE COMMAND LINE 4

2.1 USING GUI TO BUILD THE COMMAND LINE 4
2.2 USING COMMAND LINE OPTIONS TO BUILD THE COMMAND LINE 6
2.2.1 Notes about the command line options 7
2.2.2 Examples of using command line options 8
2.3 GENERATE THE REFERENCE PDB STRUCTURE TO USE IN YOUR PROGRAM 8
2.3.1 Using GUI 8
2.3.2 Using Command-line options 9
2.4 HOW TO USE THE GENERATED PDB IN YOUR PROGRAM: 9
2.4.1 Examples 9
2.5 END USER DEPLOYMENT 13

3. APPENDIX A: PALM AND DESKTOP DATATYPE CONVERSION 14

4. APPENDIX B: ABOUT THE EVALUATION VERSION 16

LIST OF FIGURES

Figure 1: Database Converter GUI.....5
Figure 2: Applying Filters.....5
Figure 3: Applying Sort Order.....6

1. INTRODUCTION

Database converter plus is an ultimate utility for palm developers, to convert any desktop database table(s) to Palm Database (.PDB) in their own application. It will help you to convert the contents of your desktop databases like MS Access, Excel, Oracle, SQL Server, FoxPro, dBase and any other ODBC enable Database.

Resultant PDB record format is given in log file so that you can use it in your palm development. The converted PDB is in the standard format as recommended by Palm OS. Any development tool like Metrowerks CodeWarrior that supports standard Palm PDB format or "NS Basic" can use this PDB.

1.1 Features

- Has Easy GUI to generate command line string.
- Support for non-file database like Oracle, SQL Server etc.
- Support for MS Access, Microsoft Excel Sheets/tables, FoxPro, dBase and other file databases
- Apply SQL Select queries on database to purify records
- Use DSN to connect to any ODBC enable database
- Filter database fields as per your requirement
- Standard Palm OS supported PDB format
- Text file (*.txt), comma separated file (*.csv) or tab delimited files can also be converted to Palm PDB.

2. BUILD THE COMMAND LINE

This section describes how to use theDBConverterPlus.exe utility to build the command line that you can invoke in your software.

When you install Cellica DatabaseConverterPlus Software it will install following three files: DBConverterPlus.exe, DBConverterPlus.dll and DBConverterPlusUserGuide.pdf

You can use DBConverterPlus.exe two ways; either as a GUI to build the command line or you can provide command line options to it to build the command line.

2.1 Using GUI to build the command line

This GUI help you to easily create command line parameter string and if you want to generate PDB of specific database before use of command line utility.

To open GUI specify OPENGUI parameter with DBConverterPlus utility.

```
C:\Program Files\Database Converter Plus\DBConverterPlus OPENGUI
```

Once GUI open, follow the steps to generate command line.

1. Select your desktop database using DSN from list box or File database using browse button.
2. You will get Table Name list, Select the table from list of Table Names.
3. You will get the field names along with its data type and content of the table.
4. Additionally if you want to provide filter or sort, select the appropriate push buttons or update SQL query manually
5. Select Creator Id from list box. Demo version work with the creator “CELL” only.
6. Enter Type id of your application. Default value of this Type is “CELL”
7. Press "Save As" button to generate .PDB at specified location by using the mentioned Creator and Type id.
8. Press "Generate" button to generate command line string. It is available in the command line string box; use this string in your application to generate PDB from command prompt.

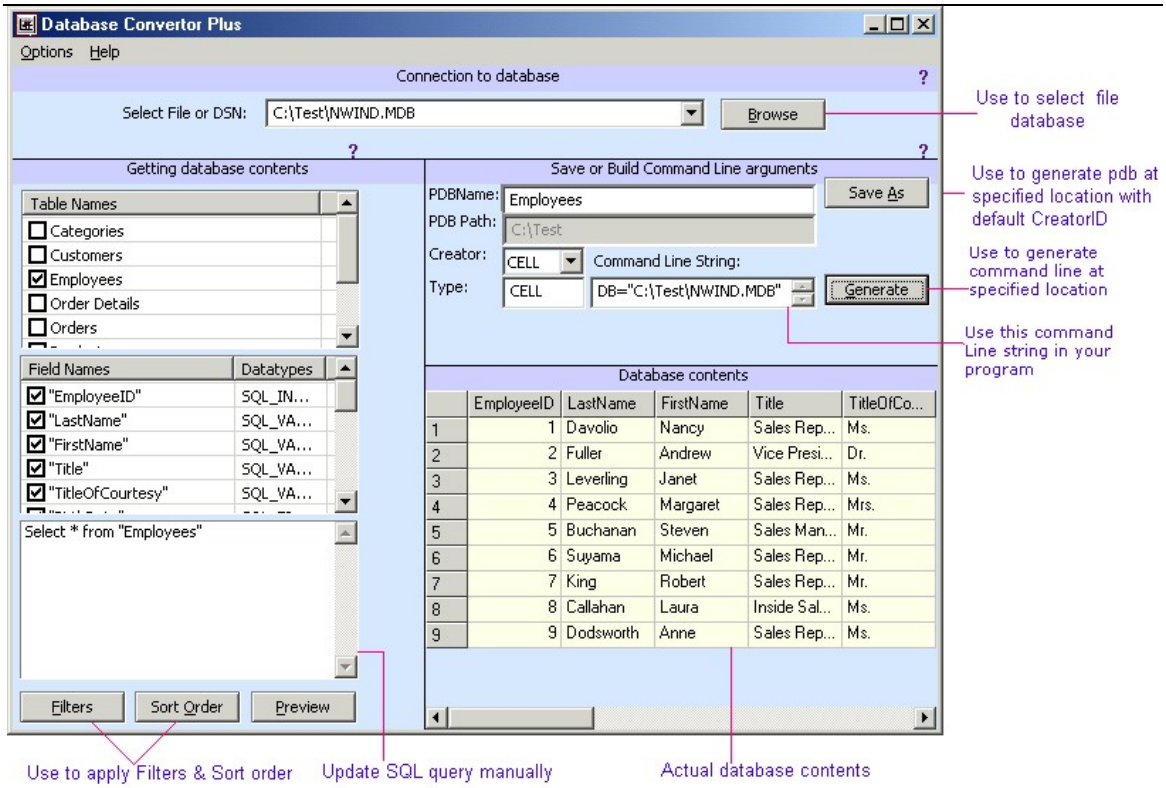

Figure 1: Database Converter GUI

You can apply the Sort Order and Filters in the following manner.

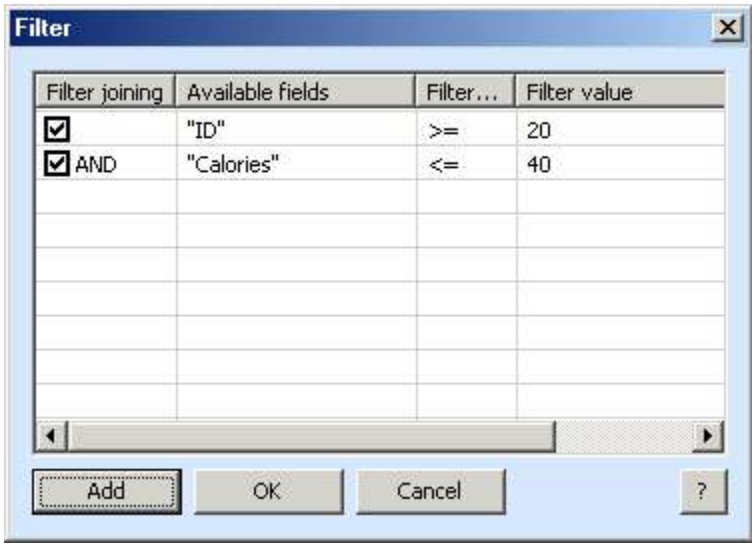

Figure 2: Applying Filters

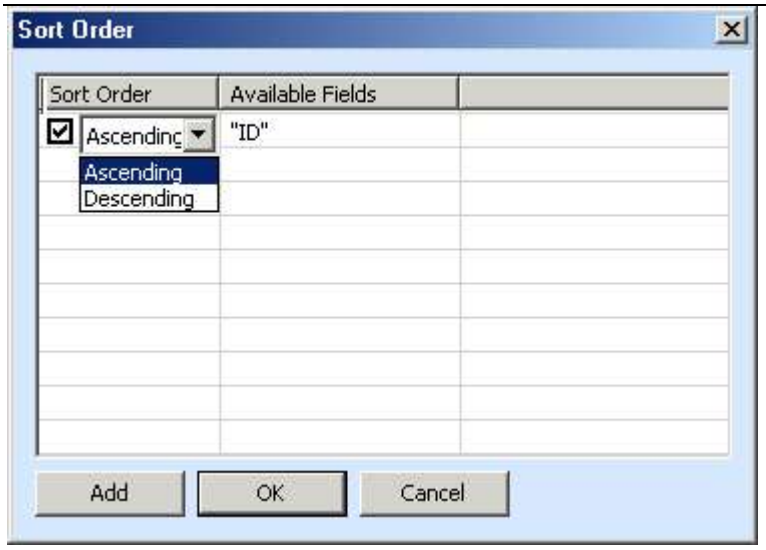

Figure 3: Applying Sort Order

2.2 Using command line options to build the command line

You can use following command line options or parameters to build the command line that you can use in your software.

Parameters	Comment	Usage
DSN	Data Source Name available on target machine	Either DSN or DB parameter, if passed both, only DSN will be considered.
DB	Database path along with name, for example:"C:\Test\NWnd.mdb"	
DatabaseType	Type of database that you entered as DB parameter. If you entered DB Parameter, you must include this parameter too	DatabaseType value must be one of the following: mdb, xls, dbf, csv or txt
Query	Any valid SQL Select query applicable for specified database	Either Query or Sheet parameter is necessary. Query parameter must be end with "EndQuery" For example, Query=Select * from "employee" EndQuery
Sheet	Valid Excel Sheet name for given excel file. Applicable only if DB parameter specified with valid Excel file.	
User	(Optional) Database authentication user, if required for the specified database	For example, User="Scott"
Password	(Optional) Database authentication password, if required for the specified database	For example Password="tiger"

PDBPath	Path for output PDB to generate. Same path also used for log file generation, if specified.	Pass valid path where PDB should be generated. If log files are specified, these will also be generated at this path.
PDBName	Name to use for output PDB	
Creator	PDB Creator for output PDB	PDB Creator must be of 4 characters. e.g. Creator="CELL"
Type	PDB Type for output PDB	PDB Type must be of 4 characters. e.g. Type=CELL
LogGen	(Optional) Specify whether Log to be generated along with output PDB	For example, LogGen ="yes" or LogGen="y" or LogGen="no" or LogGen="n"
Silent	(Optional) Silent mode. Specify if error/help message to be displayed or not	For example, Silent="yes" or Silent="no" or Silent="y"

2.2.1 Notes about the command line options

- Parameters are not case sensitive.
- Parameters can be in any sequence.
- Parameter values may be within quotations mark (") except Query.
- Either DSN or DB parameters should be used. Both parameters should not be used at a time, in such case, DSN will only be considered.
- Either Query or Sheet should be passed. Query is for general purpose and Sheet is for Excel sheet name. Sheet parameter will only be used in conjunction with DB having valid excel file path. Query parameter for excel file will also work.
- Query parameter values should not be within quotation mark ("). However, quotation mark may be part of query.

  Valid:

  Query=Select * from "Employee" EndQuery

  Invalid:

  Query="Select * from Employee" EndQuery
- Sheet parameter will create an automation object of Excel Worksheet and will retrieve excel sheet contents as displayed in Excel worksheet. If you want to use Sheet parameter, you must sure that target machine is having MS Excel installed there. This parameter will create PDB having all fields with text i.e. string type.

- You may optionally use quotation mark (") to specify value of parameter. But for Query parameter, do not use quotation mark, however, quotation mark may be part of SQL select query.

2.2.2 Examples of using command line options

- `DBConverterPlus DB="C:\test\test1.mdb" DatabaseType=mdb Query=Select * from "Employee" EndQuery User=Admin Password=""Tiger"" PDBPath="c:\test" PDBName="Employee" Creator="fffd" Type="CELL" LogGen=Yes Silent=No`
- `DBConverterPlus DSN=OracleDSN Query=Select * from tab EndQuery User=scott Password="tiger" PDBPath="C:\TestFiles" PDBName="Employee" Creator="EMPD" Type="DATA" LogGen=Yes Silent=Yes`
- `DBConverterPlus DB="C:\Test\TestExcel.xls" DatabaseType=xls Query=Select * from "sheet1$" EndQuery Password="tiger" PDBPath="C:\TestFiles" PDBName="Employee" Creator="EMPD" Type="DATA" LogGen=Yes Silent=Yes`
- `DBConverterPlus DB="C:\Test\TestExcel.xls" Sheet=sheet1  PDBPath="C:\TestFiles" PDBName="Employee" Creator="EMPD" Type="DATA" LogGen=Yes Silent=Yes`

2.3 Generate the Reference PDB Structure to use in your program

You will need to perform this step to generate the PDB structure you can refer in your program.

There are two ways the reference file is generated. One way is using the GUI and another way is using command line options.

Also, **cmdLine.log** file will be generated in your application installation path. If you are experiencing any problem in the PDB conversion, please send this log file to us via email on support@cellica.com.

2.3.1 Using GUI

When you click on **[Save As]** button, a reference file with the name DBConverterLog.txt will be generated in the same directory where you saved your PDB file.

Important: Please rename this file or move it to different location, as it will get overwritten next time when you click on **[Save As]** button.

2.3.2 Using Command-line options

By default (or if you specify LogGen=no) a reference file with the name DBConverterLog.txt will be generated in the same directory you specified in PDBPath command line option.

Important: Please rename this file or move it to different location, as it will get overwritten next time when you specify the same PDBPath.

If you specify LogGen parameter as “yes”, the reference file with your PDBName as prefix will be generated in the directory specified by PDBPath option. For example if you specify PDBName=MyPDB and PDBPath=C:\temp then the reference file MyPDBLog.txt will be created in C:\temp directory.

2.4 How to use the Generated PDB in your program:

In order to use PDB generated from DBConverterPlus, you need to have Palm Software Development tool like “Metrowerks CodeWarrior”, "NS-Basic" etc.

Use following procedure:

1. Install PDB on your device or Palm OS Emulator/Simulator
2. In your program, define a structure same as given in the reference file (described in section 2.3 above)
3. Access records using above defined structure. You will get the actual record contents present in PDB.

2.4.1 Examples

Suppose you generated a PDB “Products.PDB”. ProductsHelper.mhl helper file also gets generated with this PDB.

2.4.1.1 Using Metrowerks CodeWarrior

Following is an example shown in Metrowerks CodeWarrior to use this PDB.

1. Install Products.PDB file on your device or Emulator/Simulator
2. Create a simple “C application” using “Palm OS Application Stationary” wizard
3. Define a structure as given below. This structure is copied from the reference file (described in section 2.3 above).

```
typedef struct {
    long ProductID;
    long CategoryID;
    short int ProductSectionID;
    DateTimeType ExpiryDate;
}ProductRecType;
```

You can use this record format in your code to retrieve/manipulate PDB records. Put following code in some user defined function, for example: TestProduct

```
void TestProduct(void)
{
    unsigned long appCreator   =     'TEST';
    unsigned long dbType        =     'prdt';

    DmOpenRef dbRef;
    MemHandle handle;
    long totalRecords, recCount;
    ProductRecType *pRecord;

    dbRef=DmOpenDatabaseByTypeCreator(dbType, appCreator,      dmModeReadOnly);
    totalRecords = DmNumRecords(dbRef);
    for(recCount=0;recCount<totalRecords;recCount++)
    {
        handle = DmQueryRecord(dbRef,recCount);
        pRecord = MemHandleLock(handle);


        //Process pRecord (Verify, use ...)
        //……
        //……

        MemHandleUnlock(handle);
    }
}
```

In log file, if you were given a structure containing one or more string, then you cannot read this record directly. Instead you need to read these strings separately.

Please see the example is below.

Suppose you were given following structure in log file.

```
typedef struct {
    char name[ ];
    char address[ ];
}PersonInfo;
```

Use following program portion to read it.

```
void TestProduct(void)
{
    unsigned long appCreator = 'TEST'; //given in log
    unsigned long dbType = 'data';

    DmOpenRef dbRef;
    MemHandle handle;
    long totalRecords, recCount;
    char * ptr;
    char * name, *address;


    dbRef=DmOpenDatabaseByTypeCreator(dbType, appCreator, dmModeReadOnly);
    totalRecords = DmNumRecords(dbRef);
    for(recCount=0;recCount<totalRecords;recCount++)
    {
        handle = DmQueryRecord(dbRef,recCount);
        ptr = MemHandleLock(handle);

        name = MemPtrNew(StrLen(ptr)+1);
        StrCopy(name,ptr);
        ptr += StrLen(ptr)+1;

        address = MemPtrNew(StrLen(ptr)+1);
        StrCopy(address,ptr);

        //Process name & addrss ( Verify, use ...)
        //……
        //……

    MemHandleUnlock(handle);
    }
}
```

2.4.1.2 Using NS-BASIC

Following is an example shown in NS Basic to use this PDB.

Suppose you were given following structure in log file.

```
typedef struct {
    long ProductID;
    long CategoryID;
    short int ProductSectionID;
    char ProductName[];
    char ProductDetails[];
}ProductInfo;
```

User following NS-Basic code, to get PDB contents:

```
Type ProductInfo
    ProductID as Integer
    CategoryID as Integer
    ProductSectionID as Short
    ProductName as String
    ProductDetails as String
End Type
Dim Db as Database
Dim res as Integer
Dim rec as ProductInfo
Dim total as Integer
Dim count as Integer
res=DbOpen(Db,"Products",0)     ' PDB name is case sensitive for NS-‘Basic
If res = 0 Then
    total = DBGETNORECS(db)
    For count =1 to total total
        res=DbPosition(Db,count,0)
        res=DbGet(Db,rec)

        ' Record contents are in rec,
        'Operate on rec
    Next
  DbClose(Db)
end if
```

User can retrieve **PDB** record contents by individual members of record structure, instead of "User defined Type".

Important: In order to use this **PDB** in NS Basic

- Make sure **PDB** name is case sensitive
- Use equivalent data type available in NS-Basic having similar data size, instead of using same data types given in log e.g. for float given in log, use Single in NS-Basic
- Date, Time data types are not same in NS-Basic as supported by palm, in such case, use user defined type given in log

For using Date Time in NS-Basic, refer to the following example

Support you were given a structure

```
typedef struct {
    DateTimeType MfgDate;
    long ProductID;
    long CategoryID;
    short int ProductSectionID;
    DateTimeType ExpiryDate;
}ProductRecType;
```

Define following user defined type for NS-Basic

```
Type  DateTimeType
    second as Short
    minute as Short
    hour as Short
    day as Short
    month as Short
    year as Short
    weekDay as Short
End Type
Type ProductRecType
    ProductID as Integer
    CategoryID as Integer
    ProductSectionID as Short
End Type
Dim Db as Database
Dim res as Integer
Dim rec as ProductRecType
Dim mfgDate as DateTimeType
Dim expDate as DateTimeType
Dim total as Integer
Dim count as Integer
res=DbOpen(Db,"Products",0)
If res = 0 Then
    total = DbGetNoRecs(db)
    For count =1 to total total
        res=DbPosition(Db,count,0)
        DbGet(Db, mfgDate)
        DbGet(Db,rec)  ' retrieve remaining contents using user defined
'Type or access individual members
 DbGet(Db, expDate)

        ' Record contents are in rec, mfgDate, expDate,
        'Operate on record contents
    Next
    DbClose(Db)
End If
```

2.5 End User Deployment

You have to include following files in your setup program

DBConverterPlus.exe

DBConverterPlus.dll

Note: Make sure that DBConverterPlus.dll is registered version from Cellica. Do not use the evaluation version for the end user deployment.

3. APPENDIX A: PALM AND DESKTOP DATATYPE CONVERSION

Database converter maps Desktop data types to nearest possible palm **PDB** data type. Following table gives the details of desktop data type mappings to Palm data types.

Desktop Data Type	**Support**	**Palm Data Type**	**Comment**
Text/Char	Yes	char []	Sequence of characters terminated by '\0'. (Array of char of unlimited size i.e. a String)
Memo/Varchar/Varchar2	Yes	char []	Same as above
Number, Byte	Yes	unsigned char / Byte	Size = 1 bytes
Number, Integer	Yes	short int	Size = 2 bytes
Number, Long Integer	Yes	long	Size = 4 bytes
Number, Single	Yes	float	Size = 4 bytes
Number, Double	Yes	double	Size = 8 bytes
Number, Replication ID	Yes	char []	String
Number, Decimal	Yes	char []	String
Number, Real/Float	Yes	float	Size = 4 bytes
Numeric (Oracle, FoxPro...)	Yes	char []	String
Date/Time	Yes	DateTimeType	Palm DateTimeType structure
Date	Yes	DateType	Palm DateType structure
Time	Yes	TimeType	Palm TimeType structure
Currency	Yes	char []	String
GUID/ROWID	Yes	char []	String
Replication ID	Yes	char []	String
Yes/No (Logical)	Yes	unsigned char/Byte	Size = 1 byte

M LS Label	Yes	char []	String
Hyperlink	Yes	char []	Sequence of characters terminated by ‘\0’
OLE Object	No	n/a	
Binary	No	n/a	

4. APPENDIX B: ABOUT THE EVALUATION VERSION

Evaluation copy of the Database Converter Plus works on the basis of creator id. Evaluation version uses Creator and Type id “CELL”.

To register the application for your Creator Id, you need to send us only the creator id of the application for which you want to use the utility.

We will send you DBConverterPlus.dll. Just use this DLL to replace the evaluation version of the DLL. This DLL located in your application installation path.

You can purchase it from our website: http://www.cellica.com/dbconverterplus.htm

The price quoted on the website is for one Creator ID only, and hence the software is licensed for use with one Creator ID. For using the software with more than one Creator ID, you need to purchase the additional licenses. Please send the email to support@cellica.com for purchasing the additional licenses.