

CipherLab

Data Converter Programming Guide

C++ & .NET Programming

For 8 Series Mobile Computers

Version 3.03



Copyright © 2012~2016 CIPHERLAB CO., LTD.
All rights reserved

The software contains proprietary information of CIPHERLAB CO., LTD.; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

Due to continued product development this information may change without notice. The information and intellectual property contained herein is confidential between CIPHERLAB and the client and remains the exclusive property of CIPHERLAB CO., LTD. If you find any problems in the documentation, please report them to us in writing. CIPHERLAB does not warrant that this document is error-free.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of CIPHERLAB CO., LTD.

For product consultancy and technical support, please contact your local sales representative. Also, you may visit our web site for more information.

The CipherLab logo is a registered trademark of CIPHERLAB CO., LTD.

All brand, product and service, and trademark names are the property of their registered owners.

The editorial use of these names is for identification as well as to the benefit of the owners, with no intention of infringement.

CIPHERLAB CO., LTD.

Website: <http://www.cipherlab.com>

RELEASE NOTES

Version	Date	Notes
3.03	Aug. 23, 2016	C/C++ <ul style="list-style-type: none">▶ New: 1.2 How to Create a C++ Console Project in Linux▶ Modified: 1.3 – sample code for Linux updated
3.02	Oct. 28, 2015	C/C++ <ul style="list-style-type: none">▶ Modified: 1.2.1 AGXType – enumeration updated▶ Modified: 1.3.1 – sample code for AGX updated▶ New: 1.3.12 ProgressCallback .NET <ul style="list-style-type: none">▶ Modified: 2.4.1 AGXType – enumeration updated
3.01	Sep. 04, 2015	.NET <ul style="list-style-type: none">▶ Modified: 2.5.12 – example updated
3.00	Jan. 16, 2015	<ul style="list-style-type: none">▶ New infrastructure for C++ and .NETChapter 1 for C++Chapter 2 for .NET

- 2.04 Jun. 13, 2014 C/C++
- ▶ Modified: **1.3.5** – example updated
 - ▶ Modified: **1.3.6** – example updated
 - ▶ Modified: **1.3.7** – example updated
 - ▶ Modified: **1.3.8** – example updated
 - ▶ Modified: **1.3.9** – example updated
 - ▶ Modified: **1.3.10** – example updated
 - ▶ Modified: **1.3.11** – example updated
 - ▶ Modified: **1.3.12** – example updated
 - ▶ Modified: **1.3.13** – example updated
 - ▶ Modified: **1.3.14** – example updated
 - ▶ Modified: **1.3.15** – example updated
 - ▶ Modified: 1.4 PROGRESS_FUNC Callback Function
- .NET
- ▶ Modified: **2.6.2** – example updated
 - ▶ Modified: **2.6.3** – example updated
 - ▶ Modified: **2.6.4** – example updated
 - ▶ Modified: **2.6.5** – example updated
 - ▶ Modified: **2.6.6** – example updated
 - ▶ Modified: **2.6.7** – example updated
 - ▶ Modified: **2.6.8** – example updated
 - ▶ Modified: **2.6.9** – example updated
 - ▶ Modified: **2.6.10** – example updated
 - ▶ Modified: **2.6.11** – example updated
 - ▶ Modified: **2.6.12** – example updated
 - ▶ Modified: **2.6.13** – example updated
 - ▶ Modified: **2.6.15** – example updated
 - ▶ New: 2.7 ProgressCallback Callback function
- 2.03 Apr. 22, 2014 C/C++
- ▶ Modified: **1.3.5** – example revised
 - ▶ Modified: **1.3.6** – example revised
 - ▶ Modified: **1.3.7** – example revised
 - ▶ Modified: **1.3.8** – example revised
 - ▶ Modified: **1.3.9** – example revised
 - ▶ Modified: **1.3.10** – example revised
 - ▶ Modified: **1.3.11** – example revised
 - ▶ Modified: **1.3.12** – example revised
 - ▶ Modified: **1.3.13** – example revised
 - ▶ Modified: **1.3.14** – example revised
 - ▶ Modified: **1.3.15** – example revised
 - ▶ New: 1.4 PROGRESS_FUNC Callback Function

2.02	Apr. 01, 2014	<ul style="list-style-type: none"> ▶ Modified: 1.3.11 Pc2SdBasicProg example code revised ▶ Modified: 1.3.12 Pc2SdCProg example code revised ▶ Modified: 1.3.13 Pc2SdForgeAg example code revised
2.01	Jan. 15, 2014	<ul style="list-style-type: none"> ▶ Modified: 1.2.2 AG_RECORD_FORMAT – 12 record fields for 8600 ▶ Add: 1.2.5 SRCSTRUCT_PC2RAM2_BASIC (for 8600) ▶ Add: 1.2.7 SRCSTRUCT_PC2RAM2_CPROG (for 8600) ▶ Add: 1.2.11 SRCSTRUCT_PC2RAM2_FORGEAG (for 8600) ▶ Add: 1.3.6 Pc2RamBasicProg2 (for 8600) ▶ Add: 1.3.8 Pc2RamCProg2 (for 8600) ▶ Add: 1.3.10 Pc2RamForgeAg2 (for 8600) ▶ Modified: 2.4.3 DataConverterAPI – method table updated ▶ Add: 2.6.4 DataConverterAPI.Pc2Ram2(BasicProgDataFormat...out string[]) ▶ Add: 2.6.7 DataConverterAPI.Pc2Ram2(CProgDataFormat...out string[]) ▶ Add: 2.6.10 DataConverterAPI.Pc2Ram2(ForgeAgProgDataFormat...out string[])
2.0	Oct. 02, 2013	<ul style="list-style-type: none"> ▶ Structures & Functions are revised according to : ▶ "DataConverterAPIReference(CPP)" ▶ "DataConverterAPIReference(.NET)" ▶ Modified: Appendix I – Error Codes updated ▶ C/C++ ▶ Removed: FILETYPE enumeration ▶ Removed: RECORDFLD_PC2RAM_CPROG structure ▶ Removed: RECORDFLD_PC2RAM_BASIC structure ▶ New: 1.3.3 GetApiVersion ▶ New: 1.3.4 GetErrorMessage ▶ .NET ▶ Removed: GetOutputFiles method ▶ Removed: GetVersion method ▶ Removed: Sd2Pc(string, string) method
1.01	Aug. 15, 2013	<ul style="list-style-type: none"> ▶ Modification: 2.1 Create a Visual C# Application – obsolete .dll files removed (mfc90u.dll & msvcr90.dll) ▶ Modification: 2.2 Create a Visual Basic Application – obsolete .dll files removed (mfc90u.dll & msvcr90.dll)
1.00	Apr. 03, 2012	Initial release

CONTENTS

RELEASE NOTES	- 3 -
INTRODUCTION	1
Development Tool	2
C++ PROGRAMMING	3
1.1 How to Create a C++ Console Project in Windows	3
1.1.1 Using Visual Studio IDE to Create a C++ Console Project	3
1.1.2 Importing Library	4
1.2 How to Create a C++ Console Project in Linux	7
1.2.1 For AMD64.....	7
1.2.2 For I386	11
1.3 Sample Code.....	17
1.4 Enumerations.....	21
1.4.1 AGXType	21
1.4.2 BasicProgDbfFileName	21
1.4.3 ConversionStatus	22
1.4.4 FileType.....	22
1.4.5 ForgeAgProgDbfFileName.....	23
1.4.6 Programmingtool.....	23
1.4.7 WrongFormatAction	23
1.5 Class	25
1.5.1 Agx.....	25
1.5.2 ApiVersion.....	27
1.5.3 BasicProgDataFormat.....	28
1.5.4 ConversionResult	30
1.5.5 CProgDataFormat.....	30
1.5.6 DataFormat.....	32
1.5.7 Dbf2TxtConverter	36
1.5.8 ErrorRecord	38
1.5.9 ForgeAgProgDataFormat	40
1.5.10 formatException.....	41
1.5.11 RecordField	42
1.5.12 ProgressCallBack	43
1.5.13 Txt2DbfConverter	44
1.5.14 Txt2PackedDbfConverter	50
.NET PROGRAMMING.....	61
2.1 Create a Visual C# Application.....	61
2.2 Create a Visual Basic Application	65
2.3 Converter Delegate.....	68
2.3.1ProgressCallback Delegate.....	68
2.4 Converter API Enumerations.....	69
2.4.1 AGXType	69
2.4.2 BasicProgDbfFileName	70
2.4.3 ConversionStatus	71

2.4.4 ForgeAgProgDbfFileName.....	72
2.4.5 Programming Tool.....	72
2.4.6 WrongFormatAction	73
2.5 Classes.....	74
2.5.1 AGX.....	74
2.5.2 BasicProgDataFormat.....	74
2.5.3 ConversionResult	76
2.5.4 CProgDataFormat.....	77
2.5.5 Dbf2TxtConverter	78
2.5.6 DelimiterChar	80
2.5.7 ErrorRecord	80
2.5.8 ForgeAgProgDataFormat	81
2.5.9 GeneralInfo.....	83
2.5.10 RecordField	83
2.5.11 Txt2DbfConverter.....	84
2.5.12 Txt2PackedDbfConverter	86
2.6 Class Methods	90
2.6.1 AGX.Read	90
2.6.2 AGX.DetermineType	91
2.6.3 DataFormat.AddRecordField (RecordField).....	92
2.6.4 DataFormat.AddRecordField(List<RecordField>)	92
2.6.5 DataFormat.GetRecordFieldAt	93
2.6.6 DataFormat.ParseRecordFields	93
2.6.7 Dbf2TxtConverter.CancelAsync.....	94
2.6.8 Dbf2TxtConverter.Convert	94
2.6.9 Dbf2TxtConverter.Convert with Callback.....	94
2.6.10 Dbf2TxtConverter.ConvertAsync	95
2.6.11 Dbf2TxtConverter.ProgressChangedDelegate	95
2.6.12 Dbf2TxtConverter.RunWorkerCompletedDelegate.....	96
2.6.13 Txt2DbfConverter.CancelAsync.....	98
2.6.14 Txt2DbfConverter.Convert.....	98
2.6.15 Txt2DbfConverter.Convert with Callback.....	98
2.6.16 Txt2DbfConverter.ConvertAsync	99
2.6.17 Txt2DbfConverter.ProgressChangedDelegate	99
2.6.18 Txt2DbfConverter.RunWorkerCompletedDelegate.....	100
2.6.19 Txt2PackedDbfConverter.CancelAsync	101
2.6.20 Txt2PackedDbfConverter.Convert	102
2.6.21 Txt2PackedDbfConverter.Convert with Callback	102
2.6.22 Txt2PackedDbfConverter.ConvertAsync.....	102
2.6.23 Txt2PackedDbfConverter.ProgressChangedDelegate.....	103
2.6.24 Txt2PackedDbfConverter.RunWorkerCompletedDelegate	104
2.7 Class Events.....	106
2.7.1 Dbf2TxtConverter.ProgressChanged.....	106
2.7.2 Dbf2TxtConverter.RunWorkerCompleted	106
2.7.3 Txt2DbfConverter.ProgressChanged.....	107
2.7.4 Txt2DbfConverter.RunWorkerCompleted	107
2.7.5 Txt2PackedDbfConverter.ProgressChanged	108
2.7.6 Txt2PackedDbfConverter.RunWorkerCompleted.....	108

INTRODUCTION

This programming guide is meant for users who want to convert data files into various formats for the 8 series terminals.

Data Converter C++ API and C# API are functions performing file format conversion. Both the API sets support three types of format conversion as follows:

- ▶ Text → CipherLab 8-Series Mobile Computer RAM
- ▶ Text → CipherLab 8-Series Mobile Computer SD card
- ▶ CipherLab 8-Series Mobile Computer SD card → text

Whether you are using C++ or .NET Framework for programming, it is easy to write applications to convert data between PC and terminals. Supported data formats are Forge AG, DBF, and ASCII.

We recommend that you read the documents thoroughly before use and keep it at hand for quick reference.

Thank you for choosing CipherLab products!

DEVELOPMENT TOOL

It is assumed that the programmer has prior knowledge of Windows programming. For information on Windows application programming interface (API), essential online resource is available on Microsoft website –

<http://msdn2.microsoft.com/en-us/library/default.aspx>

The development environment and tools can be one of the following choices:

- ▶ Visual Studio 2008
- ▶ Visual Studio 2005

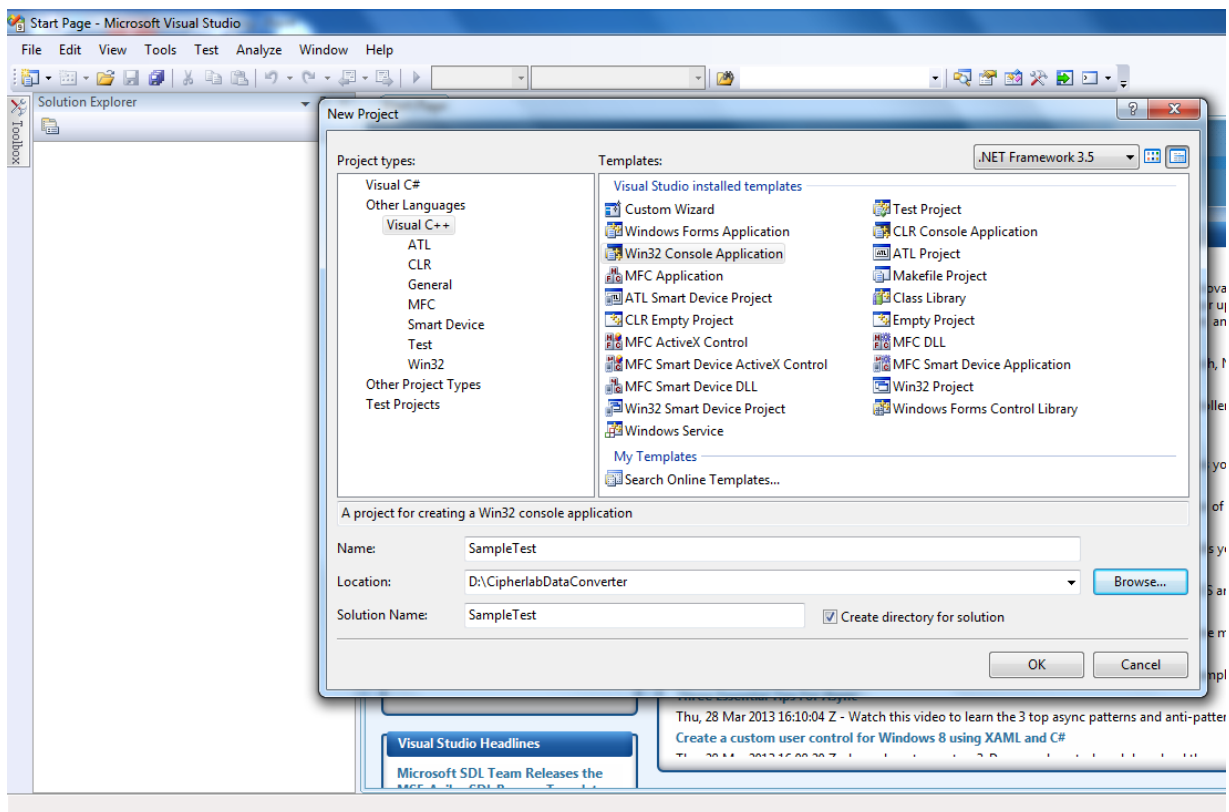
Note: For the target application to run correctly on a system without Visual Studio 2008, a prior installation of VC++ 2008 redistributable package is required.

C++ PROGRAMMING

1.1 HOW TO CREATE A C++ CONSOLE PROJECT IN WINDOWS

1.1.1 USING VISUAL STUDIO IDE TO CREATE A C++ CONSOLE PROJECT

- 1) After launching the Visual Studio integrated development environment (IDE), please point to **New** on the **File** menu, and then click **Project**.
- 2) When the **New Project** dialog shows up, click **Win32 Console Application** in the **Templates** pane.
- 3) Give a name for the project and then click **OK** to create the project.



1.1.2 IMPORTING LIBRARY

DATACONVERTER_CPP_API

The library files provided includes:

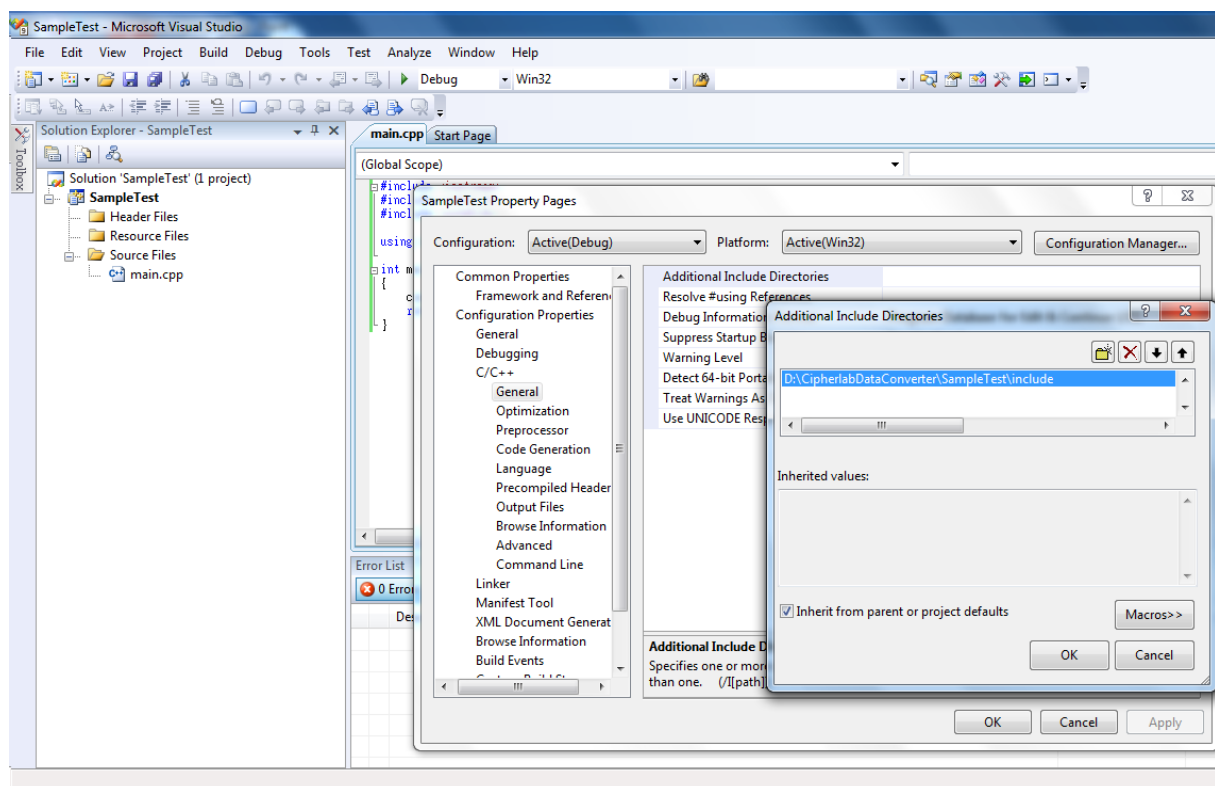
Folder	Description
\include	A bunch of header files
\lib	<i>CipherLab.DataConverter.lib</i> (Data Converter API library)
	<i>CipherLab.DataConverter.dll</i> (Data Converter API dll)

Please copy the files listed above into your project directory.

ADD THE LIBRARY HEADER DIRECTORY

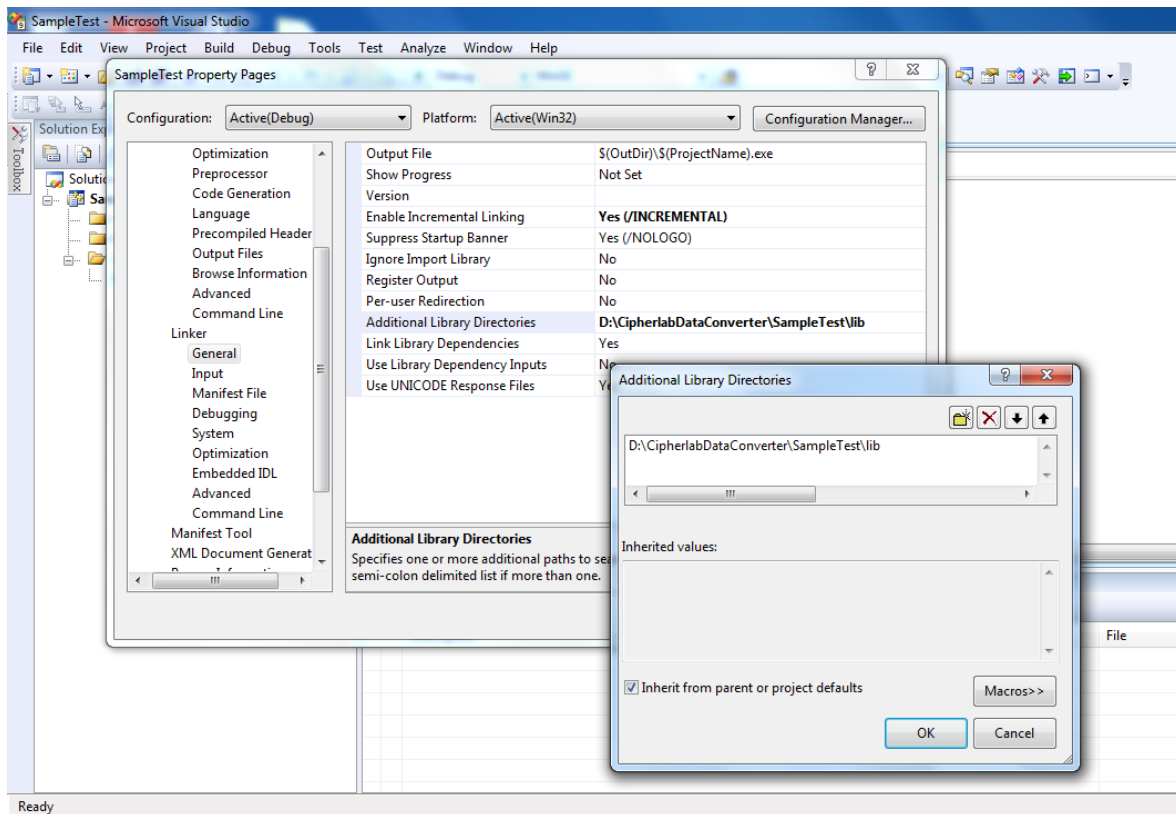
This is to tell the compiler where header files are located. Click **Properties** on the **Project** menu to bring up the project's properties dialog. In the left pane, click the plus sign to expand the **Configuration Properties** → **C/C++** branch; then click **General**.

In the **Additional Include Directories** field of the right pane, add the "*\include*" folder.

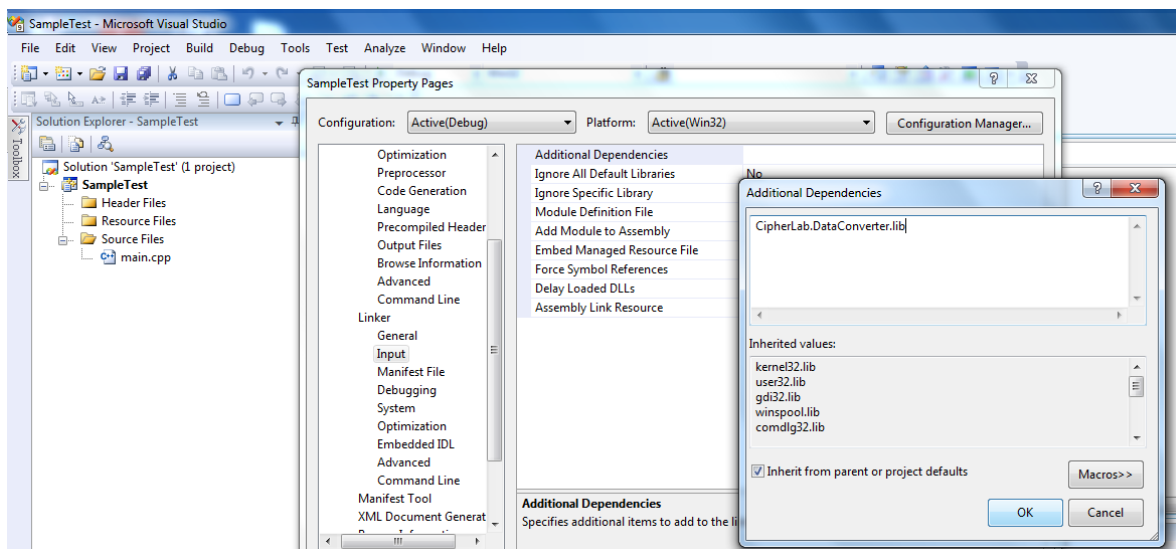


LINK TO STATIC LIBRARIES

This is to tell the linker where to find the library file(s). In the left pane, click the plus sign to expand the **Linker** branch; then click **General**. Then in the **Additional Library Directories** field of the right pane, add the "*\lib*" folder.



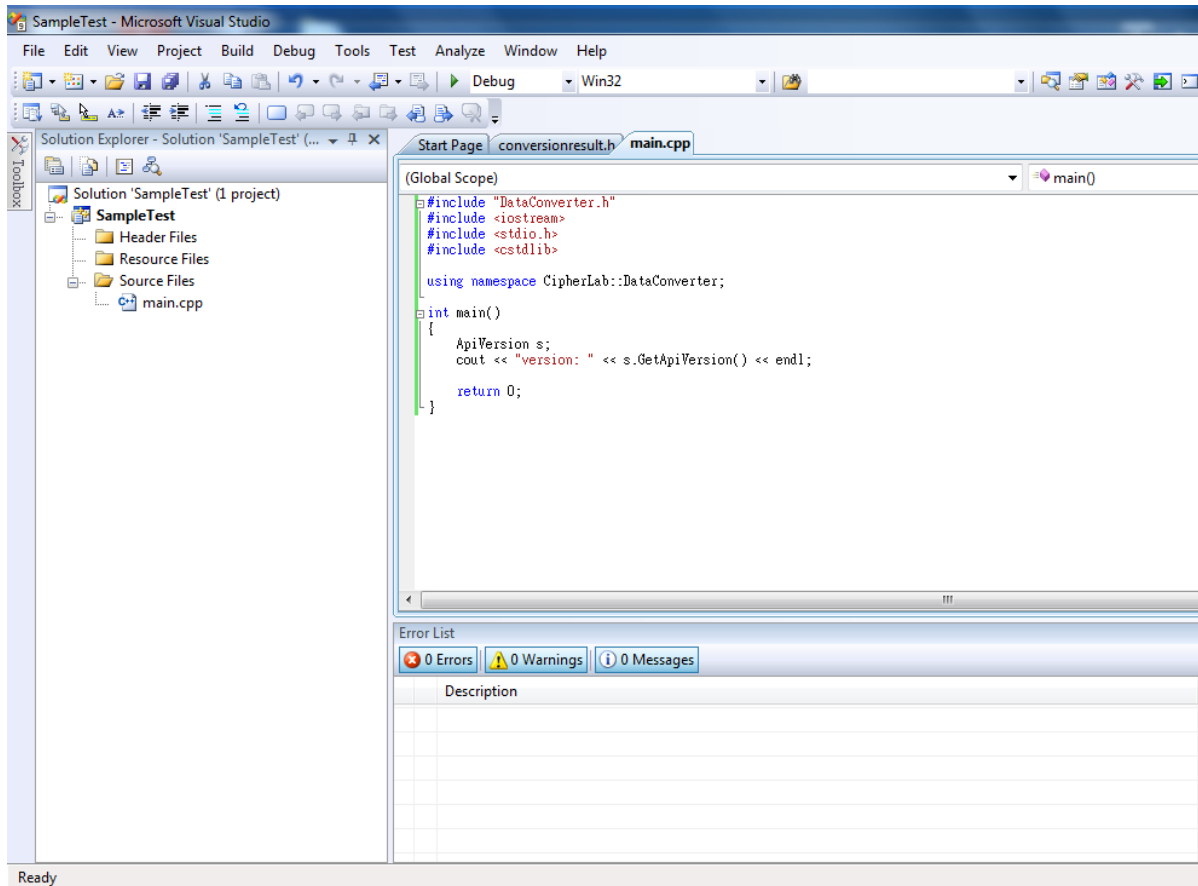
Next, you have to tell the linker what library file(s) to use. In the left pane, click **Input**. Then in the **Additional Dependencies** field of the right pane, add "*CipherLab.DataConverter.lib*".



LINK TO DLLS

Put *CipherLab.DataConverter.dll* in the same directory where the program's executable file is located.

In the main program, you could include "*DataConverter.h*" which contains all available headers.

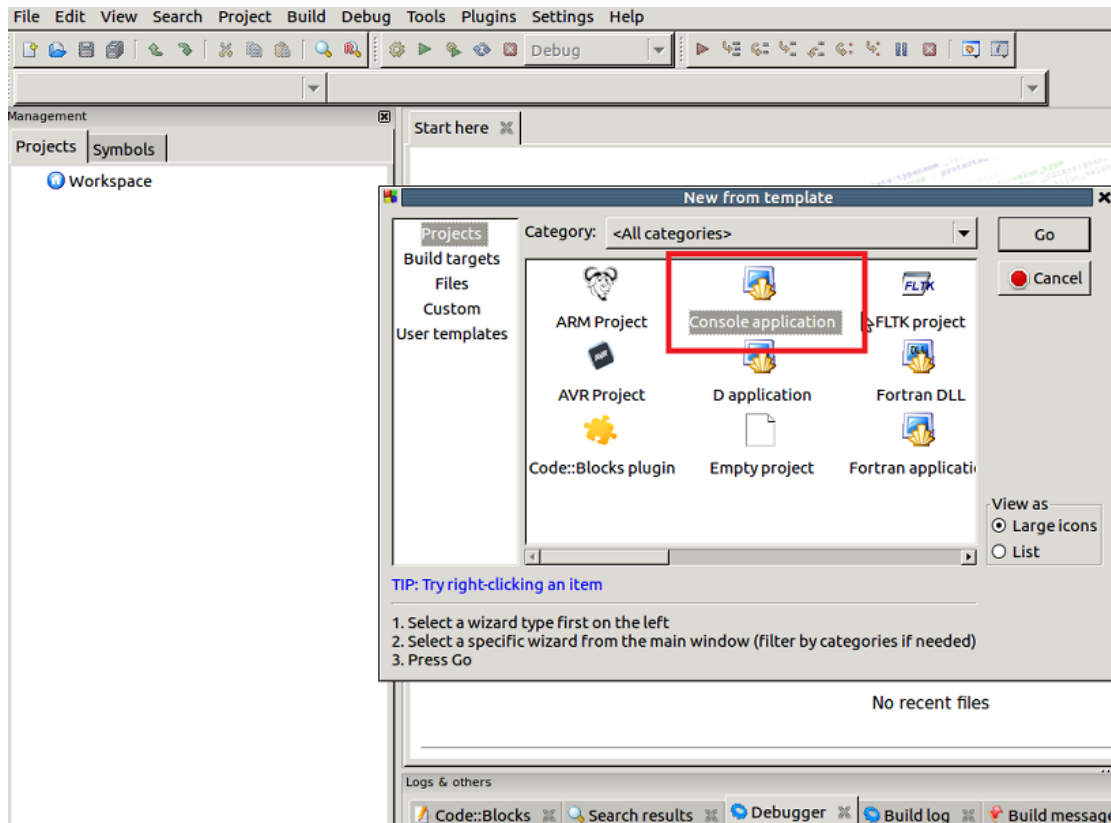


1.2 HOW TO CREATE A C++ CONSOLE PROJECT IN LINUX

For now only Linux with the GNU C++ compiler version ranging from 4.8.4 to 5 is supported.

1.2.1 FOR AMD64

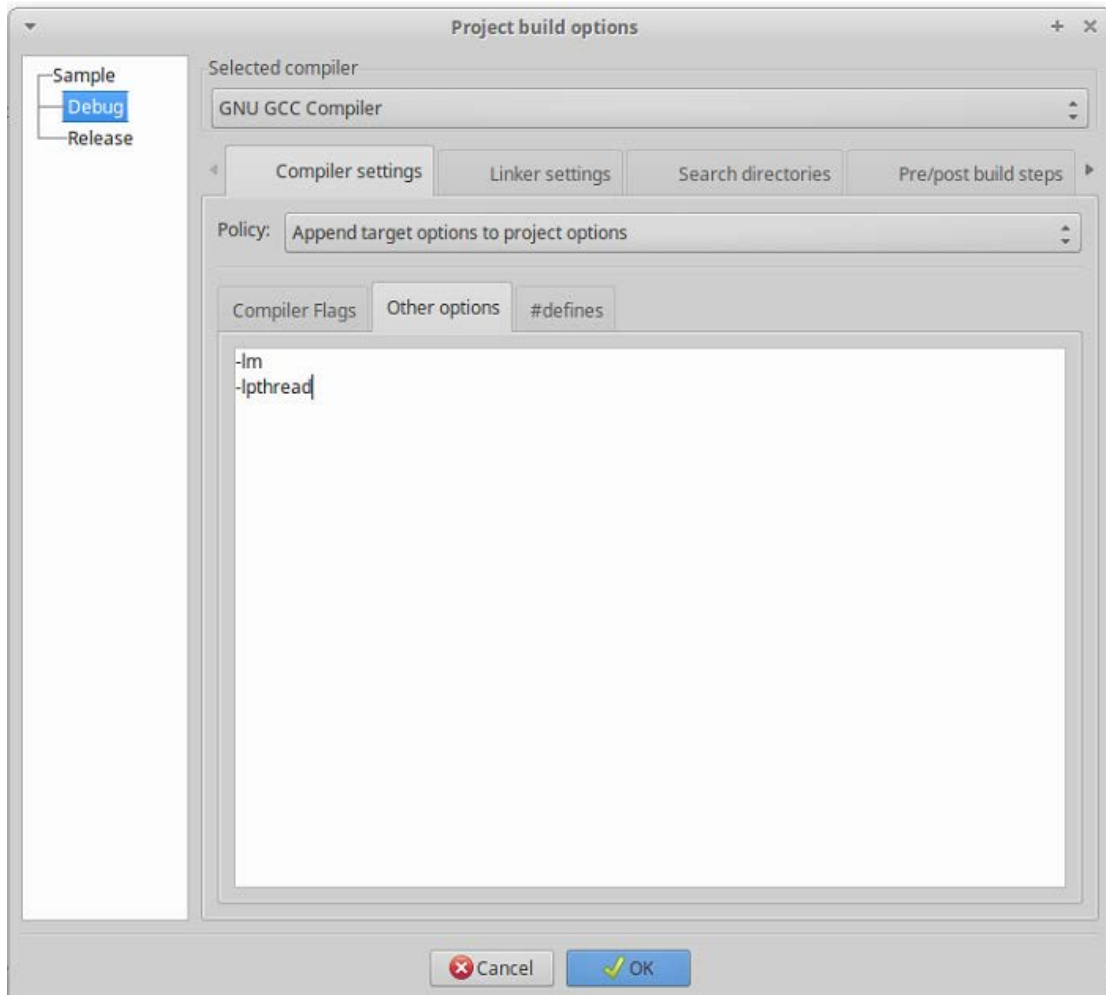
- 1) Use Code::Block IDE to create a C++ console project. On the **File** menu, point to **New**, and then click **Project**. Select **Console application**.



- 2) Copy both “include” and “lib” folders into your project directory.

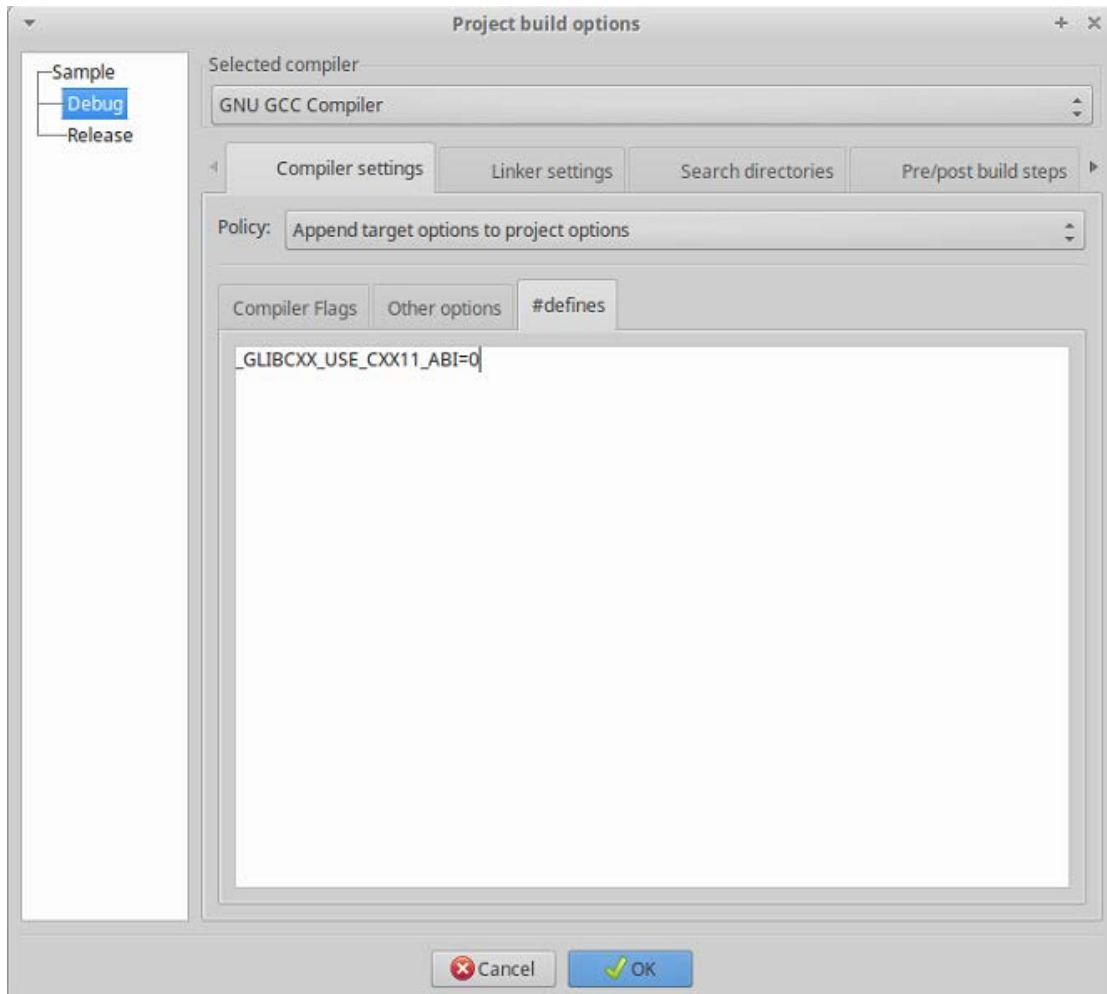
3) Click Project → Build options → Compiler settings → Other options to add compiler options:

- ▶ -lpthread
- ▶ -lm



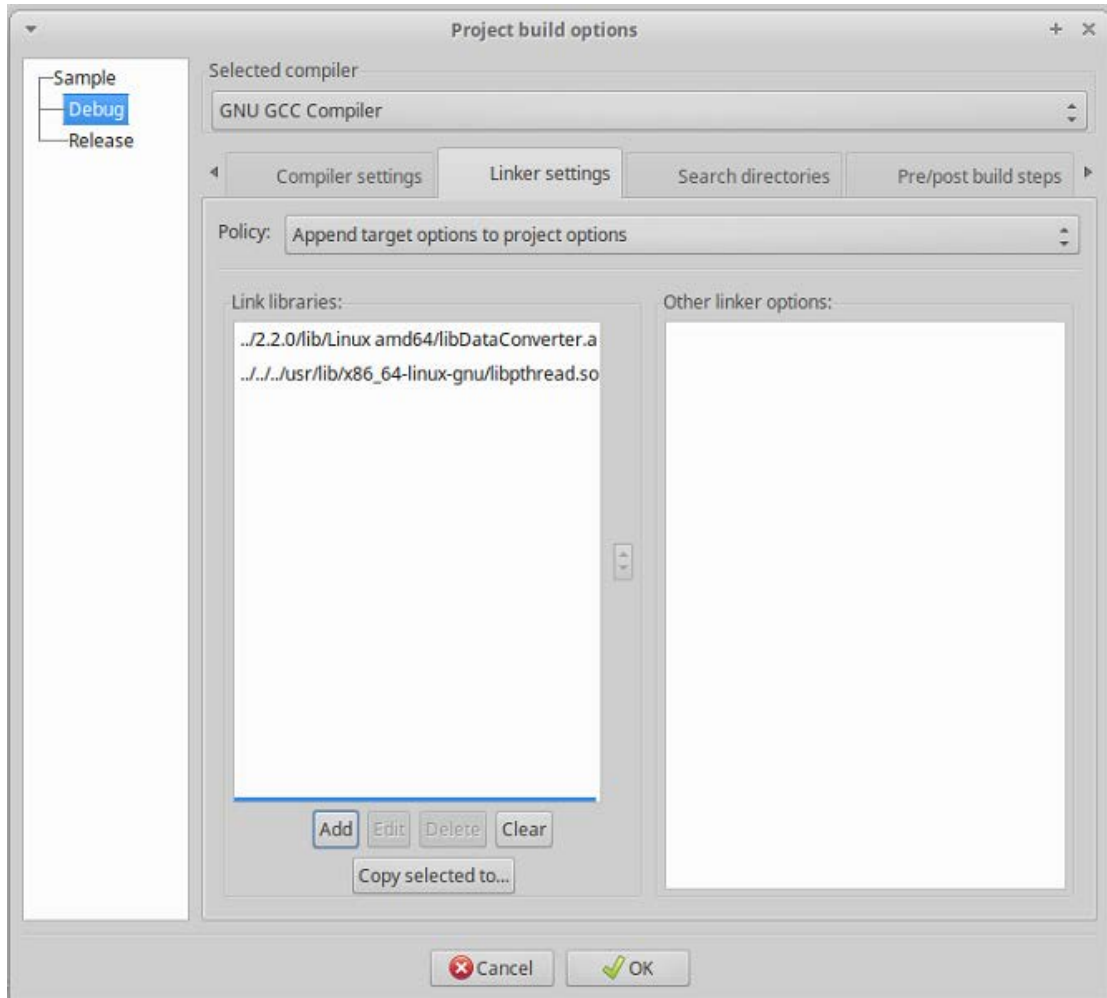
- 4) Click Project → Build options → Compiler settings → #defines to add compiler defined variables:

▶ `_GLIBCXX_USE_CXX11_ABI=0`

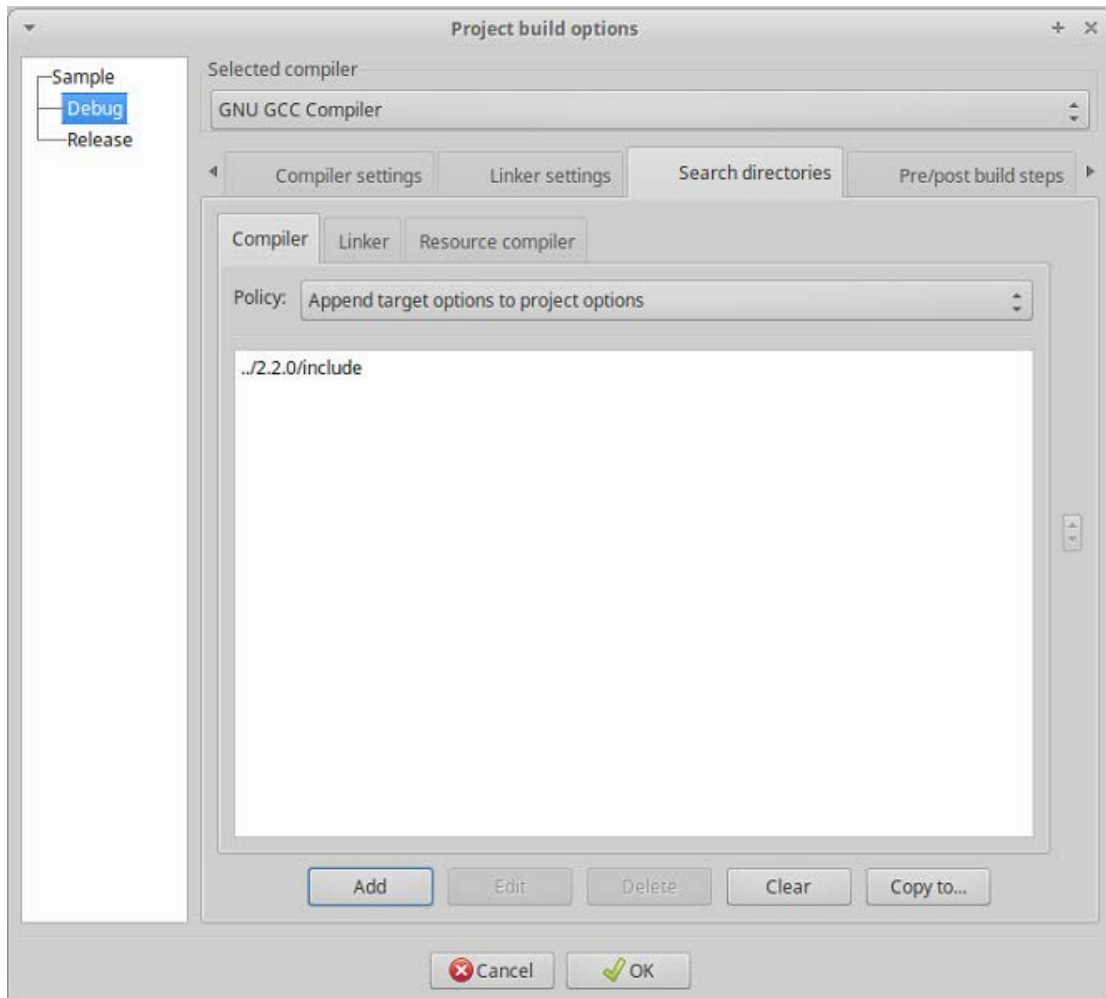


5) Click Project → Build options → Linker settings → Link libraries to add link libraries:

- ▶ Add the “Linux amd64/libDataConverter.a” which is located in the lib folder.
- ▶ Add “libpthread.so” which is located in the “/usr/lib/x86_64-linux-gnu” folder.



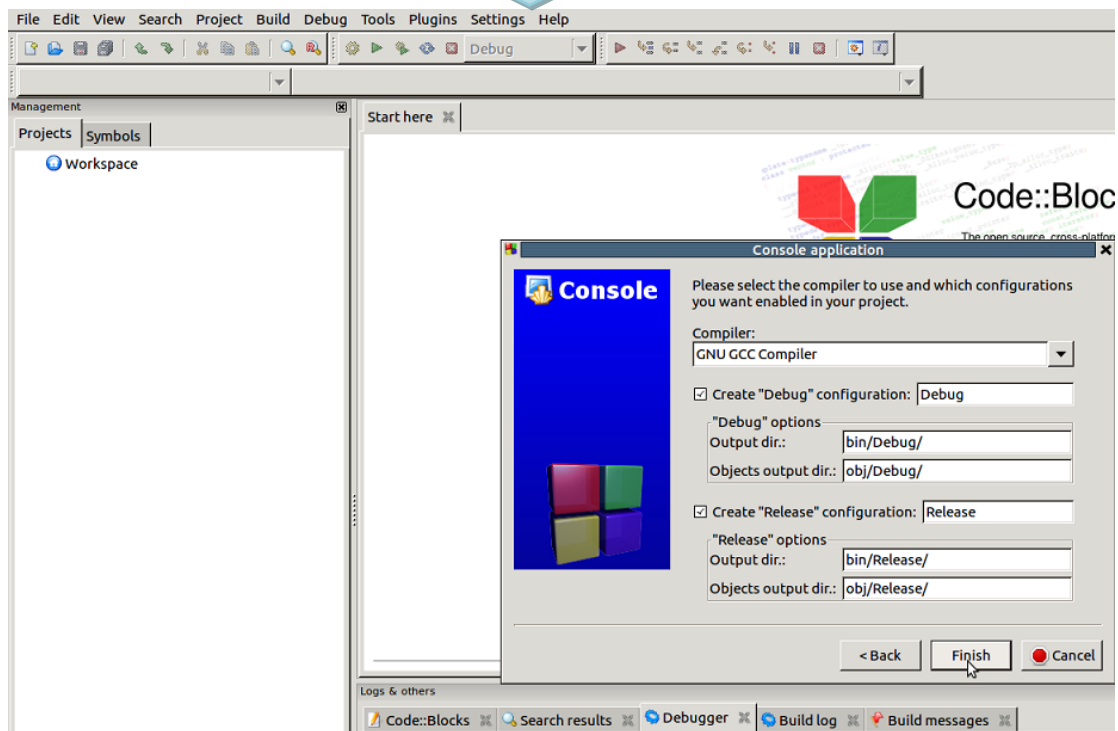
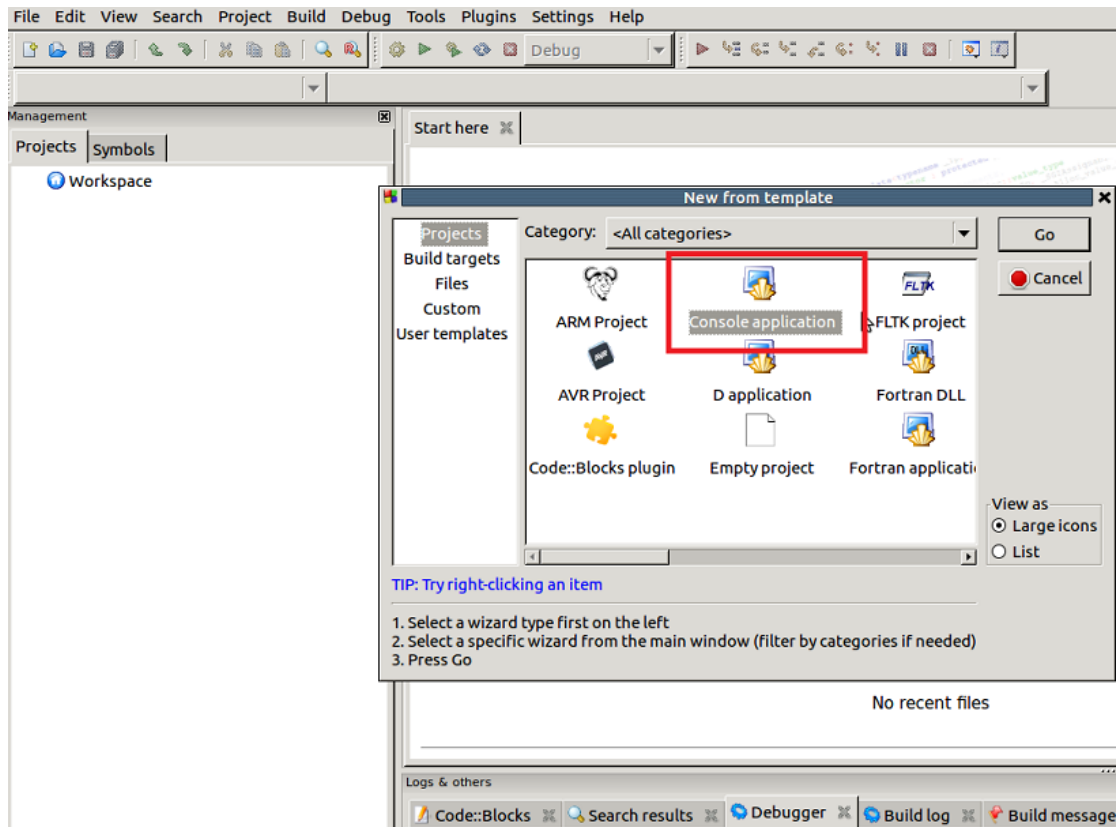
- 6) Click Project → Build options → Search Directories → Compiler to add search directories for the compiler:
- ▶ Add the “include” folder.



- 7) In the main program, you can simply include “DataConverter.h” which contains all available headers.

1.2.2 FOR I386

- 1) Use Code::Block IDE to create a C++ console project. On the **File** menu, point to **New**, and then click **Project**. Select **Console application**.

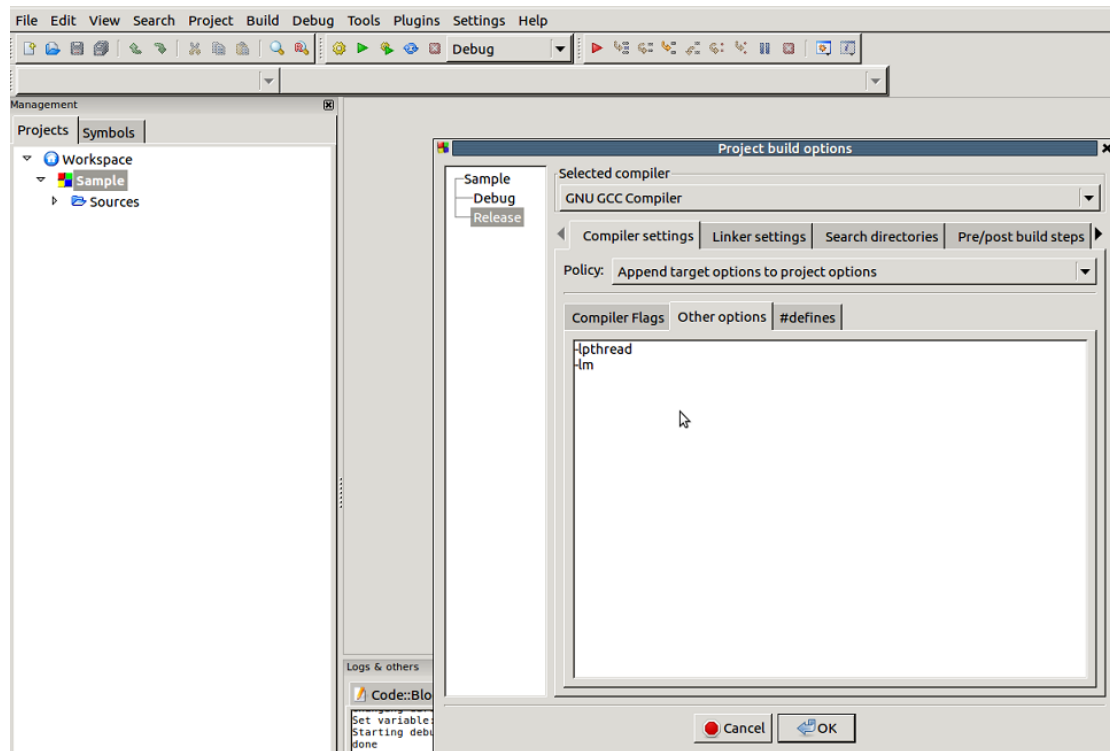


2) Copy both "*include*" and "*lib*" folders into your project directory.

- 3) Click **Project** → **Build options** → **Compiler settings** → **Other options** to add compiler options:

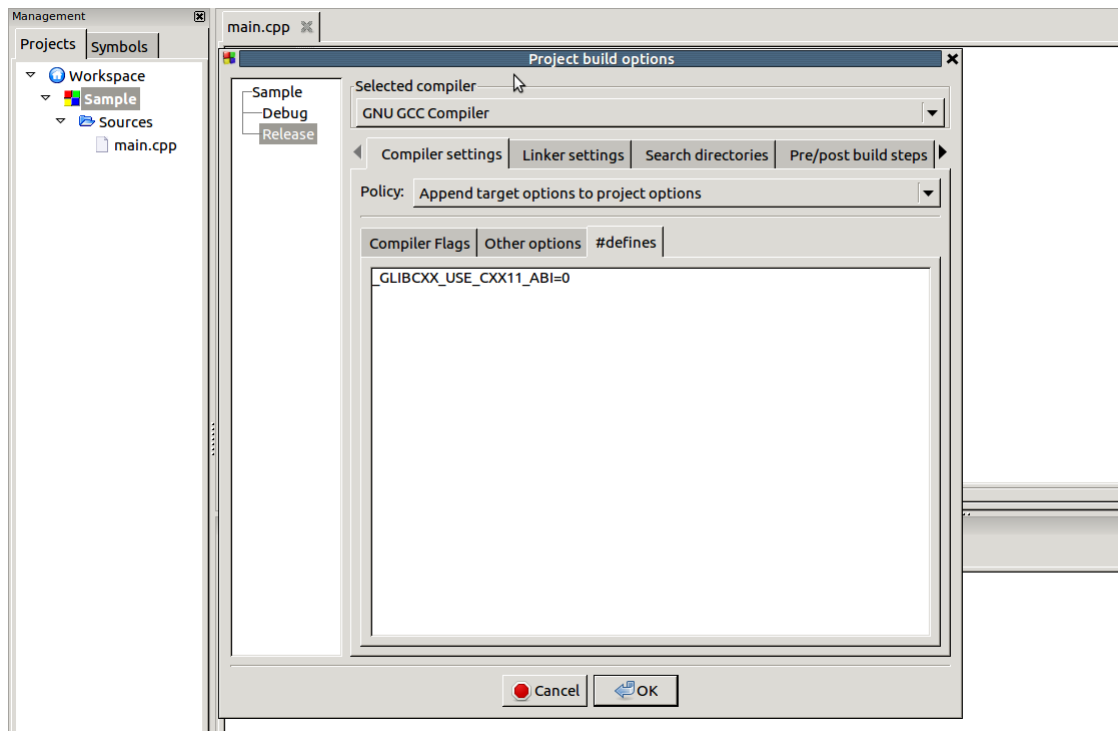
▶ `-lpthread`

▶ `-lm`



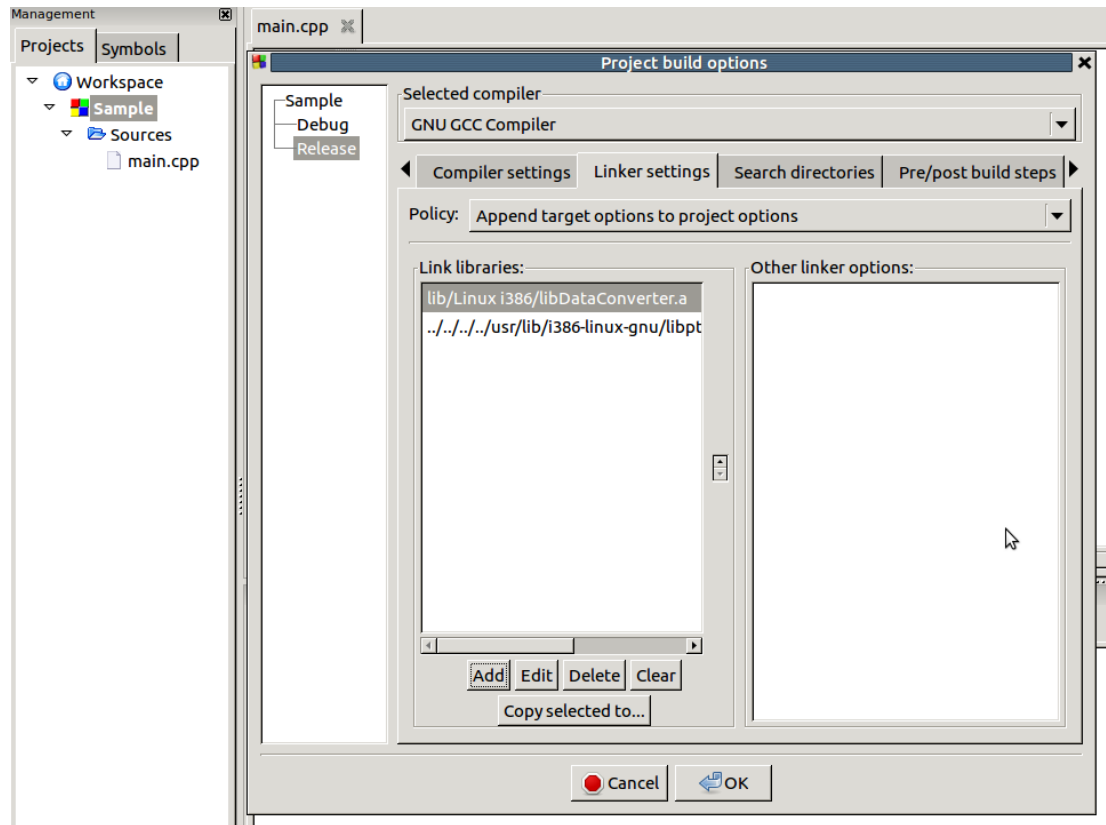
- 4) Click **Project** → **Build options** → **Compiler settings** → **#defines** to add compiler defined variables:

▶ `_GLIBCXX_USE_CXX11_ABI=0`



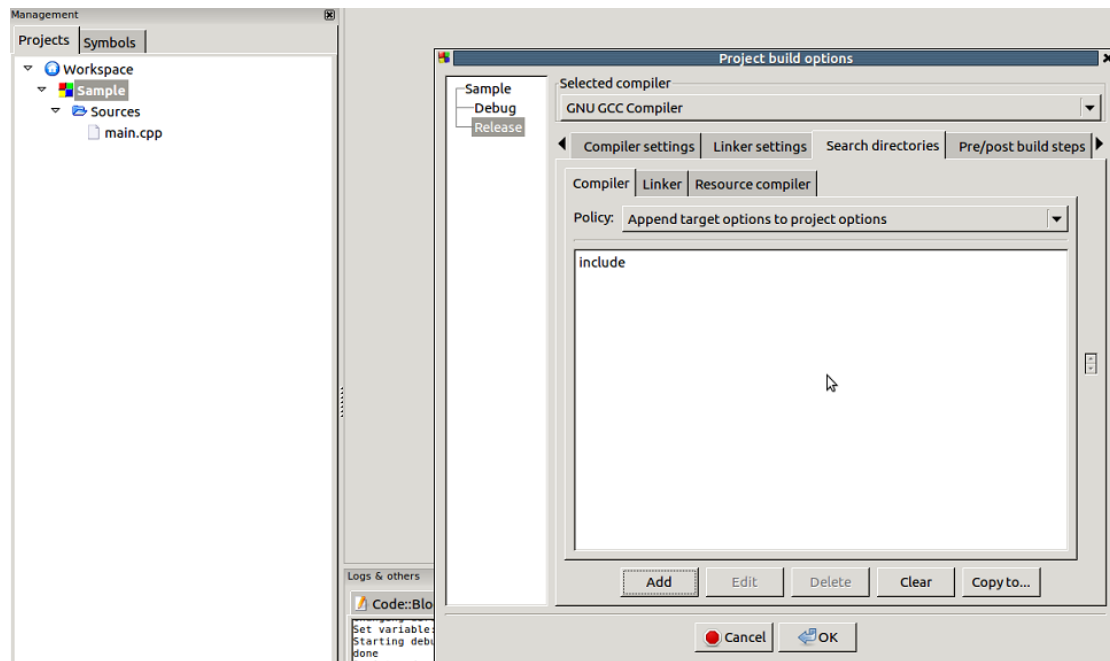
5) Click **Project** → **Build options** → **Linker settings** → **Link libraries** to add link libraries:

- ▶ Add the "*Linux i386/libDataConverter.a*" which is located in the *lib* folder.
- ▶ Add "*libpthread.so*" which is located in the "*/usr/lib/i386-linux-gnu*" folder.



- 6) Click **Project** → **Build options** → **Search Directories** → **Compiler** to add search directories for the compiler:

▶ Add the *"include"* folder.



- 7) In the main program, you can simply include *"DataConverter.h"* which contains all available headers.

1.3 SAMPLE CODE

```
#include "DataConverter.h"

#include <iostream>

#include <stdio.h>

#include <cstdlib>

#if defined _WIN32 || defined _WIN64

#else

#include <unistd.h>

#endif

using namespace std;

void progressCallBack_dbf2txt(float percent, int recordCount, const char* status)
{
    printf("Percentage: %.02f ", percent);
    printf("RecordCount: %d ", recordCount);
    printf("Status: %snn", status);
}

int main()
{
    ProgressCallBack* cbPtr = new ProgressCallBack();
    cbPtr->SetCallBack(progressCallBack_dbf2txt);

#if defined _WIN32 || defined _WIN64
    string srcC = "D://DC//testing folder//txt2dbf_c//input4.DBO";
    string tarC = "D://DC//testing folder//dbf2txt_c//input4.TXT";
#else
```



```
string srcC = "/home/DC/testing folder/txt2dbf_c/input_4f.DBO";
string tarC = "/home/DC/testing folder/dbf2txt_c/nput4.TXT";

#endif

Dbf2TxtConverter* d2tObj = new Dbf2TxtConverter(C, srcC.c_str(), tarC.c_str());

try{
    d2tObj->ConvertAsync(cbPtr);
    d2tObj->CancelAsync();
    ConversionResult* result = d2tObj->result;
    cout << "status=" << endl;
    switch(result->status)
    {
        case Failed:
            cout << "Failed" << endl;
            break;
        case Cancelled:
            cout << "Cancelled" << endl;
            break;
        #if defined _WIN32 || defined _WIN64
        case CipherLab::DataConverter::Unknown:
            #else
        case Unknown:
            #endif
            cout << "Unknown" << endl;
            break;
        case SucceededWithRecordsSkipped:
            cout << "SucceededWithRecordsSkipped" << endl;
```

```
        break;

    case Succeeded:

        cout << "Succeeded" << endl;

        break;

    default:

        cout << "Totally Unknown" << endl;

}

cout << "Output File Paths: " << endl;

for(int i = 0; i < result->getNumberOfOutputFiles(); i++)

{

    cout << i << ") " << result->getOutputFilePaths(i) << endl;

}

cout << "ErrorRecord: " << endl;

for(int i = 0; i < result->getNumberOfErrorRecords(); i++)

{

    ErrorRecord e = result->getErrorRecord(i);

    cout << i << "): ";

    cout << "filename = " << e.getFileName() << endl;

    cout << e.getRecordNum() << endl;

    cout << e.getRecordText() << endl;

}

} catch (FormatException& e)

{

    cerr << e.getMessage() << endl;

    getchar();

    exit(0);

}
```

```
#if defined _WIN32 || defined _WIN64
    cin.get();//pause console to see the message
#else
    sleep(10);
#endif

return 0;
}
```

1.4 ENUMERATIONS

1.4.1 AGXTYPE

The **AGXType** enumeration defines the type of AGX format for 8-series.

Syntax

```
typedef enum
{
    Series86;
    Series86_Encrypted;
    Others;
    Others_Encrypted;
} AGXType;
```

Members

Series86

The AGX format type for the 8600 model.

Series86_Encrypted

The AGX format type generated by Forge AG v2.0 or later for the 8600 model.

Others

The AGX format type for 8-series except for the 8600 model.

Others

The AGX format type generated by Forge AG v2.0 or later for 8-series except for the 8600 model.

1.4.2 BASICPROGDBFFILENAME

The **BasicProgDbfFileName** enumeration defines the DBF file names for Basic programming record format.

Syntax

```
typedef enum
{
    F1;
    F2;
    F3;
    F4;
    F5;
} BasicProgDbfFileName;
```

Members

F1

The file name part of the DBF file is F1.

F2

The file name part of the DBF file is F2.

F3

The file name part of the DBF file is F3.

F4

The file name part of the DBF file is F4.

F5

The file name part of the DBF file is F5.

1.4.3 CONVERSIONSTATUS

The **ConversionStatus** enumeration defines the current status of conversion process.

Syntax

```
typedef enum
{
    Failed;
    Cancelled;
    Unknown;
    SucceededWithRecordsSkipped;
    Succeeded;
} ConversionStatus;
```

Members

Failed

Conversion status is failed.

Others

Conversion status is cancelled.

Unknown

Conversion status is unknown.

SucceededWithRecordsSkipped

Conversion status is succeeded with records skipped.

Succeeded

Conversion status is succeeded.

1.4.4 FILETYPE

The **FileType** enumeration defines the type of file for doing DBF to PC conversion.

Syntax

```
typedef enum
{
    Unknown;
    Lookup;
    Transaction;
} FileType;
```

Members

Unknown

Unknown file.

Lookup

A lookup file.

Transaction

A transaction file.

1.4.5 FORGEAGPROGDBFFILENAME

The **ForgeAgProgDbfFileName** enumeration defines the type of lookup file for Forge AG programming record format.

Syntax

```
typedef enum
{
    FirstLookup;
    SecondLookup;
    ThirdLookup;
} ForgeAgProgDbfFileName;
```

Members

FirstLookup

The first lookup file.

SecondLookup

The second lookup file.

ThirdLookup

The third lookup file.

1.4.6 PROGRAMMINGTOOL

The **ProgrammingTool** enumeration defines the programming tool employed by users to program the application.

Syntax

```
typedef enum
{
    C;
    Basic;
    ForgeAg;
} ProgrammingTool;
```

Members

C

C programming tool.

Basic

Basic programming tool.

ForgeAg

Forge Ag programming tool.

1.4.7 WRONGFORMATACTION

The **WrongFormatAction** enumeration defines the actions to take when encountering wrong data format during conversion.

Syntax

```
typedef enum
```

```
{  
    Stop;  
    Reformat;  
    Skip;  
} WrongFormatAction;
```

Members

Stop

Stop the conversion process when data format is not correct.

Reformat

Reformat the data record when the format is not correct.

Skip

Skip the current record and continue with the next record.

1.5 CLASS

1.5.1 AGX

Reads data format from an AGX file.

Header File: #include <AGX.h>

Static Public Member Function:

```
DLLEXPORT static int DetermineFormatType (const char *path)
```

To determine the format type of the AGX file.

```
static DLLEXPORT ForgeAgProgDataFormat *Read (const *path, int &counter)
```

Reads record format details from the AGX file.

DETERMINEFORMATTYPE

Syntax

```
int CipherLab::DataConverter::AGX::DetermineFormatType (const char *path) [static]
```

Parameters

path

The AGX file path.

Return Value

AGXType with success; or -1 indicating unable to open the file.

Example

```
string path2 = "D:\\DC\\testing folder\\agx\\8600_Basic_Test1.AGX";
int val2 = AGX::DetermineFormatType(path2.c_str());
cout << "fileType = " ;
switch(val2)
{
case CipherLab::DataConverter::Series86:
    cout << "Series86" << endl;
    break;
case CipherLab::DataConverter::Series86_Encrypted:
    cout << "Series86_Encrypted" << endl;
    break;
```



```
case CipherLab::DataConverter::Others:
    cout << "Others" << endl;
    break;
case CipherLab::DataConverter::Others_Encrypted:
    cout << "Others_Encrypte" << endl;
    break;
default:
    cout << "Cannot open file" << endl;
}
```

READ

Syntax

```
ForgeAgProgDataFormat *CipherLab::DataConverter::AGX::Read (const char *path, int
&counter) [static]
```

Parameters

path

The AGX file path.

counter

The number of records that reference the number of records of ForgeAgProgDataFormat.

Return Value

ForgeAgProgDataFormat.

Example

```
string path = "D:\\DC\\testing folder\\agx\\82_susan_fixed.AGX";
int numOfLookup = 0;
ForgeAgProgDataFormat *for86df = AGX::Read(path2.c_str(), numOfLookup);

for(int i = 0; i < numOfLookup; i++)
{
    cout << "[" << i << "] DbfFileName: " << for86df[i].DbfFileName << endl;
    cout << for86df[i].actionTakenWhenFormatIsWrong << endl;
    for(int j = 0; j < for86df[i].getNumberOfRecordFields(); j++)
    {
```

```

RecordField rec = for86df[i].GetRecordFieldAt(j);
cout << "[" << j << "]" Length(" << rec.Length << ") Offset(" << rec.Position
<< ") Key(" << rec.IsKey << ")" << endl;
}
}

```

1.5.2 APIVERSION

Retrieves version information for the Converter C++ API.

Header File: #include <APIVersion.h>

Public Member Function:

```
DLLEXPORT ApiVersion ()
```

Constructor of the ApiVersion class.

```
DLLEXPORT ~ApiVersion()
```

Destructor of the ApiVersion class.

```
DLLEXPORT int GetApi_build_version ()
```

Gets API build number.

```
DLLEXPORT int GetApi_major_version ()
```

Gets API major number.

```
DLLEXPORT int GetApi_minor_version ()
```

Gets API minor number.

```
DLLEXPORT const char *GetApiVersion()
```

Gets API version info in const char*.

GETAPI_BUILD_VERSION

Syntax

```
int CipherLab::DataConverter::ApiVersion::GetApi_build_version ()
```

Return Value

int API build number

GETAPI_MAJOR_VERSION

Syntax

```
int CipherLab::DataConverter::ApiVersion::GetApi_major_version ()
```

Return Value

int API major number

GETAPI_MINOR_VERSION

Syntax

```
int CipherLab::DataConverter::ApiVersion::GetApi_minor_version ()
```

Return Value

int API minor number

GETAPIVERSION

Syntax

```
int CipherLab::DataConverter::ApiVersion::GetApiVersion ()
```

Return Value

const char* API version

1.5.3 BASICPROGDATAFORMAT

Inherits CipherLab::DataConverter::DataFormat

Public Member Function:

```
DLLEXPORT BasicProgDataFormat ()
```

Default constructor of the BasicProgDataFormat class.

```
DLLEXPORT BasicProgDataFormat (const char *path, BasicProgDbfFileName  
dbfFileName)
```

Constructor of the BasicProgDataFormat class.

```
DLLEXPORT BasicProgDataFormat (const char *path, BasicProgDbfFileName  
dbfFileName, char delimiter)
```

Constructor of the BasicProgDataFormat class.

```
DLLEXPORT ~BasicProgDataFormat ()
```

Destructor of the BasicProgDataFormat class.

Public Attribute:

BasicProgDbfFileName

The output DBF file name of BasicProgDataFormat.

Additional Inherited Member:**Syntax**

CipherLab::DataConverter::BasicProgDataFormat::BasicProgDataFormat (const char *path, BasicProgDbfFileName dbfFileName)

Constructor of the BasicDataFormat class.

Parameters

path

Specifies the file path.

dbfFileName

The output DBF file name.

Syntax

CipherLab::DataConverter::BasicProgDataFormat::BasicProgDataFormat (const char *path, BasicProgDbfFileName dbfFileName, char delimiter)

Constructor of the BasicDataFormat class.

Parameters

path

The file path.

dbfFileName

The output DBF file name.

delimiter

The delimiter character used in the record to separate data.

Remarks

- ▶ The maximum number of record fields for both non-8600 models and 8600 model is 8.
- ▶ Txt2PackedDbf conversion:
For non-8600 model, the format can have 1~3 key fields specified; as for 8600 model, the format can have 0~5 key fields specified.
- ▶ Txt2Dbf conversion:

For non-8600 model, the format can have 0~3 key fields specified; as for 8600 model, the format can have 0~5 key fields specified.

1.5.4 CONVERSIONRESULT

Gets the post conversion result including the possible ErrorRecord list and the output file paths.

Header File: #include <ConversionResult.h>

Public Member Function:

ConversionResult ()

Constructor of the ConversionResult class.

~ConversionResult ()

Destructor of the ConversionResult class.

ErrorRecord getErrorRecord (int index)
--

Returns the error record with specific index.

int getNumberOfErrorRecords ()

Returns the number of error records.

int getNumberOfOutputFiles ()

Returns the number of output files.

const char *GetOutputFilePaths(int index)

Returns the output file's path with specific index.

Public Attribute:

ConversionStatus status

Static Public Attribute:

static std::vector<ErrorRecord>errorRecordList
--

static std::vector<string>outputFilePaths

1.5.5 CPROGDATAFORMAT

Inherits CipherLab::DataConverter::DataFormat.

Public Member Function:

DLLEXPORT CProgDataFormat ()

Constructor of the CProgDataFormat class.

```
DLLEXPORT CProgDataFormat (const char *path, const char *dbfFileName)
```

Constructor of the CProgDataFormat class.

```
DLLEXPORT CProgDataFormat (const char *path, const char *dbfFileName, char  
delimiter)
```

Constructor of the CProgDataFormat class.

```
DLLEXPORT ~CProgDataFormat ()
```

Destructor of the CProgDataFormat class.

Public Attribute:

```
const char *DbfFileName
```

The output DBF file name of CProgDataFormat.

Additional Inherited Member:

Syntax

```
CipherLab::DataConverter::CProgDataFormat::CProgDataFormat (const char *path,  
const char *dbfFileName)
```

Constructor of the CDataFormat class.

Parameters

path

The input file path.

dbfFileName

The output DBF file name.

Syntax

```
CipherLab::DataConverter::CProgDataFormat::CProgDataFormat (const char *path,  
const char *dbfFileName, char delimiter)
```

Constructor of the CDataFormat class.

Parameters

path

The input file path.

dbfFileName

The output DBF file name.

delimiter

The delimiter character used in the record to separate data.

Remarks

- ▶ The maximum number of record fields for both non-8600 models and 8600 model is 8.
- ▶ Txt2PackedDbf conversion:
For non-8600 model, the format can have 1~8 key fields specified; as for 8600 model, the format can have 0~8 key fields specified.
- ▶ Txt2Dbf conversion:
For both non-8600 models and 8600 model, the format can have 0~8 key fields specified.

1.5.6 DATAFORMAT

Inherited by CipherLab::DataConverter::BasicProgDataFormat, CipherLab::DataConverter::CProgDataFormat, and CipherLab::DataConverter::ForgeAgProgDataFormat.

Public Member Function:

DataFormat (int maxFields, char delimiter=0x00)

Constructor of the DataFormat class.

~DataFormat ()

Destructor of the DataFormat class.

void AddRecordField (RecordField recordField)

Adds a recordField to the DataFormat class.

void ClearRecordFields ()

Clear the record fields in a data format.

void EditRecordFieldIsKey (int index, bool isKey)

Edit a specific record field whether it's a key field or not.

void EditRecordFieldLength (int index, int length)
--

Edit a specific record field's length.

void EditRecordFieldPosition (int index, int position)
--

Edit a specific record field's position.

const char *getDbfFileName ()

Get the output DBF file name.

```
char getDelimiter ()
```

Get the delimiter character.

```
int getKeyCount ()
```

Get the number of key fields in the data format.

```
int getMaxRecordFields ()
```

Get the maximum record fields in the data format.

```
int getNumberOfRecordFields ()
```

Get the number of record fields in the data format.

```
const char *getPath ()
```

Get the input path.

```
RecordField GetRecordFieldAt (int index)
```

Gets a recordfield from the specified location.

```
bool isRecordDelimited ()
```

Check whether the record data is delimited.

```
void ParseRecordFields (const char *inputFilePath)
```

Parse the input file to generate the vector of record fields.

```
void setDbfFileName (const char *fileName)
```

Set the output DBF file name.

```
void setDelimiter (char ch)
```

Set the delimiter character.

```
void setPath (const char *path)
```

Set the input path.

Protected Attribute:

```
const char *dbfFileName  
char delimiter  
int maxFieldLength  
int maxFieldPosition  
int maxRecordFields
```



```
int minFieldLength  
int minFieldPosition  
string path  
vector< RecordField > RecordFields
```

CONSTRUCTOR & DESTRUCTOR

Syntax

```
CipherLab::DataConverter::DataFormat::DataFormat (int maxFields, char delimiter =  
0x00)
```

Constructor of the DataFormat class.

Parameters

maxFields

The maximum number of fields in the data format.

delimiter

The delimiter character used in the record to separate data.

DETAILS OF MEMBER FUNCTION

Syntax

```
void CipherLab::DataConverter::DataFormat::AddRecordField (RecordField recordField)
```

Adds a RecordField object to the DataFormat class.

Parameters

recordField

A RecordField object.

Syntax

```
void CipherLab::DataConverter::DataFormat::EditRecordFieldIsKey (int index, bool  
isKey)
```

Edits a specific record field whether it's a key field or not.

Parameters

index

The index of the record field.

isKey

Changes the specific record field whether it's a key field or not.

Syntax

```
void CipherLab::DataConverter::DataFormat::EditRecordFieldLength (int index, int length)
```

Edits a specific record field's length.

Parameters

index

The index of the record field.

length

The new length of the record field.

Syntax

```
void CipherLab::DataConverter::DataFormat::EditRecordFieldPosition (int index, int position)
```

Edits a specific record field's position.

Parameters

index

The index of the record field.

position

The new position of the record field.

Syntax

```
int CipherLab::DataConverter::DataFormat::getKeyCount ()
```

Gets the number of key fields in the data format.

Return Value

The number of key fields in the data format.

Syntax

```
int CipherLab::DataConverter::DataFormat::getMaxRecordFields ()
```

Gets the maximum record fields in the data format.

Return Value

The maximum record fields in the data format.

Syntax

```
int CipherLab::DataConverter::DataFormat::getNumberOfRecordFields ()
```

Gets the number of record fields in the data format.

Return Value

The number of record fields in the data format.

Syntax

```
RecordField CipherLab::DataConverter::DataFormat::GetRecordFieldAt (int index)
```

Gets a record field from the specified location.

Parameters

index

The index of the record field.

Syntax

```
void CipherLab::DataConverter::DataFormat::ParseRecordFields ( const char  
*inputFilePath )
```

Parses the input file to generate the vector of record fields.

Return Value

The vector of record fields.

Syntax

```
void CipherLab::DataConverter::DataFormat::setDbfFileName ( const char *fileName )
```

Sets the output DBF file name.

Parameters

fileName

The output DBF file name.

1.5.7 DBF2TXTCONVERTER

Converts DBF files or transaction (DAT) files on SD card to text files.

Header File: #include <Dbf2TxtConverter.h>

Public Member Function:

```
DLLEXPORT Dbf2TxtConverter (ProgrammingTool tool, const char *source, const char *target)
```

Initializes a new instance of the Dbf2TxtConverter class with programming data format.

```
DLLEXPORT void CancelAsync ()
```

Requests cancellation of asynchronous conversion.

```
DLLEXPORT void Convert (ProgressCallBack *cbPtr)
```

Requests conversion.

```
DLLEXPORT void ConvertAsync (ProgressCallBack *cb)
```

Requests action of asynchronous conversion.

Public Attribute:

```
ConversionResult *result
```

Points to the ConversionResult object for conversion result.

DETAILS OF MEMBER FUNCTION**Syntax**

```
void CipherLab::DataConverter::Dbf2TxtConverter::ConvertAsync (ProgressCallBack *cb)
```

Requests action of asynchronous conversion.

Parameters

cb

A function pointing to the user-defined progress report function. If there is no need to report progress, set *cb* to NULL.

Example

```
#include "DataConverter.h"
#include <iostream> //for cin, cout and endl
#include <string.h>
#include <stdio.h>
#include <cstdlib>

using namespace std;
```

```
using namespace CipherLab::DataConverter;

void progressCallBack(float percent, int recordCount, const char* status)
{
    printf("Percentage: %.02f ", percent);
    printf("RecordCount: %d ", recordCount);
    printf("Status: %s\n", status);
}

int main()
{
    ProgressCallBack* cbPtr = new ProgressCallBack();
    cbPtr->SetCallBack(progressCallBack);

    string srcC = "D:\\DC\\testing folder\\txt2dbf_c\\input4.DBO";
    string tarC = "D:\\DC\\testing folder\\dbf2txt_c\\input4.TXT";

    Dbf2TxtConverter* d2tObj = new Dbf2TxtConverter(C, srcC.c_str(), tarC.c_str());
    try{
        d2tObj->ConvertAsync(cbPtr);
        d2tObj->CancelAsync();
    }catch (FormatException& e)
    {
        cerr << e.getMessage() << endl;
        getchar();
        exit(0);
    }
    cin.get();//pause console to see the message
    return 0;
}
```

1.5.8 ERRORRECORD

Records errors during processing data conversion.

Header File: #include <ErrorRecord.h>

Public Member Function:

ErrorRecord (string fileName, long recordNum, string recordText, bool isSkip)

Constructor of ErrorRecord.

ErrorRecord ()

Default Constructor of ErrorRecord.

~ErrorRecord ()

Destructor of ErrorRecord.

const char *getFileName ()

Gets the input file name.

long getRecordNum ()

Gets the sequence of the record.

const char *getRecordText ()

Gets the text of record.

bool isSkipped ()

Determines whether the ErrorRecord is skipped.

CONSTRUCTOR & DESTRUCTOR

Syntax

CipherLab::DataConverter::ErrorRecord::ErrorRecord (string fileName, long recordNum, string recordText, bool isSkip)

Constructor of ErrorRecord.

Parameters

fileName

The input file name.

recordNum

Identifies the sequence of the record.

recordText

Identifies the text of record.

isSkip

Identifies whether the action to take when detecting wrong data format during conversion is 'skip' or not.

Return Value

Object which contains the error info when detecting wrong data format during conversion.

1.5.9 FORGEAGPROGDATAFORMAT

Inherits CipherLab::DataConverter::DataFormat.

Public Member Function:

DLLEXPORT ForgeAgProgDataFormat ()

Default Constructor of ForgeAgProgDataFormat class.

DLLEXPORT ForgeAgProgDataFormat (const char *path, ForgeAgProgDbfFileName dbfFileName)
--

Constructor of ForgeAgProgDataFormat class.

DLLEXPORT ForgeAgProgDataFormat (const char *path, ForgeAgProgDbfFileName dbfFileName, char delimiter)
--

Constructor of ForgeAgProgDataFormat class.

DLLEXPORT ~ForgeAgProgDataFormat ()

Destructor of ForgeAgProgDataFormat class.

DLLEXPORT void AddRecordField (RecordField recordField)

Adds a record field to the data format.

DLLEXPORT int getKeyIndex ()

Gets the index of key field.

DLLEXPORT RecordField getKeyRecordField ()
--

Gets the key record field.

Public Attribute:

WrongFormatAction actionTakenWhenFormatIsWrong
--

The action to take when detecting wrong data format during conversion.

ForgeAgProgDbfFileName DbfFileName

The output DBF file name of ForgeAgProgDataFormat.

Additional Inherited Member:

Syntax

```
CipherLab::DataConverter::ForgeAgProgDataFormat::ForgeAgProgDataFormat (const
char *path, ForgeAgProgDbfFileName dbfFileName)
```

Constructor of the ForgeAgProgDataFormat class.

Parameters

path

The input file path.

dbfFileName

The output DBF file name.

```
CipherLab::DataConverter::ForgeAgProgDataFormat::ForgeAgProgDataFormat (const
char *path, ForgeAgProgDbfFileName dbfFileName, char delimiter)
```

Constructor of the ForgeAgProgDataFormat class.

Parameters

path

The input file path.

dbfFileName

The output DBF file name.

delimiter

The delimiter character used in the record to separate data.

Remarks

- ▶ The maximum number of record fields for non-8600 models is 8; as for 8600, the maximum number of record fields is 12.
- ▶ For both Txt2PackedDbf conversion and Txt2Dbf conversions, the format must have one and only one key field specified.

1.5.10 FORMATEXCEPTION

Inherits exception.

Header File: #include <FormatException.h>

Public Member Function:

```
FormatException (void)
```

Default constructor of the formatException class.


```
FormatException (string message, string info="")
```

Constructor of the FormatException class.

```
~FormatException (void) throw ()
```

Destructor of the FormatException class.

```
string GetMessage ()
```

Return the runtime exception message.

1.5.11 RECORDFIELD

The RecordField class.

Header File: #include <RecordField.h>

Public Member Function:

```
RecordField (int position, int length, bool isKey)
```

RecordField constructor takes 3 parameters.

```
RecordField (int position, int length)
```

RecordField constructor takes 2 parameters.

```
RecordField ()
```

Default constructor of RecordField class.

Public Attribute:

```
bool IsKey
```

A public member boolean variable.

```
int Length
```

A public member integer variable.

```
int Position
```

A public member integer variable.

CONSTRUCTOR & DESTRUCTOR

Syntax

```
CipherLab::DataConverter::RecordField::RecordField (int position, int length, bool isKey)
```

RecordField constructor takes 3 parameters.

Parameters*position*

The start position of the record field in specific line content.

length

The length of the record field.

isKey

To determine whether the record field is a key field.

Syntax

```
CipherLab::DataConverter::RecordField::RecordField (int position, int length)
```

RecordField constructor takes 2 parameters.

Parameters*position*

The start position of the record field in specific line content.

length

The length of the record field.

DETAILS OF MEMBER DATA**Syntax**

```
bool CipherLab::DataConverter::RecordField::IsKey
```

A public member boolean variable (no matter whether the record field is a key field or not).

```
int CipherLab::DataConverter::RecordField::Length
```

A public member integer variable (the length of the record field).

```
int CipherLab::DataConverter::RecordField::Position
```

A public member integer variable (the start position of the record field).

1.5.12 PROGRESSCALLBACK

Have the converting function acting as a callback function to report the percentage, record count, and status of conversion process.

Syntax

CipherLab::DataConverter::ProgressCallBack (void)

Default constructor of ProgressCallBack

void CipherLab::DataConverter::SetCallBack (CBFun fun)

Register *fn* as a callback function to be called automatically for conversion.

Parameters

fun

Pointer to the function to be called. The CBFun member type is defined as:

```
typedef void (*CBFun) (float percentage, int recordCount, const char* status);
```

percentage

the current percentage of process.

recordCount

the current processed record count.

status

the processing status.

1.5.13 TXT2DBFCONVERTER

Public Member Function:

DLLEXPORT Txt2DbfConverter (BasicProgDataFormat &dataFormat, const char *targetDir, WrongFormatAction actionTakenWhenWrong)

Initializes a new instance of the Txt2DbfConverter class in basic programming data format.

DLLEXPORT Txt2DbfConverter (CProgDataFormat &dataFormat, const char *targetDir, WrongFormat Action actionTakenWhenWrong)

Initializes a new instance of the Txt2DbfConverter class in C programming data format.

DLLEXPORT Txt2DbfConverter (ForgeAgProgDataFormat &dataFormat, const char *targetDir, WrongFormatAction actionTakenWhenWrong)

Initializes a new instance of the Txt2DbfConverter class in Forge AG programming data format.

DLLEXPORT ~Txt2DbfConverter ()

Destructor of the Txt2DbfConverter class.

```
DLLEXPORT void CancelAsync ()
```

Requests cancellation of asynchronous conversion.

```
DLLEXPORT ConversionResult Convert (ProgressCallBack *cb)
```

Requests conversion.

```
DLLEXPORT void ConvertAsync (ProgressCallBack *cb)
```

Requests action of asynchronous conversion.

Public Attribute:

```
ConversionResult result
```

The ConversionResult object for conversion result.

```
ConversionStatus status
```

The ConversionStatus object for conversion result.

CONSTRUCTOR & DESTRUCTOR

Syntax

```
CipherLab::DataConverter::Txt2DbfConverter::Txt2DbfConverter (BasicProgDataFormat & dataFormat, const char *targetDir, WrongFormatAction actionTakenWhenWrong)
```

Initializes a new instance of the Txt2DbfConverter class in basic programming data format.

Parameters

dataFormat

Reference to a BasicProgDataFormat object.

targetDir

The path for the output file(s).

actionTakenWhenWrong

The action to take when detecting wrong data format during conversion.

Syntax

```
CipherLab::DataConverter::Txt2DbfConverter::Txt2DbfConverter (CProgDataFormat & dataFormat, const char *targetDir, WrongFormatAction actionTakenWhenWrong)
```

Initializes a new instance of the Txt2DbfConverter class in C programming data format.

Parameters

dataFormat

Reference to a CProgDataFormat object.

targetDir

The path for the output file(s).

actionTakenWhenWrong

The action to take when detecting wrong data format during conversion.

Syntax

```
CipherLab::DataConverter::Txt2DbfConverter::Txt2DbfConverter  
(ForgeAgProgDataFormat & dataFormat, const char *targetDir, WrongFormatAction  
actionTakenWhenWrong)
```

Initializes a new instance of the Txt2DbfConverter class in Forge AG programming data format.

Parameters

dataFormat

Reference to a ForgeAgProgDataFormat object.

targetDir

The path for the output file(s).

actionTakenWhenWrong

The action to take when detecting wrong data format during conversion.

DETAILS OF MEMBER FUNCTION

Syntax

```
ConversionResult CipherLab::DataConverter::Txt2DbfConverter::Convert  
(ProgressCallBack *cb)
```

Requests conversion.

Parameters

cb

A function pointing to the user-defined progress report function. If there is no need to report progress, set cb to NULL.

Return Value

Points to the ConversionResult object for conversion result.

Example

```
BasicProgDataFormat bpdf;
bpdf.setDelimiter(0x2C);
bpdf.DbfFileName = CipherLab::DataConverter::F2;
string path = "D:\\DcTestFolder\\testFiles\\D\\input_delimiter_15.txt";
bpdf.setPath(path.c_str());
bpdf.ParseRecordFields(bpdf.getPath());
bpdf.EditRecordFieldIsKey(1, true);
bpdf.EditRecordFieldLength(2, 3);
bpdf.EditRecordFieldIsKey(2, true);

Txt2DbfConverter* t2dObj = new Txt2DbfConverter(bpdf,
"D:\\DcTestFolder\\PC2SD\\Delimiter\\API", Skip);
ConversionResult result;
try{
    result = t2dObj->Convert(NULL);
}
}
catch(formatException& e)
{
    cerr << e.getMessage() << endl;
    getchar();
    exit(0);
}
```

Syntax

```
void CipherLab::DataConverter::Txt2DbfConverter::ConvertAsync (ProgressCallBack
*cb)
```

Requests action of asynchronous conversion.

Parameters

cb

A function pointing to the user-defined progress report function. If there is no need to report progress, set cb to NULL.

Return Value

Points to the ConversionResult object for conversion result.

Syntax

`void CipherLab::DataConverter::Txt2DbfConverter::CancelAsync ()`

Requests cancellation of asynchronous conversion.

Example

```
CProgDataFormat cpdf;
cpdf.setDelimiter(0x2C);
cpdf.AddRecordField(RecordField(1,4, false));
cpdf.AddRecordField(RecordField(6,2, true));
cpdf.AddRecordField(RecordField(9,3, false));

cpdf.setDelimiter(0x2C);
cpdf.DbffFileName = "input_c";
string path = "D:\\DcTestFolder\\testFiles\\D\\input_delimiter_15.txt";
cpdf.setPath(path.c_str());

for(int i = 0; i < cpdf.getNumberOfRecordFields(); i++)
{
    RecordField a = cpdf.GetRecordFieldAt(i);
    cout << "recordVector[" << i << "]" << " offset(" << a.Position << ") length(" <<
    a.Length << ")" << endl;
}

Txt2DbfConverter* t2dObj = new Txt2DbfConverter(cpdf,
"D:\\DcTestFolder\\PC2SD\\Delimiter\\API", Skip);
try{
    ConversionResult result;
    ProgressCallBack* cbPtr = new ProgressCallBack();
    cbPtr->SetCallBack(progressCallBack);
    t2dObj->ConvertAsync(cbPtr);
    result = t2dObj->result;
    switch(result.status)
```

```
{
case CipherLab::DataConverter::Failed:
    cout << "Failed" << endl;
    break;
case CipherLab::DataConverter::Cancelled:
    cout << "Cancelled" << endl;
    break;
case CipherLab::DataConverter::Unknown:
    cout << "Unknown" << endl;
    break;
case CipherLab::DataConverter::SucceededWithRecordsSkipped:
    cout << "SucceededWithRecordsSkipped" << endl;
    break;
case CipherLab::DataConverter::Succeeded:
    cout << "Succeeded" << endl;
    break;
default:
    cout << "Totally Unknown" << endl;
}
cout << "Output File Paths: " << endl;
for(int i = 0; i < result.getNumberOfOutputFiles(); i++)
{
    cout << i << ") " << result.getOutputFilePaths(i) << endl;
}

cout << "ErrorRecord: " << endl;
for(int i = 0; i < result.getNumberOfErrorRecords(); i++)
{
    ErrorRecord e = result.getErrorRecord(i);
    cout << i << "): ";
    cout << "filename = " << e.getFileName() << endl;
    cout << e.getRecordNum() << endl;
    cout << e.getRecordText() << endl;
}
}
```



```
catch(formatException& e)
{
    cerr << e.getMessage() << endl;
    getchar();
    exit(0);
}
```

1.5.14 TXT2PACKEDDBFCONVERTER

Public Member Function:

```
DLLEXPORT Txt2PackedDbfConverter (BasicProgDataFormat &dataFormat, const char
*target, WrongFormatAction actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in basic programming data format.

```
DLLEXPORT Txt2PackedDbfConverter (CProgDataFormat &dataFormat, const char
*target, WrongFormatAction actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in c programming data format.

```
DLLEXPORT Txt2PackedDbfConverter (ForgeAgProgDataFormat &dataFormat, const char
*target, WrongFormatAction actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in Forge AG programming data format.

```
DLLEXPORT Txt2PackedDbfConverter (BasicProgDataFormat *dataFormat_list, int n,
const char *target, WrongFormatAction actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in multiple basic programming data formats.

```
DLLEXPORT Txt2PackedDbfConverter (CProgDataFormat *dataFormatList, int n, const
char *target, WrongFormatAction actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in multiple c programming data formats.

```
DLLEXPORT Txt2PackedDbfConverter (ForgeAgProgDataFormat *dataFormatList, int n,
const char *target, WrongFormatAction actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in multiple Forge AG programming data formats.

```
DLLEXPORT ~Txt2PackedDbfConverter ()
```

Destructor of Txt2PackedDbfConverter Class.

```
DLLEXPORT void CancelAsync ()
```

Requests cancellation of asynchronous conversion.

```
DLLEXPORT ConversionResult Convert (ProgressCallBack *cbPtr)
```

Requests conversion.

```
DLLEXPORT void ConvertAsync (ProgressCallBack *cb)
```

Requests action of asynchronous conversion.

Public Attribute:

```
ConversionResult result
```

The ConversionResult object for conversion result.

```
ConversionStatus status
```

The ConversionStatus object for conversion result.

CONSTRUCTOR & DESTRUCTOR

Syntax

```
CipherLab::DataConverter::Txt2PackedDbfConverter::Txt2PackedDbfConverter  
(BasicProgDataFormat &dataFormat, const char *target, WrongFormatAction  
actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in basic programming data format.

Parameters

dataFormat

Reference to a BasicProgDataFormat object.

target

The path for the output file(s).

actionTakenWhenWrong

The action to take when detecting wrong data format during conversion.

Syntax

```
CipherLab::DataConverter::Txt2DbfConverter::Txt2DbfConverter (CProgDataFormat &  
dataFormat, const char *targetDir, WrongFormatAction actionTakenWhenWrong)
```

Initializes a new instance of the Txt2DbfConverter class in C programming data format.

Parameters

dataFormat

Reference to a CProgDataFormat object.

targetDir

The path for the output file(s).

actionTakenWhenWrong

The action to take when detecting wrong data format during conversion.

Syntax

```
CipherLab::DataConverter::Txt2PackedDbfConverter::Txt2PackedDbfConverter  
(CProgDataFormat & dataFormat, const char *target, WrongFormatAction  
actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in c programming data format.

Parameters

dataFormat

Reference to a CProgDataFormat object.

target

The path for the output file(s).

actionTakenWhenWrong

The action to take when detecting wrong data format during conversion.

Syntax

```
CipherLab::DataConverter::Txt2PackedDbfConverter::Txt2PackedDbfConverter  
(ForgeAgProgDataFormat & dataFormat, const char *target, WrongFormatAction  
actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in Forge AG programming data format.

Parameters

dataFormat

Reference to a ForgeAgProgDataFormat object.

target

The path for the output file(s).

actionTakenWhenWrong

The action to take when detecting wrong data format during conversion.

Syntax

```
CipherLab::DataConverter::Txt2PackedDbfConverter::Txt2PackedDbfConverter  
(BasicProgDataFormat *dataFormat_list, int n, const char *target, WrongFormatAction  
actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in multiple basic programming data formats.

Parameters

dataFormat_list

Points to BasicProgDataFormat object(s) array.

n

The number of dataformat.

target

The path for the output file(s).

actionTakenWhenWrong

The action to take when detecting wrong data format during conversion.

Syntax

```
CipherLab::DataConverter::Txt2PackedDbfConverter::Txt2PackedDbfConverter  
(CProgDataFormat *dataFormatList, int n, const char *target, WrongFormatAction  
actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in multiple c programming data formats.

Parameters

dataFormat_list

Points to BasicProgDataFormat object(s) array.

n

The number of dataformat.

target

The path for the output file(s).

actionTakenWhenWrong

The action to take when detecting wrong data format during conversion.

Syntax

```
CipherLab::DataConverter::Txt2PackedDbfConverter::Txt2PackedDbfConverter  
(ForgeAgProgDataFormat *dataFormatList, int n, const char *target, WrongFormatAction  
actionTakenWhenWrong)
```

Initializes a new instance of the Txt2PackedDbfConverter class in multiple Forge AG programming data formats.

Parameters

dataFormat_list

Points to ForgeAgProgDataFormat object(s) array.

n

The number of dataformat.

target

The path for the output file(s).

actionTakenWhenWrong

The action to take when detecting wrong data format during conversion.

DETAILS OF MEMBER FUNCTION

Syntax

```
ConversionResult CipherLab::DataConverter::Txt2PackedDbfConverter::Convert  
( ProgressCallBack *cbPtr )
```

Requests conversion.

Parameters

cbPtr

A function pointing to the user-defined progress report function. If there is no need to report progress, set cbPtr to NULL.

Return Value

Points to the ConversionResult object for conversion result.

Example

```
BasicProgDataFormat bpdf = BasicProgDataFormat();
bpdf.setDelimiter(0x2C);
bpdf.DbfFileName = CipherLab::DataConverter::F1;
string path = "D:\\DcTestFolder\\testFiles\\D\\input_delimiter_15.txt";
bpdf.setPath(path.c_str());
bpdf.ParseRecordFields(bpdf.getPath());

bpdf.EditRecordFieldIsKey(0, true);
bpdf.EditRecordFieldIsKey(1, true);
bpdf.EditRecordFieldIsKey(2, true);

for(int i = 0; i < bpdf.getNumberOfRecordFields(); i++)
{
    RecordField a = bpdf.GetRecordFieldAt(i);
    cout << "recordVector[" << i << "]" << " offset(" << a.Position << ") length(" <<
    a.Length << ")" << endl;
}
Txt2PackedDbfConverter* t2dObj = new Txt2PackedDbfConverter(bpdf,
"D:\\DcTestFolder\\PC2SD\\Delimiter\\API\\output", Reformat);

try{
    ConversionResult result;
    ProgressCallBack* cbPtr = new ProgressCallBack();
    cbPtr->SetCallBack(progressCallBack_txt2packedDbf);
    result = t2dObj->Convert(cbPtr);

    switch(result.status)
    {
    case CipherLab::DataConverter::Failed:
        cout << "Failed" << endl;
        break;
    case CipherLab::DataConverter::Cancelled:
        cout << "Cancelled" << endl;
        break;
    case CipherLab::DataConverter::Unknown:
        cout << "Unknown" << endl;
```

```
        break;
    case CipherLab::DataConverter::SucceededWithRecordsSkipped:
        cout << "SucceededWithRecordsSkipped" << endl;
        break;
    case CipherLab::DataConverter::Succeeded:
        cout << "Succeeded" << endl;
        break;
    default:
        cout << "Totally Unknown" << endl;
}

cout << "Output File Paths: " << endl;
for(int i = 0; i < result.getNumberOfOutputFiles(); i++)
{
    cout << i << ") " << result.getOutputFilePaths(i) << endl;
}

for(int i = 0; i < result.getNumberOfErrorRecords(); i++)
{
    if (i == 0)
        cout << "ErrorRecord: " << endl;
    ErrorRecord e = result.getErrorRecord(i);
    cout << i << "): ";
    cout << "filename = " << e.getFileName() << endl;
    cout << e.getRecordNum() << endl;
    cout << e.getRecordText() << endl;
}
}
catch(formatException& e)
{
    cerr << e.getMessage() << endl;
    getchar();
    exit(0);
}
```

Syntax

```
void CipherLab::DataConverter::Txt2PackedDbfConverter::ConvertAsync
(ProgressCallBack *cb)
```

Requests action of asynchronous conversion.

Parameters

cb

A function pointing to the user-defined progress report function. If there is no need to report progress, set *cb* to NULL.

Example

```
ForgeAgProgDataFormat fpdf = ForgeAgProgDataFormat();
fpdf.setDelimiter(0x2C);
fpdf.DbfFileName = CipherLab::DataConverter::FirstLookup;
string path = "D:\\DcTestFolder\\testFiles\\D\\input_delimiter_15.txt";
fpdf.setPath(path.c_str());
fpdf.ParseRecordFields(fpdf.getPath());
fpdf.EditRecordFieldIsKey(0, true);

ForgeAgProgDataFormat fpdf1 = ForgeAgProgDataFormat();
fpdf1.setDelimiter(0x2C);
fpdf1.DbfFileName = CipherLab::DataConverter::FirstLookup;
path = "D:\\DcTestFolder\\testFiles\\D\\input_delimiter_15.txt";
fpdf1.setPath(path.c_str());
fpdf1.ParseRecordFields(fpdf1.getPath());
fpdf1.EditRecordFieldIsKey(0, true);

ForgeAgProgDataFormat a[2];
a[0] = fpdf;
a[1] = fpdf1;

Txt2PackedDbfConverter* t2dObj = new Txt2PackedDbfConverter(a, 2,
"D:\\DcTestFolder\\PC2SD\\Delimiter\\API\\ForgeAg\\output", Reformat);

try{
    ConversionResult result;
    ProgressCallBack* cbPtr = new ProgressCallBack();
    cbPtr->SetCallBack(progressCallBack_txt2packedDbf);
    t2dObj->ConvertAsync(cbPtr);
```



```
result = t2dObj->result;

switch(result.status)
{
case CipherLab::DataConverter::Failed:
    cout << "Failed" << endl;
    break;
case CipherLab::DataConverter::Cancelled:
    cout << "Cancelled" << endl;
    break;
case CipherLab::DataConverter::Unknown:
    cout << "Unknown" << endl;
    break;
case CipherLab::DataConverter::SucceededWithRecordsSkipped:
    cout << "SucceededWithRecordsSkipped" << endl;
    break;
case CipherLab::DataConverter::Succeeded:
    cout << "Succeeded" << endl;
    break;
default:
    cout << "Totally Unknown" << endl;
}

cout << "Output File Paths: " << endl;
for(int i = 0; i < result.getNumberOfOutputFiles(); i++)
{
    cout << i << ") " << result.getOutputFilePaths(i) << endl;
}

for(int i = 0; i < result.getNumberOfErrorRecords(); i++)
{
    if (i == 0)
        cout << "ErrorRecord: " << endl;
    ErrorRecord e = result.getErrorRecord(i);
    cout << i << "): ";
```

```
        cout << "filename = " << e.GetFileName() << endl;
        cout << e.getRecordNum() << endl;
        cout << e.getRecordText() << endl;
    }
}
catch(formatException& e)
{
    cerr << e.getMessage() << endl;
    getchar();
    exit(0);
}
```


.NET PROGRAMMING

Please prepare the necessary library file (**CipherLab.Data.ConverterLib.dll**) on the CD-ROM before you start to write and implement your application.

In order for your application to run on 64-bit Windows, you must have Microsoft Visual C++ 2008 Redistributable Package installed on the 64-bit Windows machine. Set the platform target property to 'x86' for your C# or VB projects.

For C# Projects:

- 1) Right click the project in the solution explorer and open 'Properties'
- 2) Choose the Build tab
- 3) Set the Platform Target property to 'X86'

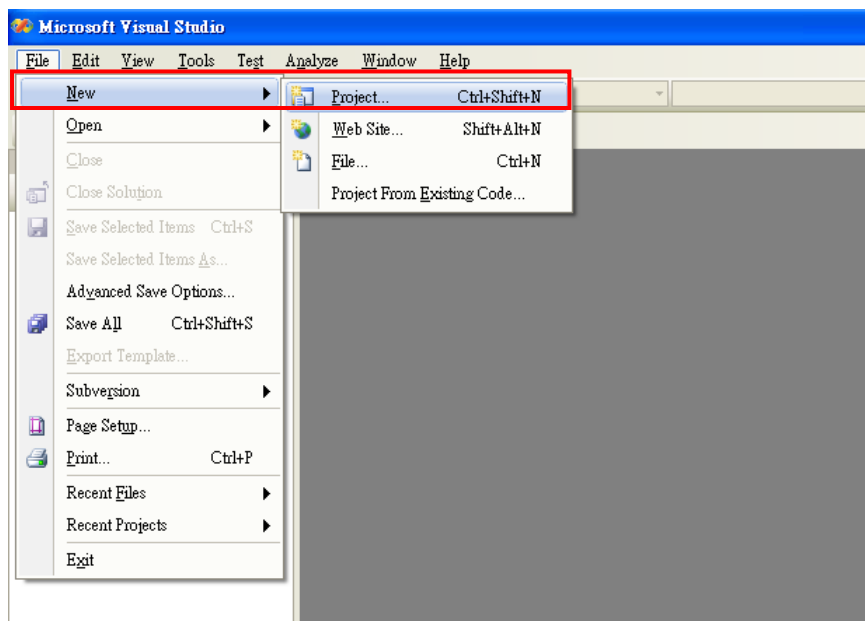
For VB Projects:

- 1) Right click the project in the solution explorer and open 'Properties'
- 2) Choose the Compile tab
- 3) Press the Advanced Compile Options... button
- 4) Set the Target CPU property to 'X86'

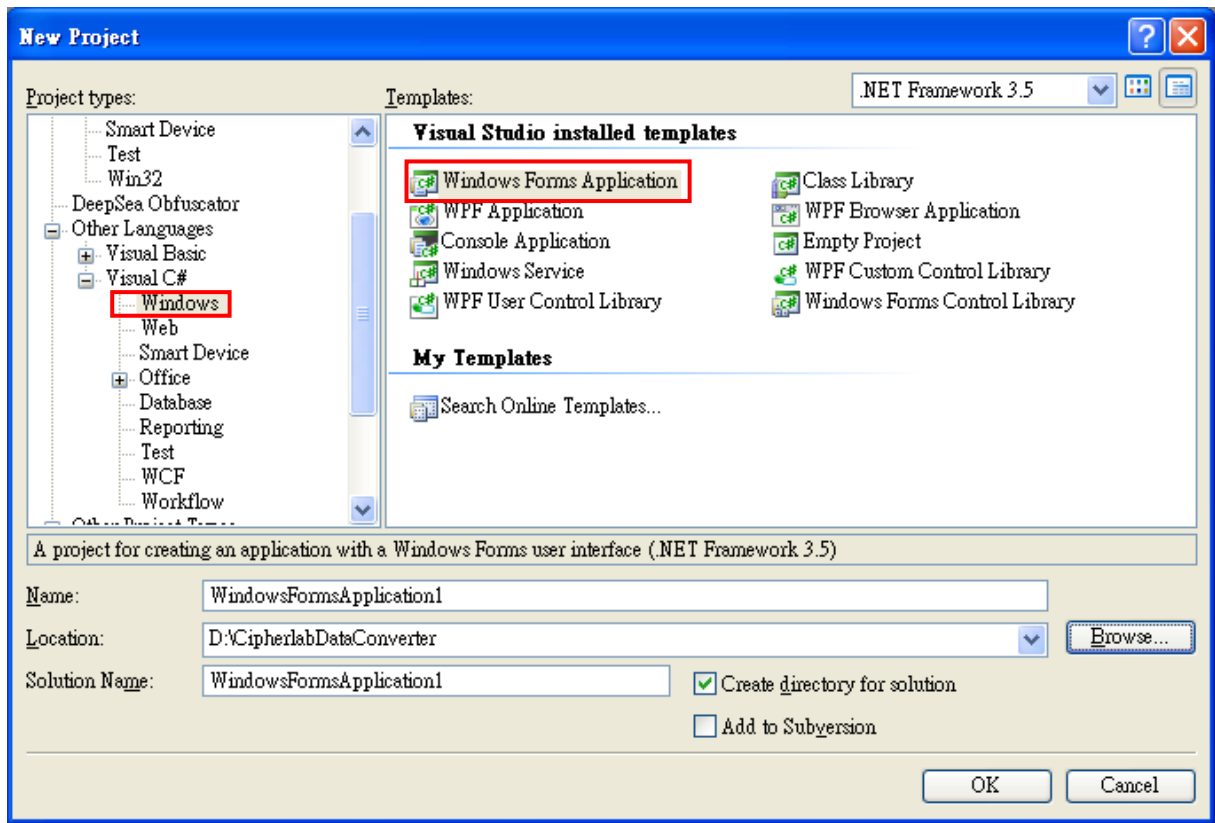
2.1 CREATE A VISUAL C# APPLICATION

Follow these steps to create a Visual C# application in Visual Studio 2008.

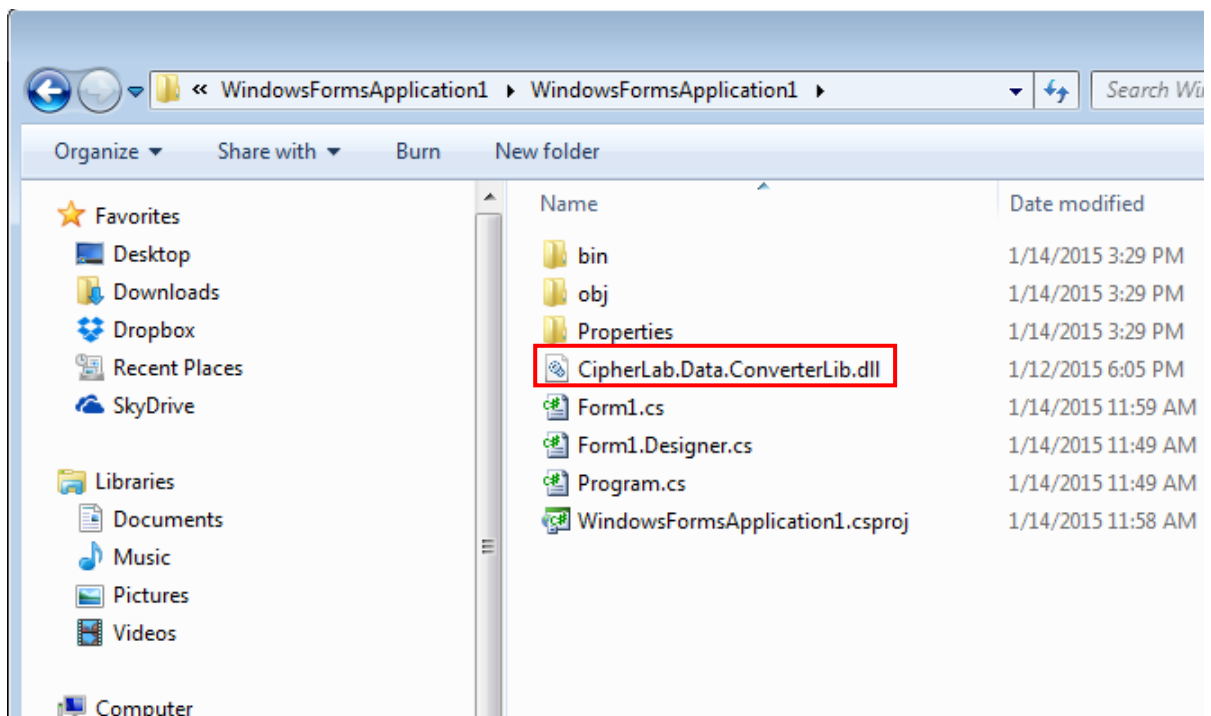
- 1) On the **File** menu, point to **New**, and then click **Project**.



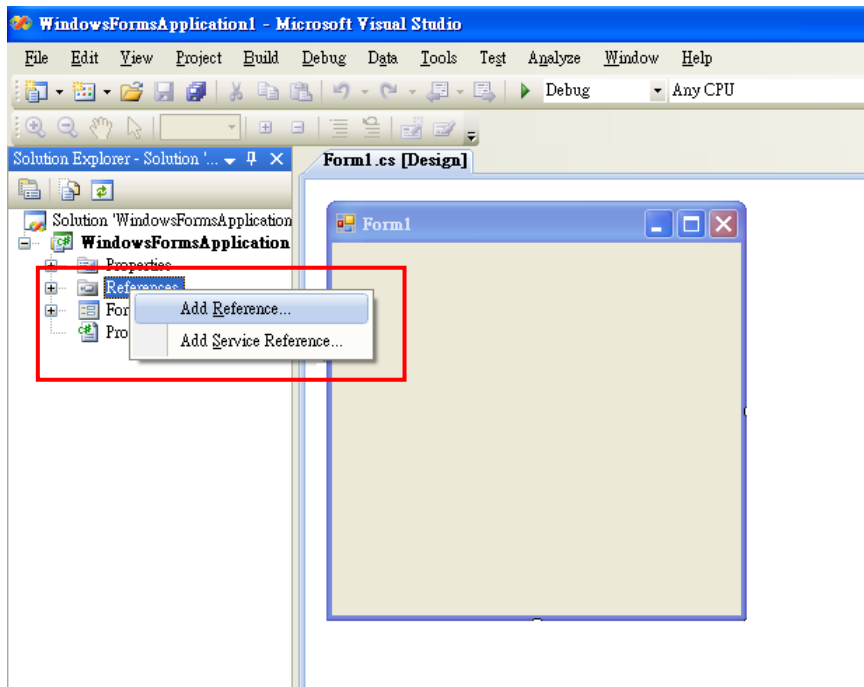
- 2) Select the **Windows** project type in the left pane and then the **Windows Forms Application** template in the right pane.



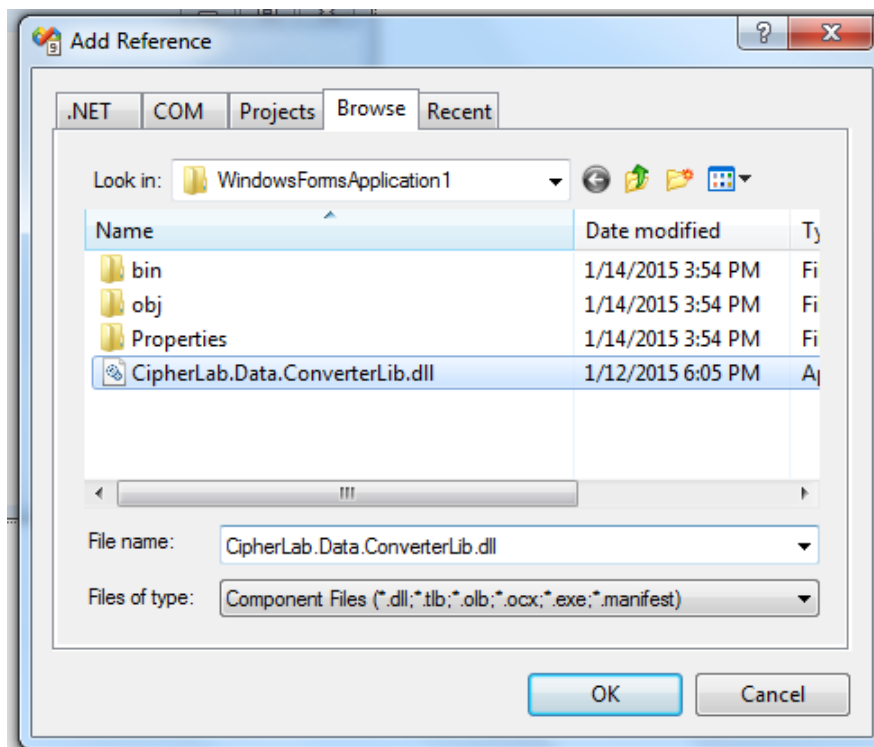
- 3) Copy the **CipherLab.Data.ConverterLib.dll** file to the project folder.



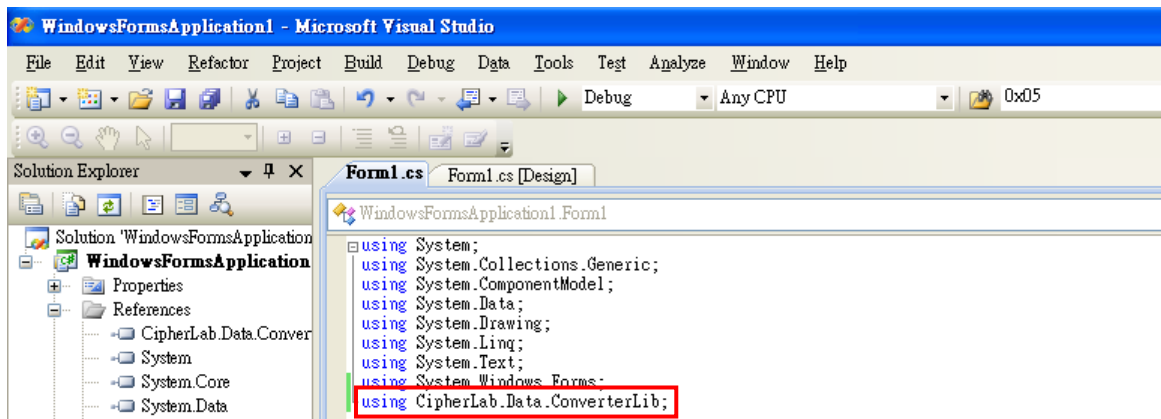
- 4) Right-click **References** and select **Add Reference**.



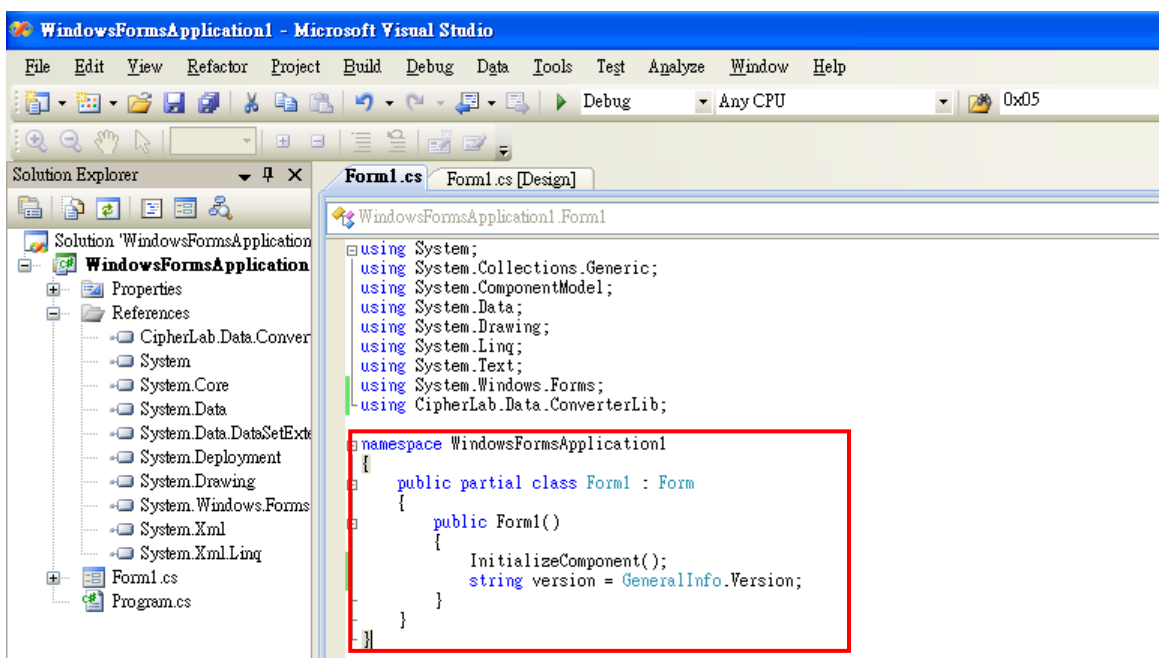
- 5) Click the **Browse** tab and select the **CipherLab.Data.ConverterLib.dll** file. Click **OK**.



- 6) You should include “using CipherLab.Data.ConverterLib” to use the System DLL file.



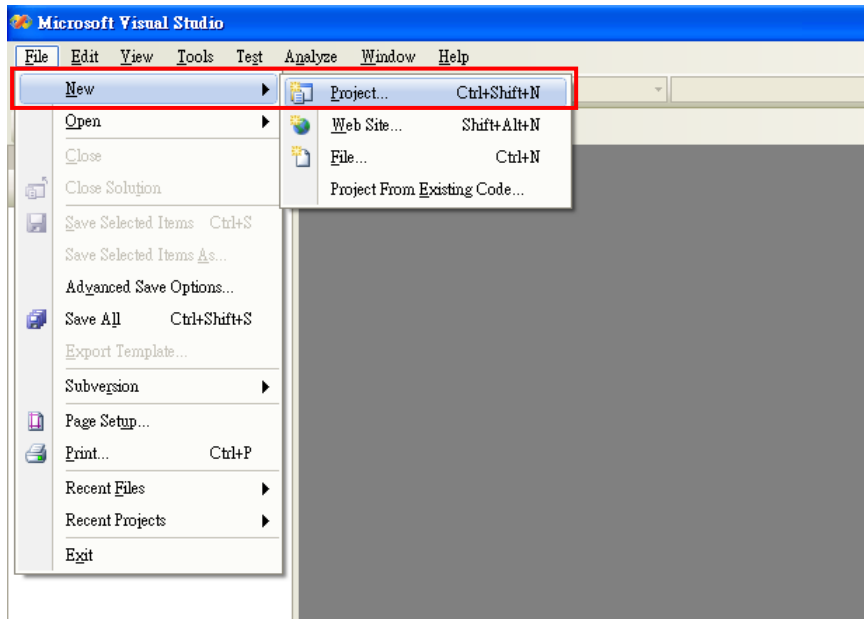
- 7) Start to write your code.



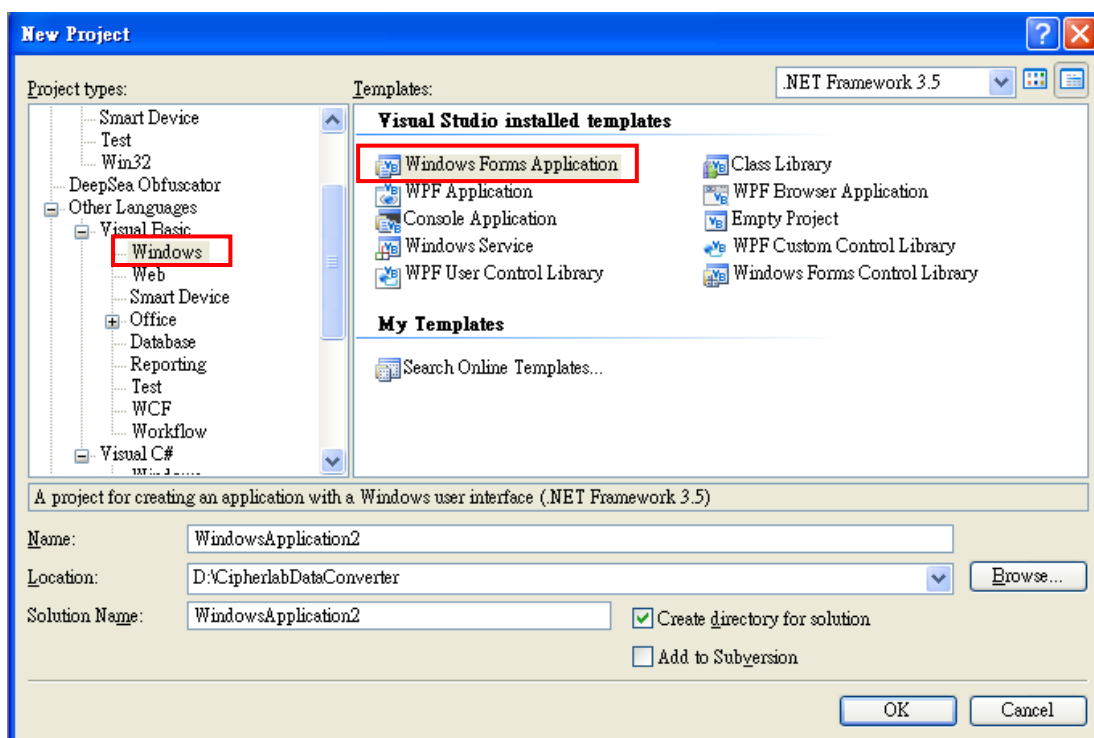
2.2 CREATE A VISUAL BASIC APPLICATION

Follow these steps to create a Visual Basic application in Visual Studio 2008.

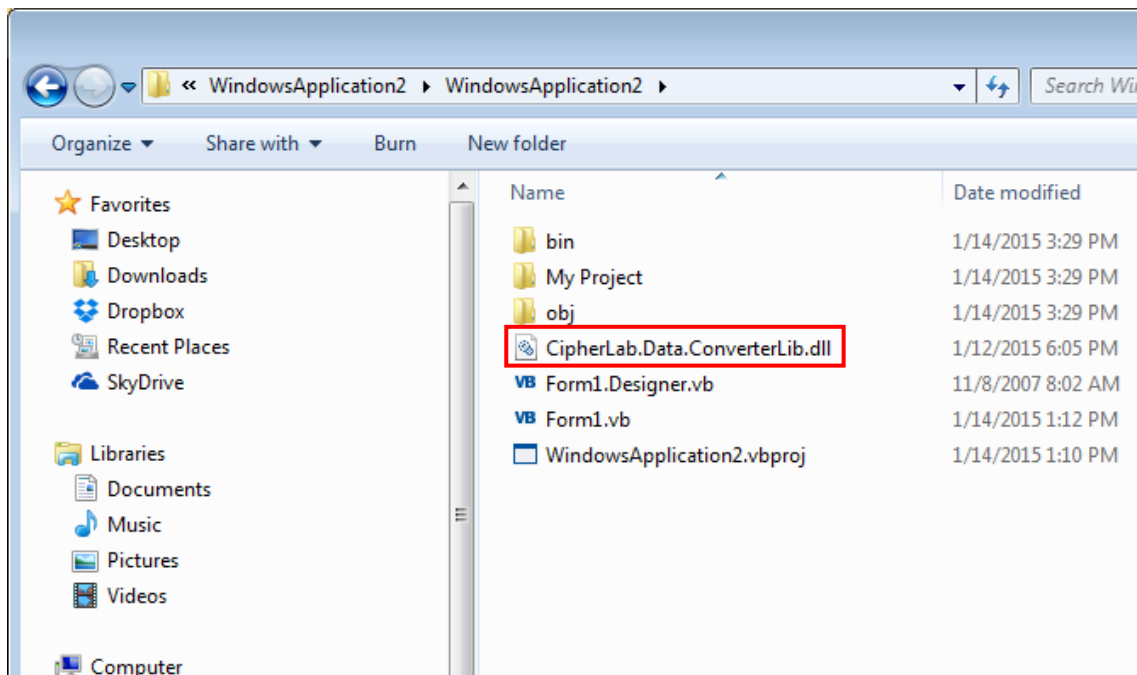
- 1) On the **File** menu, point to **New**, and then click **Project**.



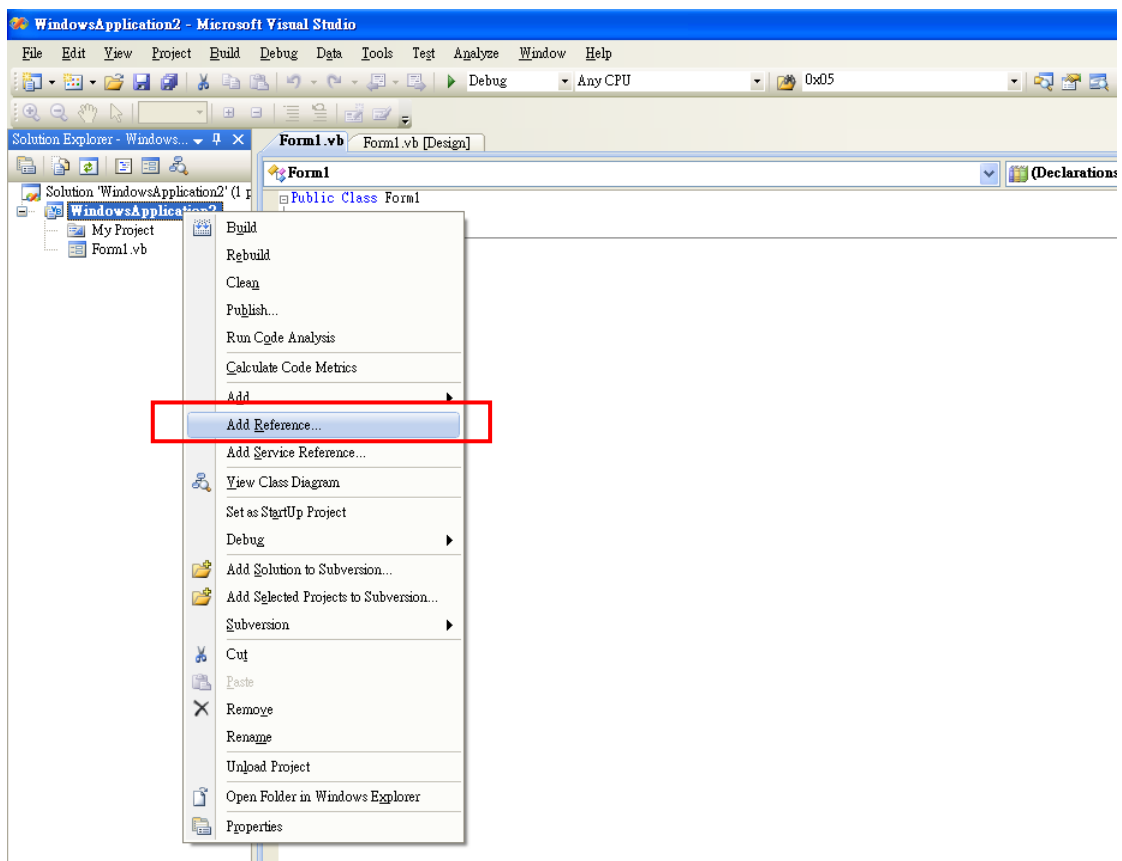
- 2) Select the **Windows** project type in the left pane and then the **Windows forms Application** template in the right pane.



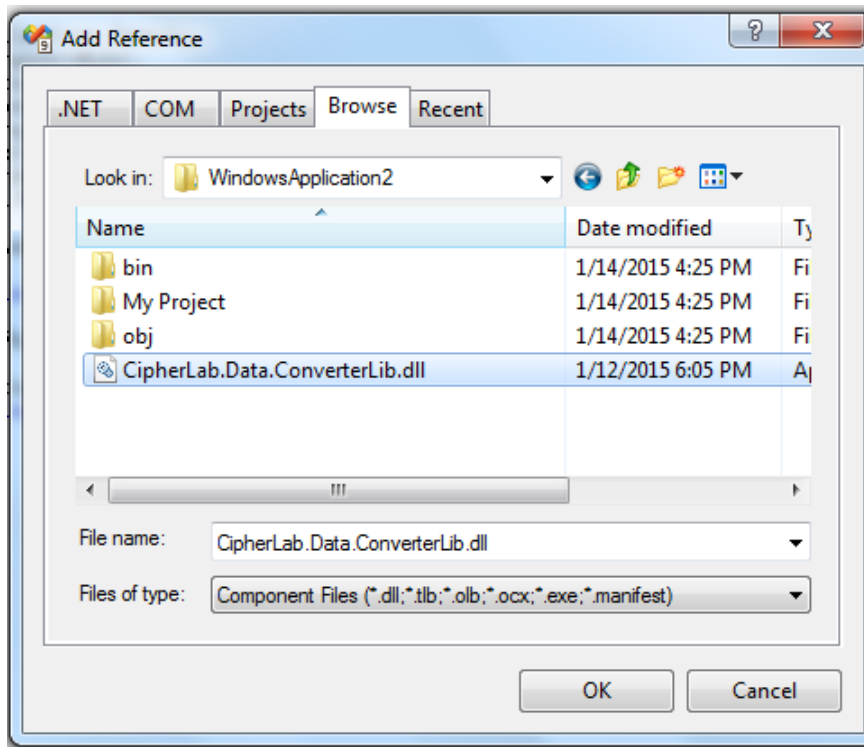
- 3) Copy the **CipherLab.Data.ConverterLib.dll** file to the project folder.



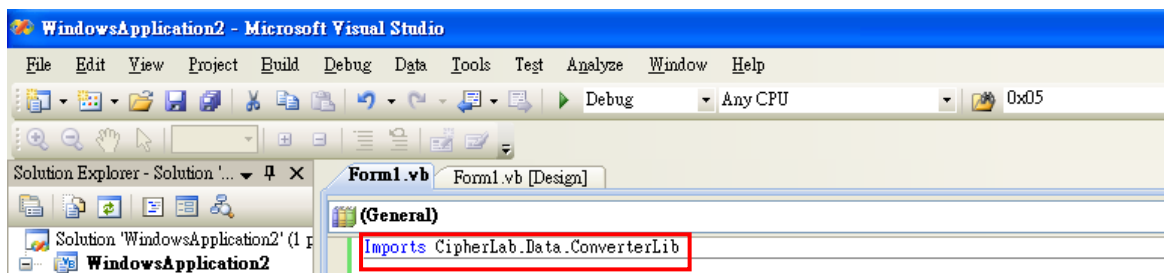
- 4) Right-click **WindowsApplication2** and then select **Add Reference**.



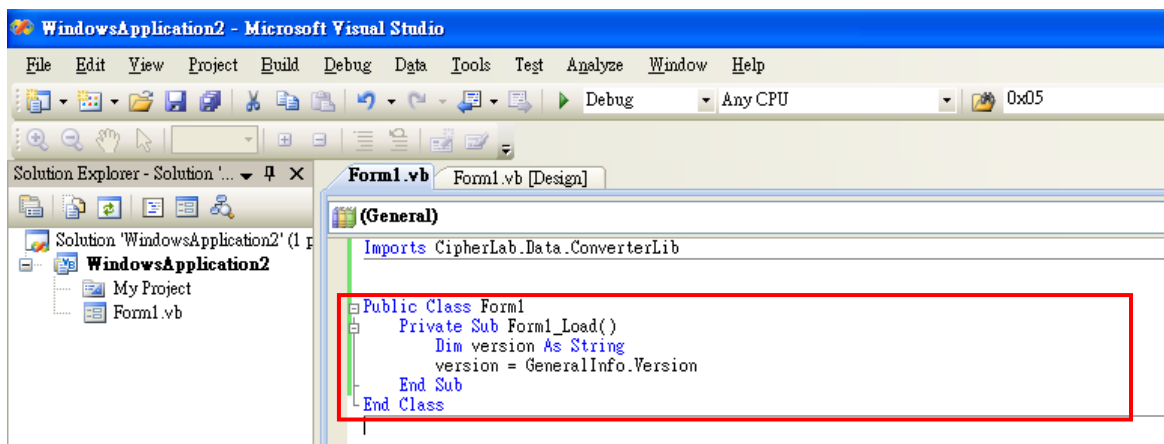
- 5) Click the **Browse** tab and then select the **CipherLab.Data.ConverterLib.dll** file. Click **OK**.



- 6) You should include "Imports CipherLab" to use the System DLL file.



- 7) Start to write your code.



2.3 CONVERTER DELEGATE

2.3.1 PROGRESSCALLBACK DELEGATE

The ProgressCallback delegate is for the convert function acting as a callback function to report the percentage, record count and status of conversion process.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public delegate void ProgressCallback(float value, long recordCount, string status);
```

Parameters

value

Type: float

Specifies the percentage of progress.

recordCount

Type: long

Specifies the current processed record count.

status

Type: string

Indicates the processing status.

2.4 CONVERTER API ENUMERATIONS

All the following examples refer to Programming in Visual Studio C#.

2.4.1 AGXTYPE

The **AGXType** enumeration defines the type of AGX format for 8-series.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public enum AGXType
{
    Series86,
    Series86_Encrypted,
    Others,
    Others_Encrypted
}
```

Members

Series86

The AGX format type for the 8600 model.

Series86_Encrypted

The AGX format type for the 8600 model to support Forge AG 2.0 or later.

Others

The AGX format type for 8-series except for 8600.

Others_Encrypted

The AGX format type for 8-series except for 8600 to support Forge AG 2.0 or later.

2.4.2 BASICPROGDBFFILENAME

The **BasicProgDbfFileName** enumeration defines the DBF file names for the Basic programming PC to RAM and PC to SD card conversions.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public enum BasicProgDbfFileName
{
    F1 = 1,
    F2,
    F3,
    F4,
    F5
}
```

Members

F1

The filename part of the DBF file is F1.

F2

The filename part of the DBF file is F2.

F3

The filename part of the DBF file is F3.

F4

The filename part of the DBF file is F4.

F5

The filename part of the DBF file is F5.

2.4.3 CONVERSIONSTATUS

The **ConversionStatus** enumeration defines the current status of conversion process.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public enum ConversionStatus
{
    Failed = -2,
    Cancelled,
    Unknown,
    SucceededWithRecordsSkipped,
    Succeeded
}
```

Members

Failed=-2

Conversion status is failed.

Cancelled

Conversion status is cancelled.

Unknown

Conversion status is unknown.

SucceededWithRecordsSkipped

Conversion status succeeds with some records skipped.

Succeeded

Conversion status succeeds.

2.4.4 FORGEAGPROGDBFFILENAME

The **ForgeAgProgDbfFileName** enumeration defines the type of lookup file for Forge AG programming record format.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public enum ForgeAgProgDbfFileName
{
    FirstLookup = 1,
    SecondLookup,
    ThirdLookup
}
```

Members

FirstLookup

First lookup file.

SecondLookup

Second lookup file.

ThirdLookup

Third lookup file.

2.4.5 PROGRAMMING TOOL

The **ProgrammingTool** enumeration defines the programming tool employed by users to program the application.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.DataConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public enum ProgrammingTool
{
    C,
    Basic,
}
```

Members

C

C programming tool.

Basic

Basic programming tool.

2.4.6 WRONGFORMATACTION

The **WrongFormatAction** enumeration defines the actions to take when encountering wrong data format during conversion.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.DataConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public enum WrongFormatAction
{
    Stop,
    Reformat,
    Skip
}
```

Members

Stop

Stop the conversion process when data format is not correct.

Reformat

Truncate and reformat the data record when the format is not correct.

Skip

Skip the current record and continue with the next record.

2.5 CLASSES

2.5.1 AGX

Reads data format from an AGX file.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public static class AGX
```

Methods

	Name	Description
public	Read	Reads data format from an AGX file.
public	DetermineType	Reads the header of an AGX file to determine the AGX format type of 8-series mobile computers.

2.5.2 BASICPROGDATAFORMAT

Defines record format for Basic programming conversions.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public sealed class BasicProgDataFormat: DataFormat
```

Constructors

	Name	Description
public	BasicProgDataFormat	Initializes a new instance of the BasicProgDataFormat class.
public	BasicProgDataFormat(string, BasicProgDbfFileName, WrongFormatAction)	Initializes a new instance of the BasicProgDataFormat class with input file path, output DBF file name, and the action to take when detecting wrong data format during conversion.
public	BasicProgDataFormat(string, BasicProgDbfFileName, char, WrongFormatAction)	Initializes a new instance of the BasicProgDataFormat class with input file path, output DBF file name, data delimiter, and the action to take when detecting wrong data format during conversion.

Properties

	Name	Description
public	Delimiter	Gets or sets the delimiter character that is used in the record to separate data.
public	KeyCount	Gets the number of key fields
public	IsRecordDelimited	Gets a value indicating whether the record uses a delimiter character to separate data.
public	MaxRecordFields	Gets the maximum number of fields allowed.
public	NumberOfRecordFields	Gets the number of data fields in a record.
public	Path	Gets or sets the input file path

Methods

	Name	Description
public	AddRecordField(RecordField)	Adds a record field to the data format.
public	AddRecordField(List<RecordField>)	Adds a list of record fields to the data format.
public	GetRecordFieldAt	Gets the record field at specified location.
public	ParseRecordFields	Auto parses the lookup file to list of record fields.

Fields

	Name	Description
public	DbfFileName	Identifies the output DBF file name.
public	KeyIndices	Identifies the index of the key record.
public	Keys	Identifies the record fields of the key record.

Remarks

- ▶ The maximum number of record fields for both non-8600 models and 8600 model is 8.
- ▶ Txt2PackedDbf conversion:
For non-8600 model, the format can have 1~3 key fields specified; as for 8600 model, the format can have 0~5 key fields specified.
- ▶ Txt2Dbf conversion:
For non-8600 model, the format can have 0~3 key fields specified; as for 8600 model, the format can have 0~5 key fields specified.

2.5.3 CONVERSIONRESULT

Gets the post conversion result including the possible ErrorRecord list and the output file paths.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public sealed class ConversionResult
```

Constructors

	Name	Description
public	Status	Returns the defined type of ConversionStatus enumeration.
public	ErrorRecords	Returns the array of ErrorRecord objects.
public	OutputFilePaths	Returns the array of output files' path.

2.5.4 CPROGDATAFORMAT

Defines record format for C programming conversions.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public sealed class CProgDataFormat : DataFormat
```

Constructors

	Name	Description
public	CProgDataFormat	Initializes a new instance of the CProgDataFormat class.
public	CProgDataFormat(string, WrongFormatAction)	Initializes a new instance of the CProgDataFormat class with input file path and the action to take when detecting wrong data format during conversion.
public	CProgDataFormat(string, string, WrongFormatAction)	Initializes a new instance of the CProgDataFormat class with input file path, output DBF file name, and the action to take when detecting wrong data format during conversion.
public	CProgDataFormat(string, string, string, WrongFormatAction)	Initializes a new instance of the CProgDataFormat class with input file path, output DBF file name, data delimiter, and the action to take when detecting wrong data format during conversion.

Properties

	Name	Description
public	Delimiter	Gets or sets the delimiter character that is used in the record to separate data.
public	IsRecordDelimited	Gets a value indicating whether the record uses a delimiter character to separate data.
public	KeyCount	Gets the number of key fields
public	MaxRecordField	Gets the maximum number of fields allowed.
public	NumberOfRecordFields	Gets the number of data fields in a record.

public	Path	Gets or sets the input file path
--------	------	----------------------------------

Methods

	Name	Description
public	AddRecordField(RecordField)	Adds a record field to the data format.
public	AddRecordField(List<RecordField>)	Adds a list of record fields to the data format.
public	GetRecordFieldAt	Gets the record field at specified location.
public	ParseRecordFields	Auto parses the lookup file to list of record fields.

Fields

	Name	Description
public	DbfFileName	Identifies the output DBF file name.
public	KeyIndices	Identifies the index of the key record.
public	Keys	Identifies the record fields of the key record.

Remarks

- ▶ The maximum number of record fields for both non-8600 models and 8600 model is 8.
- ▶ Txt2PackedDbf conversion:
For non-8600 model, the format can have 1~8 key fields specified; as for 8600 model, the format can have 0~8 key fields specified.
- ▶ Txt2DbfConversion:
For both non-8600 models and 8600 model, the format can have 0~8 key fields specified.

2.5.5 DBF2TXTCONVERTER

Converts DBF or transaction (DAT) files on SD card to text files.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public class Dbf2TxtConverter
```

Constructors

	Name	Description
public	Dbf2TxtConverter (ProgrammingTool, string, string)	Initializes a new instance of the Dbf2TxtConverter class with specific programming language, input file path, and the output file path.

Methods

	Name	Description
public	CancelAsync	Requests cancellation of conversion.
public	ConvertAsync	Requests action of conversion.
public	Convert	Requests conversion.
public	Convert with Callback	Requests conversion with callback function.
public	ProgressChangedDelegate	Raises the ProgressChanged event.
public	RunWorkerCompletedDelegate	Raises the RunWorkerCompleted event.

Events

	Name	Description
public	ProgressChanged	Occurs when user raises the ProgressChanged event.
public	RunWorkerCompleted	Occurs when the conversion has completed, has been canceled, or has raised an exception.

Example

```
using CipherLab.Data.ConverterLib;

static void Main(string[] args)
{
    string source = "D:\\DCTester3\\transaction\\DBF1.DAT";
    string target = "D:\\DBF1.txt";

    Dbf2TxtConverter t2TInstance = new Dbf2TxtConverter (ProgrammingTool.Basic,
source, target);
```

```
t2TInstance.ProgressChanged += new ProgressChangedEventHandler(Report);  
t2TInstance.RunWorkerCompleted += new  
RunWorkerCompletedEventHandler(Complete);  
  
t2TInstance.ConvertAsync();  
}
```

2.5.6 DELIMITERCHAR

Defines the character of delimiter.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public static class DelimiterChar
```

Fields

	Name	Description
public	None	The character represents 0x00.

2.5.7 ERRORRECORD

Records errors during processing data conversion.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public class ErrorRecord
```

Constructors

	Name	Description
public	ErrorRecord (string, long, string, bool)	Initializes a new instance of the ErrorRecord class with input file path, the sequence of the record, the record text and whether the action to take when detecting wrong data format during conversion is 'skip' or not.

Fields

	Name	Description
public	filename	Identifies the input filename.
public	numOfLines	Identifies the sequence of the record.
public	textOfRecord	Identifies the text of record.
public	isSkip	Identifies whether the action to take when detecting wrong data format during conversion is 'skip' or not.

2.5.8 FORGEAGPROGDATAFORMAT

Defines record format for Forge AG programming conversions.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public sealed class ForgeAgProgDataFormat: DataFormat
```

Constructors

	Name	Description
public	ForgeAgProgDataFormat	Initializes a new instance of the ForgeAgProgDataFormat class.
public	ForgeAgProgDataFormat(string, ForgeAgProgDbfFileName, WrongFormatAction)	Initializes a new instance of the ForgeAgProgDataFormat class with input file path, output DBF file name, and the action to take when detecting wrong data format during conversion.
public	ForgeAgProgDataFormat(string,	Initializes a new instance of the

	ForgeAgProgDbfFileName, char, WrongFormatAction)	ForgeAgProgDataFormat class with input file path, output DBF file name, data delimiter, and the action to take when detecting wrong data format during conversion.
--	--	--

Properties

	Name	Description
public	Delimiter	Gets or sets the delimiter character that is used in the record to separate data.
public	IsRecordDelimited	Gets a value indicating whether the record uses a delimiter character to separate data.
public	KeyCount	Gets the number of key fields
public	MaxRecordField	Gets the maximum number of fields allowed.
public	NumberOfRecordFields	Gets the number of data fields in a record.
public	Path	Gets or sets the input file path

Methods

	Name	Description
public	AddRecordField(RecordField)	Adds a record field to the data format.
public	AddRecordField(List<RecordField>)	Adds a list of record fields to the data format.
public	GetRecordFieldAt	Gets the record field at specified location.
public	ParseRecordFields	Auto parses the lookup file to list of record fields.

Fields

	Name	Description
public	ActionTakenWhenFormatIsWrong	Gets the wrong format action.
public	DbfFileName	Identifies the output DBF file name.
public	KeyIndices	Identifies the index of the key record.
public	Keys	Identifies the record fields of the key record.

Remarks

- ▶ The maximum number of record fields for non-8600 models is 8; as for 8600, the maximum number of record fields is 12.
- ▶ For both Txt2PackedDbf conversion and Txt2Dbf conversions, the format must have one and only one key field specified.

2.5.9 GENERALINFO

This class represents the general information associated with the DLL file.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public static class GeneralInfo
```

Fields

	Name	Description
public	Version	Gets a string representing the version associated with the DLL file.

2.5.10 RECORDFIELD

This class represents position, length, and key field information that define a record field in the CipherLab data format.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public class RecordField
```

Constructors

	Name	Description
public	RecordField(int, int)	Initializes a new instance of the RecordField class with data position and length specified by integer values.
public	RecordField(int, int, bool)	Initializes a new instance of the RecordField class with data position, data length, and a bool value that specifies whether the field is a key field in the CipherLab format.

Fields

	Name	Description
public	IsKey	Gets or sets a value indicating whether this record field is a key field.
public	Length	Gets or sets the Length of this data field.
public	Position	Gets or sets the start position of this data field.

2.5.11 TXT2DBFCONVERTER

Converts lookup (.txt) files to DBF files for the SD card device.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public class Txt2DbfConverter
```

Constructors

	Name	Description
public	Txt2DbfConverter(DataFormat, string)	Initializes a new instance of the Txt2DbfConverter class with specific data format and the output file path.

Methods

	Name	Description
public	CancelAsync	Requests cancellation of conversion.
public	ConvertAsync	Requests action of conversion.
public	Convert	Requests conversion without callback function.
public	Convert with Callback	Requests conversion with callback function.
public	ProgressChangedDelegate	Raises the ProgressChanged event.
public	RunWorkerCompletedDelegate	Raises the RunWorkerCompleted event.

Events

	Name	Description
public	ProgressChanged	Occurs when user raises the ProgressChanged event.
public	RunWorkerCompleted	Occurs when the conversion has completed, has been canceled, or has raised an exception.

Fields

	Name	Description
public	errorList	Returns a list of error records.

Example

```
using CipherLab.Data; ConverterLib;

public static void myCallback(int value)
{
    Console.WriteLine("Progress: {0}", value);
}

static void Main(string[] args)
{
    ForgeAgProgDataFormat[] fag=
    AGX.Read("D:\\Work\\DataConverter\\AG_8600\\8600_Basic_Test1.AGX");

    List<DataFormat> forgeList = new List<DataFormat>();

    for (int i = 0; i < fag.Length; i++)
    {
        DataFormat f = fag[i];
```

```
        Console.WriteLine("Action Taken: {0}",
((ForgeAgProgDataFormat)f).ActionTakenWhenFormatIsWrong);
        Console.WriteLine("*****{0}*****", i);
        for(int j = 0; j < f.NumberOfRecordFields; j++)
        {
            RecordField a = f.GetRecordFieldAt(j);
            Console.WriteLine("IsKey: {0}", a.IsKey);
            Console.WriteLine("Length: {0}", a.Length);
            Console.WriteLine("Position: {0}", a.Position);
        }
        Console.WriteLine("Delimierter = {0}", fag[i].Delimiter);
        Console.WriteLine("Action = {0}",
fag[i].ActionTakenWhenFormatIsWrong);
        if (i == 0)
            fag[i].Path = "D:\\ 8600_Basic_Test1_12Fields.txt";
        else if (i == 1)
            fag[i].Path = "D:\\ 8600_Basic_Test1_11Fields.txt";
        else
            fag[i].Path = "D:\\ 8600_Basic_Test1_10FieldsND.txt";
        forgeList.Add((DataFormat)fag[i]);
    }

    Txt2PackedDbfConverter ainstance = new
Txt2PackedDbfConverter(forgeList, "D:\\ AG_8600\\Test1_Result\\aoutput",
fag[0].ActionTakenWhenFormatIsWrong);
    ainstance.Convert(myCallback);
}
```

2.5.12 TXT2PACKEDDBFCONVERTER

Converts lookup (.txt) files to DBF files for the RAM device.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public class Txt2PackedDbfConverter
```

Constructors

	Name	Description
public	Txt2PackedDbfConverter (DataFormat, string)	Initializes a new instance of the Txt2PackedDbf class with specific data format

		and the output file path.
public	Txt2PackedDbfConverter (List<DataFormat>, string)	Initializes a new instance of the Txt2PackedDbfConverter class with a list of data format and the output file path.

Methods

	Name	Description
public	CancelAsync	Requests cancellation of conversion.
public	ConvertAsync	Requests action of conversion.
public	Convert	Requests conversion
public	Convert with Callback	Requests conversion with callback function.
public	ProgressChangedDelegate	Raises the ProgressChanged event.
public	RunWorkerCompletedDelegate	Raises the RunWorkerCompleted event.

Events

	Name	Description
public	ProgressChanged	Occurs when user raises the ProgressChanged event.
public	RunWorkerCompleted	Occurs when the conversion has completed, has been canceled, or has raised an exception.

Fields

	Name	Description
public	errorList	Returns a list of error records.

Example

```
using CipherLab.Data. ConverterLib;

static void Main(string[] args)
{
    //Txt2PackedDbf ~ Multiple files in C

    List<DataFormat> cList = new List<DataFormat>();

    CProgDataFormat cpdf = new CProgDataFormat();
    cpdf.Delimiter = ',';
    cpdf.AddRecordField(new RecordField(1, 4, true));
}
```

```
cpdf.AddRecordField(new RecordField(6, 2, true));
cpdf.AddRecordField(new RecordField(9, 3, true));
cpdf.AddRecordField(new RecordField(13, 7, true));
cpdf.AddRecordField(new RecordField(21, 4, true));
cpdf.AddRecordField(new RecordField(26, 4, true));
cpdf.AddRecordField(new RecordField(31, 2, true));
cpdf.AddRecordField(new RecordField(34, 2, true));
cpdf.DbFileName = "CPCRAM1";
cpdf.Path = "D:\\DCTester\\CPC2RAM_M\\input8.txt";

cList.Add(cpdf);

CProgDataFormat cpdf2 = new CProgDataFormat();
cpdf2.Delimiter = ',';
cpdf2.AddRecordField(new RecordField(1, 4, true));
cpdf2.AddRecordField(new RecordField(6, 2, true));
cpdf2.AddRecordField(new RecordField(9, 3, false));
cpdf2.AddRecordField(new RecordField(13, 7, true));
cpdf2.AddRecordField(new RecordField(21, 4, true));
cpdf2.AddRecordField(new RecordField(26, 4, true));
cpdf2.AddRecordField(new RecordField(31, 2, true));
cpdf2.AddRecordField(new RecordField(34, 2, false));
cpdf2.DbFileName = "CPCRAM2";

cpdf2.Path = "D:\\DCTester\\CPC2RAM_M\\input8_1.txt";

cList.Add(cpdf2);

Txt2PackedDbfConverter test = new Txt2PackedDbfConverter(cList,
"D:\\DCTester\\CPC2RAM_M\\Input8k2", WrongFormatAction.Stop);

test.ProgressChanged += new ProgressChangedEventHandler(Report);
test.RunWorkerCompleted += new
RunWorkerCompletedEventHandler(Complete);

test.ConvertAsync();
}

public static void Report(object sender, ProgressChangedEventArgs e)
{
    Console.WriteLine("Current Percentage is = {0}", e.ProgressPercentage);
}

public static void Complete(object sender, RunWorkerCompletedEventArgs e)
{
    if (e.Cancelled)
    {
        Console.WriteLine("Canceled");
    }
    else
    {
        if (e.Result != null)
        {

```

```
        if (sender is Txt2PackedDbfConverter)
        {
            foreach (ErrorRecord record in ((Txt2PackedDbfConverter)sender).list)
            {
                Console.WriteLine("record.textOfRecord = {0}",
record.textOfRecord);
            }
        }
        else
        {
            Console.WriteLine("so null");
        }
    }

    if (e.Error != null)
    {
        Console.WriteLine(e.Error.Message);
    }
    else
    {
        // Finally, handle the case where the operation
        // succeeded.
        Console.WriteLine("Completed");
    }
}
}
```


2.6 CLASS METHODS

2.6.1 AGX.READ

Reads data format from a Forge AG (AGX) file.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public static ForgeAgProgDataFormat[] Read(  
    string path  
)
```

Parameters

path

Type: string

Specifies the AGX file path.

Return Value

Type: CipherLab.Data.ConverterLib.ForgeAgProgDataFormat[]

The array contains Forge AG programming data format.

Remarks

The AGX.Read method won't fill the Path property of the ForgeAgProgDataFormat class. Therefore, the Path property will always be null.

Example

```
using CipherLab.Data. ConverterLib;  
ForgeAgProgDataFormat[] agxDataFormats = AGX.Read("D:\\Files\\MyAgx.agx");
```

2.6.2 AGX.DETERMINETYPE

Reads the header of an AGX file to determine the AGX format type of 8-series.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public static AGXType DetermineType(string path) (  
    string path  
)
```

Parameters

path

Type: string

Specifies the AGX file path.

Return Value

Type: CipherLab.Data.ConverterLib.AGXType

The defined type of AGXType enumeration.

Example

```
using CipherLab.Data. ConverterLib;  
AGXType type = AGX.DetermineType("D:\\Files\\MyAgx.agx");
```

2.6.3 DATAFORMAT.ADDRECORDFIELD (RECORDFIELD)

Adds a RecordField to the DataFormat class.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public virtual void AddRecordField (  
    RecordField recordField  
)
```

Parameters

recordField

Type: CipherLab.Data.ConverterLib.RecordField

Specifies the record field.

2.6.4 DATAFORMAT.ADDRECORDFIELD(LIST<RECORDFIELD>)

Adds a list of RecordFields to the DataFormat class.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public virtual void AddRecordField (  
    List<RecordField>  
)
```

Parameters

List<recordField>

Type: List<CipherLab.Data.ConverterLib.RecordField>

Specifies the list of record fields.

2.6.5 DATAFORMAT.GETRECORDFIELDAT

Gets a RecordField from the specified location.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public RecordField GetRecordFieldAt (  
    int index  
)
```

Parameters

index

Type: int

Index of the record field.

2.6.6 DATAFORMAT.PARSERECORDFIELDS

Auto parses the lookup file to list of record fields.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public List<RecordField> ParseRecordFields (  
    string inputFilePath  
)
```

Parameters

inputFilePath

Type: string

The path of input lookup file.

2.6.7 DBF2TXTCONVERTER.CANCELASYNC

Requests cancellation of conversion.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public void CancelAsync()
```

Example

```
using CipherLab.Data. ConverterLib;  
Txt2DbfConverterConverter test = new Txt2DbfConverterConverter (bpdf,  
"D:\\Work\\8200");  
test.CancelAsync();
```

2.6.8 DBF2TXTCONVERTER.CONVERT

Requests action of conversion without callback request.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Example

```
using CipherLab.Data. ConverterLib;  
Dbf2Txt t2TInstance = new Dbf2Txt(ProgrammingTool.Basic, source, target);  
t2TInstance.Convert();
```

2.6.9 DBF2TXTCONVERTER.CONVERT WITH CALLBACK

Requests action of conversion with callback request.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Example

```
using CipherLab.Data. ConverterLib;  
Dbf2Txt t2TInstance = new Dbf2Txt(ProgrammingTool.Basic, source, target);  
t2TInstance.Convert(callback);
```

2.6.10 DBF2TXTCONVERTER.CONVERTASYNC

Requests action of conversion.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Example

```
using CipherLab.Data. ConverterLib;  
Dbf2Txt t2TInstance = new Dbf2Txt(ProgrammingTool.Basic, source, target);  
t2TInstance.ConvertAsync();
```

2.6.11 DBF2TXTCONVERTER.PROGRESSCHANGEDDELEGATE

Raises the ProgressChanged event.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public delegate void ProgressChangedDelegate(Object sender,  
ProgressChangedEventArgs e)
```

Parameters

sender

Type: CipherLab.Data.ConverterLib.Dbf2TxtConverter

The current class which can raise an event.

e

Type: System.ComponentModel.ProgressChangedEventArgs

An EventArgs that contains the event data.

Example

```
using CipherLab.Data. ConverterLib;
Dbf2TxtConverter t2TInstance = new Dbf2TxtConverter(ProgrammingTool.Basic, source,
target);
t2TInstance.ProgressChanged += new ProgressChangedEventHandler(Report);
public static void Report(object sender, ProgressChangedEventArgs e)
{
    Console.WriteLine("Current Percentage is = {0}", e.ProgressPercentage);
}
```

2.6.12 DBF2TXTCONVERTER.RUNWORKERCOMPLETEDDELEGATE

Raises the ProgressChanged event.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public delegate void RunWorkerCompletedDelegate(Object sender,
RunWorkerCompletedEventArgs e)
```

Parameters

sender

Type: CipherLab.Data.ConverterLib.Dbf2TxtConverter

The current class which can raise an event.

e

Type: System.ComponentModel.RunWorkerCompletedEventArgs

An EventArgs that contains the event data.

Example

```
using CipherLab.Data. ConverterLib;
Dbf2TxtConverter t2TInstance = new Dbf2TxtConverter(ProgrammingTool.Basic, source,
target);
t2TInstance.RunWorkerCompleted += new
RunWorkerCompletedEventHandler(Complete);
public static void Complete(object sender, RunWorkerCompletedEventArgs e)
{
    if (e.Cancelled)
    {
        Console.WriteLine("Canceled");
    }
    else
    {
        if (e.Error != null)
        {
            Console.WriteLine(e.Error.Message);
        }
        else
        {
            // Finally, handle the case where the operation
            // succeeded.
            Console.WriteLine("Completed");
        }
    }
}
```


2.6.13 TXT2DBFCONVERTER.CANCELASYNC

Requests cancellation of conversion.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public void CancelAsync()
```

Example

```
using CipherLab.Data. ConverterLib;  
Txt2DbfConverter test = new Txt2DbfConverter(cpdf,  
"D:\\Task\\DataConverter3\\Test"); test.CancelAsync();
```

2.6.14 TXT2DBFCONVERTER.CONVERT

Requests action of conversion without callback request.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Example

```
using CipherLab.Data. ConverterLib;  
Txt2DbfConverter test = new Txt2DbfConverter(cpdf,  
"D:\\Task\\DataConverter3\\Test");  
test.Convert();
```

2.6.15 TXT2DBFCONVERTER.CONVERT WITH CALLBACK

Requests action of conversion with callback request.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Example

```
using CipherLab.Data. ConverterLib;
Txt2DbfConverter test = new Txt2DbfConverter(cpdf,
"D:\\Task\\DataConverter3\\Test");
test.Convert(callback);
```

2.6.16 TXT2DBFCONVERTER.CONVERTASYNC

Requests action of conversion.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Example

```
using CipherLab.Data. ConverterLib;
Txt2DbfConverter test = new Txt2DbfConverter(cpdf,
"D:\\Task\\DataConverter3\\Test");
test.ConvertAsync();
```

2.6.17 TXT2DBFCONVERTER.PROGRESSCHANGEDDELEGATE

Raises the ProgressChanged event.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public delegate void ProgressChangedDelegate(Object sender,
ProgressChangedEventArgs e)
```

Parameters

sender

Type: CipherLab.Data.ConverterLib.Db2TxtConverter

The current class which can raise an event.

e

Type: System.ComponentModel.ProgressChangedEventArgs

An EventArgs that contains the event data.

Example

```
using CipherLab.Data. ConverterLib;
Txt2DbfConverter test = new Txt2DbfConverter(cpdf,
"D:\\Task\\DataConverter3\\Test");
test.ProgressChanged += new ProgressChangedEventHandler(Report);
public static void Report(object sender, ProgressChangedEventArgs e)
{
    Console.WriteLine("Current Percentage is = {0}", e.ProgressPercentage);
}
```

2.6.18 TXT2DBFCONVERTER.RUNWORKERCOMPLETEDDELEGATE

Raises the RunWorkerCompleted event.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public delegate void RunWorkerCompletedDelegate(Object sender,
RunWorkerCompletedEventArgs e)
```

Parameters

sender

Type: CipherLab.Data.ConverterLib.Txt2DbfConverter

The current class which can raise an event.

e

Type: System.ComponentModel.RunWorkerCompletedEventArgs

An EventArgs that contains the event data.

Example

```
using CipherLab.Data. ConverterLib;

Txt2DbfConverter test = new Txt2DbfConverter(cpdf,
"D:\\Task\\DataConverter3\\Test");
test.RunWorkerCompleted += new RunWorkerCompletedEventHandler(Complete);

public static void Complete(object sender, RunWorkerCompletedEventArgs e)
{
    if (e.Cancelled)
        Console.WriteLine("Canceled");
    else
    {
        if (e.Error != null)
            Console.WriteLine(e.Error.Message);
        else
            Console.WriteLine("Completed");
    }
}
```

2.6.19 TXT2PACKEDDBFCONVERTER.CANCELASYNC

Requests cancellation of conversion.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public void CancelAsync()
```

Example

```
using CipherLab.Data. ConverterLib;
```

```
Txt2PackedDbfConverter test = new Txt2PackedDbfConverter(cList,
"D:\\DCTester\\Input8k2");
test.CancelAsync();
```

2.6.20 TXT2PACKEDDBFCONVERTER.CONVERT

Requests action of conversion without callback request.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Example

```
using CipherLab.Data. ConverterLib;
Txt2PackedDbfConverter test = new Txt2PackedDbfConverter(cList,
"D:\\DCTester\\Input8k2");
test.Convert();
```

2.6.21 TXT2PACKEDDBFCONVERTER.CONVERT WITH CALLBACK

Requests action of conversion with callback request.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Example

```
using CipherLab.Data. ConverterLib;
Txt2PackedDbfConverter test = new Txt2PackedDbfConverter(cList,
"D:\\DCTester\\Input8k2");
test.Convert(callback);
```

2.6.22 TXT2PACKEDDBFCONVERTER.CONVERTASYNC

Requests action of conversion.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Example

```
using CipherLab.Data. ConverterLib;
Txt2PackedDbfConverter test = new Txt2PackedDbfConverter(cList,
"D:\\DCTester\\Input8k2");
test.ConvertAsync();
```

2.6.23 TXT2PACKEDDBFCONVERTER.PROGRESSCHANGEDDELEGATE

Raises the ProgressChanged event.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public delegate void ProgressChangedDelegate(Object sender,
ProgressChangedEventArgs e)
```

Parameters

sender

Type: CipherLab.Data.ConverterLib.Txt2PackedDbfConverter

The current class which can raise an event.

e

Type: System.ComponentModel.ProgressChangedEventArgs

An EventArgs that contains the event data.

Example

```
using CipherLab.Data. ConverterLib;
Txt2PackedDbfConverter test = new Txt2PackedDbfConverter(cList,
"D:\\DCTester\\Input8k2");
test.ProgressChanged += new ProgressChangedEventHandler(Report);
public static void Report(object sender, ProgressChangedEventArgs e)
```

```
{  
    Console.WriteLine("Current Percentage is = {0}", e.ProgressPercentage);  
}
```

2.6.24 TXT2PACKEDDBFCONVERTER.RUNWORKERCOMPLETEDDELEGATE

Raises the RunWorkerCompleted event.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public delegate void RunWorkerCompletedDelegate(Object sender,  
RunWorkerCompletedEventArgs e)
```

Parameters

sender

Type: CipherLab.Data.ConverterLib.Txt2PackedDbfConverter

The current class which can raise an event.

e

Type: System.ComponentModel.RunWorkerCompletedEventArgs

An EventArgs that contains the event data.

Example

```
using CipherLab.Data. ConverterLib;  
using CipherLab.Data. ConverterLib;  
  
Txt2PackedDbfConverter test = new Txt2PackedDbfConverter(cList,  
"D:\\DCTester\\Input8k2");  
test.RunWorkerCompleted += new RunWorkerCompletedEventHandler(Complete);  
  
public static void Complete(object sender, RunWorkerCompletedEventArgs e)
```

```
{  
    if (e.Cancelled)  
        Console.WriteLine("Canceled");  
    else  
    {  
        if (e.Error != null)  
            Console.WriteLine(e.Error.Message);  
        else  
            Console.WriteLine("Completed");  
    }  
}
```


2.7 CLASS EVENTS

2.7.1 DBF2TXTCONVERTER.PROGRESSCHANGED

The event occurs when users raise the ProgressChanged event.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public event ProgressChangedEventHandler ProgressChanged
```

Example

```
using CipherLab.Data. ConverterLib;  
Dbf2TxtConverter test = new Dbf2TxtConverter(ProgrammingTool.Basic, source, target);  
test.ProgressChanged += new ProgressChangedEventHandler(Report);
```

2.7.2 DBF2TXTCONVERTER.RUNWORKERCOMPLETED

The event occurs when the conversion has completed, has been canceled, or has raised an exception.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public event RunWorkerCompletedEventHandler RunWorkerCompleted
```

Example

```
using CipherLab.Data. ConverterLib;  
Dbf2TxtConverter test = new Dbf2TxtConverter(ProgrammingTool.Basic, source, target);  
test.RunWorkerCompleted += new RunWorkerCompletedEventHandler(Complete);
```

2.7.3 TXT2DBFCONVERTER.PROGRESSCHANGED

The event occurs when users raise the ProgressChanged event.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public event ProgressChangedEventHandler ProgressChanged
```

Example

```
using CipherLab.Data. ConverterLib;  
Txt2DbfConverter test = new Txt2DbfConverter(cpdf,  
"D:\\Task\\DataConverter3\\Test");  
test.ProgressChanged += new ProgressChangedEventHandler(Report);
```

2.7.4 TXT2DBFCONVERTER.RUNWORKERCOMPLETED

The event occurs when the conversion has completed, has been canceled, or has raised an exception.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public event RunWorkerCompletedEventHandler RunWorkerCompleted
```

Example

```
using CipherLab.Data. ConverterLib;  
Txt2DbfConverter test = new Txt2DbfConverter(cpdf,  
"D:\\Task\\DataConverter3\\Test");  
test.RunWorkerCompleted += new RunWorkerCompletedEventHandler(Complete);
```

2.7.5 TXT2PACKEDDBFCONVERTER.PROGRESSCHANGED

The event occurs when users raise the ProgressChanged event.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public event ProgressChangedEventHandler ProgressChanged
```

Example

```
using CipherLab.Data. ConverterLib;  
Txt2PackedDbfConverter test = new Txt2PackedDbfConverter(cList,  
"D:\\DCTester\\Input8k2");  
test.ProgressChanged += new ProgressChangedEventHandler(Report);
```

2.7.6 TXT2PACKEDDBFCONVERTER.RUNWORKERCOMPLETED

The event occurs when the conversion has completed, has been canceled, or has raised an exception.

Namespace: CipherLab.Data.ConverterLib

Assembly: CipherLab.Data.ConverterLib (in CipherLab.Data.ConverterLib.dll)

Syntax

```
public event RunWorkerCompletedEventHandler RunWorkerCompleted
```

Example

```
using CipherLab.Data. ConverterLib;  
Txt2PackedDbfConverter test = new Txt2PackedDbfConverter(cList,  
"D:\\DCTester\\Input8k2");  
test.RunWorkerCompleted += new RunWorkerCompletedEventHandler(Complete);
```