Dynamsoft

15 Years of Experience in TWAIN SDKs
and Version Control Solutions

Developer's Guide

Dynamsoft Barcode Reader

Table of Contents

Developer Guide for Linux Edition

Dynamsoft's Barcode Reader SDK enables you to efficiently embed barcode reading functionality in your web, desktop or mobile application using just a few lines of code. This can save you months of added development time and extra costs. With our SDK, you can create high-speed and reliable barcode scanner software to meet your business needs.

This guide will show you how to use Dynamsoft Barcode Reader Linux Edition. The core of Linux Edition is written in C and C++ for performance. The library is also wrapped for use with Java, Python, etc.

Supported Barcode Types

- 1D barcodes: Code 39, Code 93, Code 128, Codabar, ITF, EAN-13, EAN-8, UPC-A, UPC-E, INDUSTRIAL 2 OF 5
- 2D barcodes: QRCode, PDF417, DataMatrix, Aztec

Dynamsoft Barcode Reader Editions

Edition	Supported Programing Languages
Windows Edition	C, C++, C#, VB .NET, PHP, Java
Linux Edition	C, C++, Java
JavaScript Edition	JavaScript
iOS Edition	Swift and Objective-C
Android Edition	Java
Mac Edition (Labs project)	C, C++, Java
Raspberry Pi Edition (Labs project)	C, C++, Java

Installation

The Installation section provides information on:

System Requirements

Operating Systems

- Linux x64 (Ubuntu 14.04.4+ LTS, Debian 8+, etc.)

Note: If you want to support Linux arm, please check out our Raspberry Pi Edition

Languages and Environment

Dynamsoft Barcode Reader Linux Edition provides C, C++ and Java APIs.

API	Environment
C, C++	64-bit
Java	JDK 1.7

How to Obtain the Dynamsoft Barcode Reader SDK

To install Dynamsoft Barcode Reader Linux Edition on your development machine, you can download the SDK from Dynamsoft website and unpack the .tar.gz file.

After extracting, you can find samples for supported platforms in the **samples** folder.

Build a Hello World Application

Prerequisites

Before getting started, please make sure you've already installed Dynamsoft Barcode Reader SDK. If not, please refer to the Installation section to install the development kit on your machine.

The trial installer includes a 30-day free trial license by default that allows you to evaluate full features of the SDK. For more information, please refer to the Use a trial key section.

Start Coding

This section will show how to build a Hello World application that reads barcodes from a static image in different programming languages.

- Create a C app for barcode reading
- Create a C++ app for barcode reading
- Create a Java app for barcode reading
- Create a Python app for barcode reading
- Create a PHP app for barcode reading

Create a C app for barcode reading

To build a C application that reads barcodes from an image, you can follow the steps below:

1. Create a makefile with the following code.

```
CC=gcc
CCFLAGS=-c
DBRLIB_PATH=<relative path>/lib
LDFLAGS=-L $(DBRLIB_PATH) -Wl,-rpath=$(DBRLIB_PATH) -Wl,-rpath=./
DBRLIB=-lDynamsoftBarcodeReader
TARGET=ReadBarcode
OBJECT=ReadBarcode.o
SOURCE=ReadBarcode.c
# build rule for target.
$(TARGET): $(OBJECT)
$(CC) -o $(TARGET) $(OBJECT) $(STDLIB) $(DBRLIB) $(LDFLAGS)
# target to build an object file
$(OBJECT): $(SOURCE)
$(CC) $(CCFLAGS) $(SOURCE)
# the clean target
.PHONY : clean
clean:
rm -f $(OBJECT) $(TARGET)
```

   Note Please replace with your own relative path in the code.

2. Create an empty ReadBarcode.c file and copy the following code to the .c file.

```
#include <stdio.h>
#include "<relative path>/DynamsoftBarcodeReader.h"
//Please replace <relative path> with your own relative path in the code.
int main()
{
 // Define variables
 void *hBarcode = NULL;
 int iRet = -1;
 int iIndex = 0;
 int iLicMsg = -1;
 TextResultArray* pResult = NULL;
 hBarcode = DBR_CreateInstance();

 // Initialize license prior to any decoding
 iLicMsg = DBR_InitLicense(hBarcode, "<your license key here>");

 //If error occurs to the license initialization
  if (iLicMsg != DBR_OK)
 {
```

```
    printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, DBR_GetErrorString(iLicMsg));
    return iLicMsg;
}

// Started decoding
DBR_DecodeFile(hBarcode, "<your image file full path>", "");

// Get the text result
iRet = DBR_GetAllTextResults(hBarcode, &pResult);

// If error occurs
if (iRet != DBR_OK)
{
    printf("Failed to read barcode: %d\r\n%s\r\n", iRet, DBR_GetErrorString(iRet));
    return iRet;
}

// If succeeds
printf("%d total barcode(s) found. \n", pResult->resultsCount);
for (iIndex = 0; iIndex < pResult->resultsCount; iIndex++)
{
    printf("Result %d\n", iIndex + 1);
    printf("Barcode Format: %s\n", pResult->results[iIndex]->barcodeFormatString);
    printf("Text reads: %s \n", pResult->results[iIndex]->barcodeText);
}

// Finally release BarcodeResultArray and destroy instance
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
system("pause");
return 0;
}
```

3. Run the project.

   Open the Terminal, use command cd to change the current directory to the one where the makefile resides, use command make to build the executable program.

To deploy your application, make sure the DLLs are in the same folder as the EXE file. See the Distribution section for more details.

Create a C++ app for barcode reading

To build a C++ application that reads barcodes from an image, you can follow the steps below:

1. Create a makefile with the following code.

```
CC=gcc
CCFLAGS=-c
DBRLIB_PATH=<relative path>/lib
LDFLAGS=-L $(DBRLIB_PATH) -Wl,-rpath=$(DBRLIB_PATH) -Wl,-rpath=./
DBRLIB=-lDynamsoftBarcodeReader
STDLIB=-lstdc++
TARGET=ReadBarcode
OBJECT=ReadBarcode.o
SOURCE=ReadBarcode.cpp
# build rule for target.
$(TARGET): $(OBJECT)
$(CC) -o $(TARGET) $(OBJECT) $(STDLIB) $(DBRLIB) $(LDFLAGS)
# target to build an object file
$(OBJECT): $(SOURCE)
$(CC) $(CCFLAGS) $(SOURCE)
# the clean target
.PHONY : clean
clean:
rm -f $(OBJECT) $(TARGET)
```

   Note Please replace with your own relative path in the code.

2. Create an empty ReadBarcode.cpp file and copy the following code to the .cpp file.

```
#include <stdio.h>
#include "<relative path>/DynamsoftBarcodeReader.h"
int main()
```

```
{
 // Define variables
 int iRet = -1;
 int iLicMsg = -1;
 TextResultArray *paryResult = NULL;

 // Initialize license prior to any decoding
 CBarcodeReader reader;
 iLicMsg = reader.InitLicense("<your license key here>");

 //If error occurs to the license initialization
 if (iLicMsg != DBR_OK)
 {
     printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, DBR_GetErrorString(iLicMsg));
     return iLicMsg;
 }

 // Start decoding
 iRet = reader.DecodeFile("<your image file full path>", "");

 // If error occurs
 if (iRet != DBR_OK)
 {
     printf("Failed to read barcode: %d\r\n%s\r\n", iRet, DBR_GetErrorString(iRet));
     return iRet;
 }

 // If succeeds
 reader.GetAllTextResults(&paryResult);
 printf("%d total barcodes found. \r\n", paryResult->resultsCount);
 for (int iIndex = 0; iIndex < paryResult->resultsCount; iIndex++)
 {
     printf("Result %d\r\n", iIndex + 1);
     printf("BarcodeFormat: %s\r\n", paryResult->results[iIndex]->barcodeFormatString);
     printf("Text read: %s\r\n", paryResult->results[iIndex]->barcodeText);
 }

 // Finally release BarcodeResultArray
 CBarcodeReader::FreeTextResults(&paryResult);
 system("pause");
 return 0;
}
```

Note Please update the , and with valid values respectively in the code.

3. Run the project.

   Open the Terminal, use command cd to change the current directory to the one where the makefile resides, use command make to build the executable program.

To deploy your application, please make sure the DLLs are in the same folder as the EXE file. See the Distribution section for more details.

Create a Java app for barcode reading

To build a Java application that reads barcodes from an image, you can follow the steps below:

1. Open Eclipse and create a new Java project `HelloDBR` .
2. Add the required JAR file to your project.

   Click **File** > **Properties** > **Java Build Path** > **Libraries** > **Add external JARs**, add `DynamsoftBarcodeReader_{edition}.jar` and click **Apply**.

   The JAR file can be found at `[INSTALLATION FOLDER]\Components\Java\` .
3. Import the header.

```
import com.dynamsoft.barcode.*;
```

4. Replace the code of `HelloDBR` with the following.

   Please update `<your image file full path>` and `<your license key here>` in the code accordingly.

```
public class HelloDBR {
```

```
    public static void main(String[] args) {
        try {
            BarcodeReader dbr = new BarcodeReader();
            dbr.initLicense("<Put your license key here>");
            TextResult[] result = dbr.decodeFile("<your image file full path>", "");
            String output = "";
            for(int i =0; i<result.length;i++){
                output += "Barcode Text: ";
                output += result[i].barcodeText + "\n\n";
            }
            System.out.println(output);
        }
        catch(Exception ex) {
        }
    }
}
```

5. Run the project.

Create a Python app for barcode reading

You can easily create a Python extension with the C/C++ API of Dynamsoft Barcode Reader.

For more information on how to create a Python extension for barcode reading, please refer to DBR Python Extension .

Create a PHP app for barcode reading

Dynamsoft Barcode Reader provides a PHP wrapper for Linux. With it, you can easily implement barcode reading in your PHP web applications.

For more information on how to create a PHP application for barcode scanning, Please refer to Use Dynamsoft Barcode Reader to Build a Barcode Scanner in PHP on Linux.

Decoding Methods

The SDK provides multiple decoding methods that support reading barcodes from different sources, including static images, video stream, files in memory, base64 string, bitmap, etc. Here is a list of all decoding methods:

- DecodeFile: Reads barcodes from a specified image file (BMP, JPEG, PNG, GIF, or TIFF).
- DecodeBase64String: Reads barcodes from a base64 encoded string of a file.
- DecodeBitmap: Reads barcodes from a bitmap. When handling multi-page images, it will only decode the current page.
- DecodeBuffer: Reads barcodes from raw buffer.
- DecodeFileInMemory: Decodes barcodes from an image file in memory.

Code snippets are provided in C/C++ as examples below. Not all supported programming languages are covered in this guide. You can find more samples in more programming languages at Code Gallery or Github Repositories.

DecodeFile

Reads barcodes from a statc image file.

Code snippet in C:

```
void *hBarcode = NULL;
TextResultArray* pResult = NULL;
hBarcode = DBR_CreateInstance();
// Initialize license prior to any decoding
//Replace “Put your license key here” with your own license
DBR_InitLicense(hBarcode, "<Put your license key here>");
//Replace “Put the path of your file here” with your own file path
DBR_DecodeFile(hBarcode,"<Put your file path here>","");
DBR_GetAllTextResults(hBarcode, &pResult);
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
```

Code snippet in C++:

```
char sError[512];
TextResultArray* paryResult = NULL;
CBarcodeReader* reader = new CBarcodeReader();
//Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
reader->InitLicense("<Put your license key here>");
//Replace “Put the path of your file here” with your own file path
reader->DecodeFile("<Put your file path here>", "");
// If succeeds
reader->GetAllTextResults(&paryResult);
CBarcodeReader::FreeTextResults(&paryResult);
delete reader;
```

DecodeBase64String

Reads barcodes from the base64 encoded string of a file.

Code snippet in C:

```
void* hBarcode = DBR_CreateInstance();
int iRet = -1;
int iIndex = 0;
int iLicMsg = -1;
unsigned char* pBufferBytes;
int nFileSize = 0;
char* strBase64String;
TextResultArray* pResult = NULL;

// Replace "Put your license key here" with your own trial license
iLicMsg = DBR_InitLicense(hBarcode, "Put your license key here");

// If error occurs to the license
if (iLicMsg != DBR_OK) {
   printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, DBR_GetErrorString(iLicMsg));
```

```
    return iLicMsg;
}
// Replace "Put the path of your file here" with your own file path
GetFileStream("Put the path of your file here", &pBufferBytes, &nFileSize);
GetFileBase64String(pBufferBytes, &strBase64String);
DBR_DecodeBase64String(hBarcode, strBase64String, "");
iRet = DBR_GetAllTextResults(hBarcode, &pResult);
// If error occurs
if (iRet != DBR_OK) {
    printf("Failed to read barcode: %d\r\n%s\r\n", iRet, DBR_GetErrorString(iRet));
    return iRet;
}
printf("%d total barcode(s) found. \n", pResult->resultsCount);
for (iIndex = 0; iIndex < pResult->resultsCount; iIndex++)
{
    printf("Result %d\n", iIndex + 1);
    printf("Barcode Format: %s\n", pResult->results[iIndex]->barcodeFormatString);
    printf("Text reads: %s \n", pResult->results[iIndex]->barcodeText);
}
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
return 0;
```

Code snippet in C++:

```
int main()
{
    int iRet = -1;
    int iLicMsg = -1;
    char* strBase64String;
    TextResultArray* paryResult = NULL;
    PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
    CBarcodeReader* reader = new CBarcodeReader();
    // Initialize license prior to any decoding
    // Replace "Put your license key here" with your own license
    iLicMsg = reader->reader.InitLicense("Put your license key here ");
    // If error occurs to the license
    if (iLicMsg != DBR_OK)
    {
        printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, CBarcodeReader::GetErrorString(iLicMsg));
        return iLicMsg;
    }
    // Replace "Put the path of your file here" with your own file path
    GetFileStream("Put the path of your file here ", &pBufferBytes, &nFileSize);
    GetFileBase64String(pBufferBytes, &strBase64String);
    iRet = reader->DecodeBase64String(strBase64String, "");
    // If error occurs
    if (iRet != DBR_OK)
    {
        printf("Failed to read barcode: %d\r\n%s\r\n", iRet, CBarcodeReader::GetErrorString(iRet));
        return iRet;
    }
    // If succeeds
    reader->GetAllTextResults(&paryResult);
    printf("%d total barcodes found. \r\n", paryResult->resultsCount);
    for (int iIndex = 0; iIndex < paryResult->resultsCount; iIndex++)
    {
        printf("Result %d\r\n", iIndex + 1);
        printf("BarcodeFormat: %s\r\n", paryResult->results[iIndex]->barcodeFormatString);
        printf("Text read: %s\r\n", paryResult->results[iIndex]->barcodeText);
    }
    CBarcodeReader::FreeTextResults(&paryResult);
    delete runtimeSettings;
    delete reader;
    getchar();
    return 0;
}
```

DecodeBitmap

`DecodeBitmap()` reads barcodes from a bitmap. It will only decode the current page when handling multi-page images.

Code snippet in C:

```c
void* hBarcode = DBR_CreateInstance();
HANDLE pDIB;
int iRet = -1;
int iIndex = 0;
int iLicMsg = -1;
TextResultArray* pResult = NULL;
// Replace "Put your license key here" with your own license
iLicMsg = DBR_InitLicense(hBarcode, "Put your license key here");
// If error occurs to the license
if (iLicMsg != DBR_OK) {
   printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, DBR_GetErrorString(iLicMsg));
   return iLicMsg;
}
// Replace "Put the path of your file here" with your own file path
GetDIBFromImage("Put the path of your file here ", &pDIB);
DBR_DecodeDIB(hBarcode, pDIB, "");
iRet = DBR_GetAllTextResults(hBarcode, &pResult);
// If error occurs
if (iRet != DBR_OK) {
   printf("Failed to read barcode: %d\r\n%s\r\n", iRet, DBR_GetErrorString(iRet));
   return iRet;
}
printf("%d total barcode(s) found. \n", pResult->resultsCount);
for (iIndex = 0; iIndex < pResult->resultsCount; iIndex++)
{
   printf("Result %d\n", iIndex + 1);
   printf("Barcode Format: %s\n", pResult->results[iIndex]->barcodeFormatString);
   printf("Text reads: %s \n", pResult->results[iIndex]->barcodeText);
}
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
return 0;
```

Code snippet in C++:

```c
int main()
{
   int iRet = -1;
   int iLicMsg = -1;
   HANDLE pDIB;
   TextResultArray* paryResult = NULL;
   PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
   CBarcodeReader* reader = new CBarcodeReader();
   // Initialize license prior to any decoding
   //Replace "Put your license key here" with your own license
   iLicMsg = reader->reader.InitLicense("Put your license key here ");
   // If error occurs to the license
   if (iLicMsg != DBR_OK)
   {
      printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, CBarcodeReader::GetErrorString(iLicMsg));
      return iLicMsg;
   }
   // Replace "Put the path of your file here" with your own file path
   GetDIBFromImage("Put the path of your file here ", &pDIB);
   iRet = reader->DecodeDIB(pDIB, "");
   // If error occurs
   if (iRet != DBR_OK)
   {
      printf("Failed to read barcode: %d\r\n%s\r\n", iRet, CBarcodeReader::GetErrorString(iRet));
      return iRet;
   }
   // If succeeds
   reader->GetAllTextResults(&paryResult);
   printf("%d total barcodes found. \r\n", paryResult->resultsCount);
   for (int iIndex = 0; iIndex < paryResult->resultsCount; iIndex++)
   {
      printf("Result %d\r\n", iIndex + 1);
      printf("BarcodeFormat: %s\r\n", paryResult->results[iIndex]->barcodeFormatString);
      printf("Text read: %s\r\n", paryResult->results[iIndex]->barcodeText);
   }
   CBarcodeReader::FreeTextResults(&paryResult);
   delete runtimeSettings;
```

```
    delete reader;
    getchar();
    return 0;
}
```

DecodeBuffer

Reads barcodes from raw buffer.

Code snippet in C:

```
void* hBarcode = DBR_CreateInstance();
int iRet = -1;
int iIndex = 0;
int iLicMsg = -1;
unsigned char* pBufferBytes;
int iWidth = 0;
int iHeight = 0;
int iStride = 0;
ImagePixelFormat format;
TextResultArray* pResult = NULL;
// Replace "Put your license key here" with your own license
iLicMsg = DBR_InitLicense(hBarcode, "Put your license key here");
// If error occurs to the license
if (iLicMsg != DBR_OK) {
   printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, DBR_GetErrorString(iLicMsg));
   return iLicMsg;
}
// Replace "Put the path of your file here" with your own file path
GetBufferFromFile("Put the path of your file here ", &pBufferBytes, &iWidth, &iHeight, &iStride, &format);
DBR_DecodeBuffer(hBarcode, pBufferBytes, iWidth, iHeight, iStride, format, "");
iRet = DBR_GetAllTextResults(hBarcode, &pResult);
// If error occurs
if (iRet != DBR_OK) {
   printf("Failed to read barcode: %d\r\n%s\r\n", iRet, DBR_GetErrorString(iRet));
   return iRet;
}
printf("%d total barcode(s) found. \n", pResult->resultsCount);
for (iIndex = 0; iIndex < pResult->resultsCount; iIndex++)
{
   printf("Result %d\n", iIndex + 1);
   printf("Barcode Format: %s\n", pResult->results[iIndex]->barcodeFormatString);
   printf("Text reads: %s \n", pResult->results[iIndex]->barcodeText);
}
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
return 0;
```

Code snippet in C++:

```
int main()
{
   int iRet = -1;
   int iLicMsg = -1;
   int iWidth = 0;
   int iHeight = 0;
   int iStride = 0;
   unsigned char* pBufferBytes;
   ImagePixelFormat format;
   TextResultArray* paryResult = NULL;
   PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
   CBarcodeReader* reader = new CBarcodeReader();
   // Initialize license prior to any decoding
   // Replace "Put your license key here" with your own license
   iLicMsg = reader->reader.InitLicense("Put your license key here ");
   // If error occurs to the license
   if (iLicMsg != DBR_OK)
   {
      printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, CBarcodeReader::GetErrorString(iLicMsg));
      return iLicMsg;
   }
   // Replace "Put the path of your file here" with your own file path
```

```
    GetBufferFromFile("Put the path of your file here ", &pBufferBytes, &iWidth, &iHeight, &iStride, &format);
    iRet = reader->DecodeBuffer(pBufferBytes, iWidth, iHeight, iStride, format, "");
    if (iRet != DBR_OK)
    {
       printf("Failed to read barcode: %d\r\n%s\r\n", iRet, CBarcodeReader::GetErrorString(iRet));
       return iRet;
    }
    // If succeeds
    reader->GetAllTextResults(&paryResult);
    printf("%d total barcodes found. \r\n", paryResult->resultsCount);
    for (int iIndex = 0; iIndex < paryResult->resultsCount; iIndex++)
    {
       printf("Result %d\r\n", iIndex + 1);
       printf("BarcodeFormat: %s\r\n", paryResult->results[iIndex]->barcodeFormatString);
       printf("Text read: %s\r\n", paryResult->results[iIndex]->barcodeText);
    }
    CBarcodeReader::FreeTextResults(&paryResult);
    delete runtimeSettings;
    delete reader;
    getchar();
    return 0;
}
```

DecodeFileInMemory

Decodes barcodes from an image file in memory.

Code snippet in C:

```
void* hBarcode = DBR_CreateInstance();
int iRet = -1;
int iIndex = 0;
int iLicMsg = -1;
unsigned char* pFileBytes;
int nFileSize = 0;
TextResultArray* pResult = NULL;
// Replace "Put your license key here" with your own license
iLicMsg = DBR_InitLicense(hBarcode, "Put your license key here");
// If error occurs to the license
if (iLicMsg != DBR_OK) {
   printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, DBR_GetErrorString(iLicMsg));
   return iLicMsg;
}
// Replace "Put the path of your file here" with your own file path
GetFileStream("Put the path of your file here ", &pFileBytes, &nFileSize);
DBR_DecodeFileInMemory(hBarcode, pFileBytes, nFileSize, "");

iRet = DBR_GetAllTextResults(hBarcode, &pResult);

// If error occurs
if (iRet != DBR_OK) {
   printf("Failed to read barcode: %d\r\n%s\r\n", iRet, DBR_GetErrorString(iRet));
   return iRet;
}
printf("%d total barcode(s) found. \n", pResult->resultsCount);
for (iIndex = 0; iIndex < pResult->resultsCount; iIndex++)
{
   printf("Result %d\n", iIndex + 1);
   printf("Barcode Format: %s\n", pResult->results[iIndex]->barcodeFormatString);
   printf("Text reads: %s \n", pResult->results[iIndex]->barcodeText);
}
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
return 0;
```

Code snippet in C++:

```
int main()
{
   int iRet = -1;
   int iLicMsg = -1;
   int nFileSize = 0;
```

```
    unsigned char* pFileBytes;
    TextResultArray* paryResult = NULL;
    PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
    CBarcodeReader* reader = new CBarcodeReader();
    // Initialize license prior to any decoding
    // Replace "Put your license key here" with your own license
    iLicMsg = reader->InitLicense("Put your license key here ");
    // Replace "Put the path of your file here" with your own file path
    GetFileStream("Put the path of your file here ", &pFileBytes, &nFileSize);
    iRet = reader->DecodeFileInMemory(pFileBytes, nFileSize, "");
    // If error occurs
    if (iRet != DBR_OK)
    {
       printf("Failed to read barcode: %d\r\n%s\r\n", iRet, CBarcodeReader::GetErrorString(iRet));
       return iRet;
    }
    // If succeeds
    reader->GetAllTextResults(&paryResult);
    printf("%d total barcodes found. \r\n", paryResult->resultsCount);
    for (int iIndex = 0; iIndex < paryResult->resultsCount; iIndex++)
    {
       printf("Result %d\r\n", iIndex + 1);
       printf("BarcodeFormat: %s\r\n", paryResult->results[iIndex]->barcodeFormatString);
       printf("Text read: %s\r\n", paryResult->results[iIndex]->barcodeText);
    }
    CBarcodeReader::FreeTextResults(&paryResult);
    delete runtimeSettings;
    delete reader;
    getchar();
    return 0;
}
```

Barcode Reading Settings

Calling the decoding methods directly will use the default scanning modes and it will satisfy most of the needs. The SDK also allows you to adjust the scanning settings to optimize the scanning performance for different usage scenarios.

There are two ways to change the barcode reading settings - using the `PublicRuntimeSettings` Struct or template. For new developers, We recommend you to start with the `PublicRuntimeSettings` struct; For those who are experienced with the SDK, you may use a template which is more flexible and easier to update.

- Use PublicRuntimeSettings Struct to Change Settings
- Use A Template to Change Settings

Use PublicRuntimeSettings Struct to Change Settings

Here are some common scanning settings you might find helpful:

- Specify Barcode Type to Read
- Specify Maximum Barcode Count
- Specify a Scan Region

For more scanning settings guide, check out the How To section.

Learn more about PublicRuntimeSettings Struct.

Specify Which Barcode Type to Read

By default, the SDK will read all the supported barcode formats from the image. (See Product Overview for the full supported barcode list.)

If your full license only covers some barcode formats, you can use `barcodeFormatIds` to specify the barcode format(s).

For example, to enable only 1D barcode reading, you can use the following code:

Code snippet in C:

```
void *hBarcode = NULL;
char sError[512];
TextResultArray* pResult = NULL;
PublicRuntimeSettings runtimeSettings;
hBarcode = DBR_CreateInstance();

// Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
DBR_InitLicense(hBarcode, "<Put your license key here>");
//Set the barcode format
DBR_GetRuntimeSettings(hBarcode, &runtimeSettings);
runtimeSettings.barcodeFormatIds = 2047; // OneD barcode
DBR_UpdateRuntimeSettings(hBarcode, &runtimeSettings,sError,512);

//Replace "Put the path of your file here" with your own file path
DBR_DecodeFile(hBarcode,"<Put your file path here>","");
DBR_GetAllTextResults(hBarcode, &pResult);
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
```

Code snippet in C++:

```
char sError[512];
TextResultArray* paryResult = NULL;
PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
CBarcodeReader* reader = new CBarcodeReader();
//Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
reader->InitLicense("<Put your license key here>");

//Set the barcode format
reader->GetRuntimeSettings(runtimeSettings);
runtimeSettings->barcodeFormatIds = 2047; //OneD barcode
reader->UpdateRuntimeSettings(runtimeSettings, sError, 512);

//Replace "<Put the path of your file here>" with your own file path
reader->DecodeFile("<Put your file path here>", "");
```

```
// If succeeds
reader->GetAllTextResults(&paryResult);
CBarcodeReader::FreeTextResults(&paryResult);
delete runtimeSettings;
delete reader;
```

Specify Maximum Barcode Count

By default, the SDK will read as many barcodes as it can. To increase the recognition efficiency, you can use `expectedBarcodesCount` to specify the maximum number of barcodes to recognize according to your scenario.

Code snippet in C:

```
void *hBarcode = NULL;
char sError[512];
TextResultArray* pResult = NULL;
PublicRuntimeSettings runtimeSettings;
hBarcode = DBR_CreateInstance();
// Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
DBR_InitLicense(hBarcode, "<Put your license key here>");

//Set the number of barcodes to be expected
DBR_GetRuntimeSettings(hBarcode, &runtimeSettings);
runtimeSettings.expectedBarcodesCount = 1;
DBR_UpdateRuntimeSettings(hBarcode, &runtimeSettings, sError, 512);

//Replace "<Put the path of your file here>" with your own file path
DBR_DecodeFile(hBarcode,"<Put your file path here>","");
DBR_GetAllTextResults(hBarcode, &pResult);
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
```

Code snippet in C++:

```
char sError[512];
TextResultArray* paryResult = NULL;
PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
CBarcodeReader* reader = new CBarcodeReader();
//Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
reader->InitLicense("<Put your license key here>");

//Set the number of barcodes to be expected
reader->GetRuntimeSettings(runtimeSettings);
runtimeSettings->expectedBarcodesCount = 1;
reader->UpdateRuntimeSettings(runtimeSettings, sError, 512);

//Replace "Put the path of your file here" with your own file path
reader->DecodeFile("<Put your file path here>", "");
// If succeeds
reader->GetAllTextResults(&paryResult);
CBarcodeReader::FreeTextResults(&paryResult);
delete runtimeSettings;
delete reader;
```

Specify a Scan Region

By default, the barcode reader will search the whole image for barcodes. This can lead to poor performance especially when dealing with high-resolution images. You can speed up the recognition process by restricting the scanning region.

To specify a region, you will need to define an area. The following code shows how to create a template string and define the region.

Code snippet in C:

```
void *hBarcode = NULL;
char sError[512];
TextResultArray* pResult = NULL;
PublicRuntimeSettings runtimeSettings;
hBarcode = DBR_CreateInstance();
// Initialize license prior to any decoding
```

```
//Replace "<Put your license key here>" with your own license
DBR_InitLicense(hBarcode, "<Put your license key here>");

//Decode the barcodes on the left half of the image
DBR_GetRuntimeSettings(hBarcode, &runtimeSettings);
runtimeSettings.region.regionBottom = 100;
runtimeSettings.region.regionLeft = 0;
runtimeSettings.region.regionRight = 50;
runtimeSettings.region.regionTop = 0;
runtimeSettings.region.regionMeasuredByPercentage = 1; //The region is determined by percentage
DBR_UpdateRuntimeSettings(hBarcode, &runtimeSettings,sError,512);

//Replace "<Put the path of your file here>" with your own file path
DBR_DecodeFile(hBarcode,"put your file path here","");
DBR_GetAllTextResults(hBarcode, &pResult);
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
```

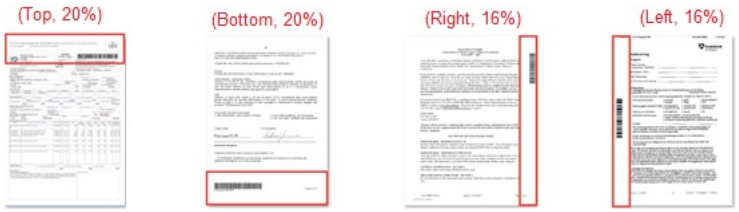
Code snippet in C++:

```
char sError[512];
TextResultArray* paryResult = NULL;
PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
CBarcodeReader* reader = new CBarcodeReader();
//Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
reader->InitLicense("Put your license key here");

//Decode the barcodes on the left of the image
reader->GetRuntimeSettings(runtimeSettings);
runtimeSettings->region.regionBottom = 100;
runtimeSettings->region.regionLeft = 0;
runtimeSettings->region.regionRight = 50;
runtimeSettings->region.regionTop = 0;
runtimeSettings->region.regionMeasuredByPercentage = 1; //The region is determined by percentage
reader->UpdateRuntimeSettings(runtimeSettings, sError, 512);

//Replace "<Put the path of your file here>" with your own file path
reader->DecodeFile("<Put your file path here>", "");
// If succeeds
reader->GetAllTextResults(&paryResult);
CBarcodeReader::FreeTextResults(&paryResult);
delete runtimeSettings;
delete reader;
```

For example:



Use A Template to Change Settings

Besides the option of using the `PublicRuntimeSettings` struct, the SDK also provides `InitRuntimeSettingsWithString()` and `InitRuntimeSettingsWithFile()` APIs that enable you to use a template to control all the runtime settings. With a template, instead of writing many codes to modify the settings, you can manage all the runtime settings in a JSON file/string.

Code snippet in C:

```
void *hBarcode = NULL;
char sError[512];
TextResultArray* pResult = NULL;
hBarcode = DBR_CreateInstance();
// Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
DBR_InitLicense(hBarcode, "<Put your license key here>");
```

```
//Use a template to modify the runtime settings
//DBR_InitRuntimeSettingsWithString() can also be used to modify the runtime settings with a json string
DBR_InitRuntimeSettingsWithFile(hBarcode, "<Put your file path here>", CM_OVERWRITE, sError, 512);

//Output runtime settings to a json file.
//DBR_OutputLicenseToString() can also be used to output the settings to a string
DBR_OutputSettingsToFile(hBarcode, "<Put your file path here>", "runtimeSettings");

//Replace "<Put your file path here>" with your own file path
DBR_DecodeFile(hBarcode,"put your file path here","");
DBR_GetAllTextResults(hBarcode, &pResult);
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
```

Code snippet in C++:

```
char sError[512];
TextResultArray* paryResult = NULL;
CBarcodeReader* reader = new CBarcodeReader();
//Initialize license prior to any decoding
//Replace "Put your license key here" with your own license
reader->InitLicense("Put your license key here");

//Use a template to modify the runtime settings
//InitRuntimeSettingsWithString() can also be used to modify the runtime settings with a json string
reader->InitRuntimeSettingsWithFile("<Put your file path here>", CM_OVERWRITE, sError, 512);

//Output runtime settings to a json file.
//OutputSettingsToString() can also be used to output the settings to a string
reader->OutputSettingsToFile("<Put your file path here>","currentruntime");

//Replace "Put the path of your file here" with your own file path
reader->DecodeFile("Put your file path here", "");
// If succeeds
reader->GetAllTextResults(&paryResult);
CBarcodeReader::FreeTextResults(&paryResult);
delete reader;
```

Here is a template for your reference:

```
{
   "ImageParameter" : {
      "BarcodeFormatIds" : [ "BF_ALL" ],
      "BinarizationModes" : [
         {
            "BlockSizeX" : 0,
            "BlockSizeY" : 0,
            "EnableFillBinaryVacancy" : 1,
            "ImagePreprocessingModesIndex" : -1,
            "Mode" : "BM_LOCAL_BLOCK",
            "ThreshValueCoefficient" : 10
         }
      ],
      "DeblurLevel" : 9,
      "Description" : "",
      "ExpectedBarcodesCount" : 0,
      "GrayscaleTransformationModes" : [
         {
            "Mode" : "GTM_ORIGINAL"
         }
      ],
      "ImagePreprocessingModes" : [
         {
            "Mode" : "IPM_GENERAL"
         }
      ],
      "IntermediateResultSavingMode" : {
         "Mode" : "IRSM_MEMORY"
      },
      "IntermediateResultTypes" : [ "IRT_NO_RESULT" ],
```

```
      "MaxAlgorithmThreadCount" : 4,
      "Name" : "runtimesettings",
      "PDFRasterDPI" : 300,
      "Pages" : "",
      "RegionDefinitionNameArray" : null,
      "RegionPredetectionModes" : [
         {
            "Mode" : "RPM_GENERAL"
         }
      ],
      "ResultCoordinateType" : "RCT_PIXEL",
      "ScaleDownThreshold" : 2300,
      "TerminatePhase" : "TP_BARCODE_RECOGNIZED",
      "TextFilterModes" : [
         {
            "MinImageDimension" : 65536,
            "Mode" : "TFM_GENERAL_CONTOUR",
            "Sensitivity" : 0
         }
      ],
      "TextResultOrderModes" : [
         {
            "Mode" : "TROM_CONFIDENCE"
         },
         {
            "Mode" : "TROM_POSITION"
         },
         {
            "Mode" : "TROM_FORMAT"
         }
      ],
      "TextureDetectionModes" : [
         {
            "Mode" : "TDM_GENERAL_WIDTH_CONCENTRATION",
            "Sensitivity" : 5
         }
      ],
      "Timeout" : 10000
   },
   "Version" : "3.0"
}
```

How-To Guides

This section covers the following topics:

- Read Barcodes from Camera Stream
- Read Barcodes with Different Colours
- Filter out Unwanted Barcode Results
- Get Intermediate Results
- Turn on or off Text Filter
- Scan in Multiple Threads

How to Read Barcodes from Camera Stream

Since v7.0, Dynamsoft Barcode Reader SDK added `StartFrameDecoding()` and `StopFrameDecoding()` APIs. With the APIs and OpenCV library, you can easily read barcodes from video stream.

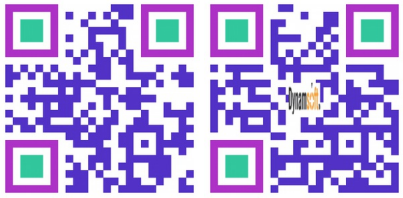
Get a sample: C++ Decode Video Frame >

Note: The sample is for Windows Edition, but the C++ code is the same as the Linux Edition. You can refer to the source code for reference and build your own application.

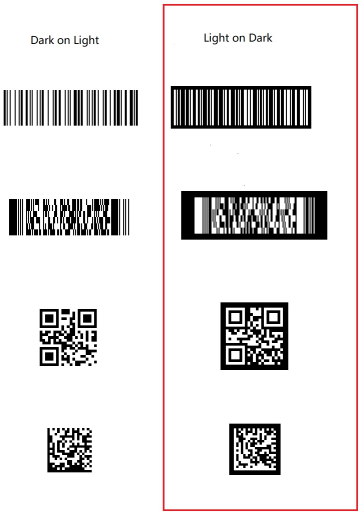
To learn more about the `StartFrameDecoding()` and `StopFrameDecoding()` APIs, see C++ API Reference.

How to Read Barcodes with Different Colors

Common barcodes are black and white. However, some barcodes could be in different colors such as the below left barcode. Also, in some cases, a QR code may appear with an image or a logo inside as shown below right. Dynamsoft can decode such special barcodes as well.



In some other cases, the barcodes can be white with a dark background color as shown on the right part of the image below:



To decode such kind of barcodes, the default runtime settings may fail. You can adjust the `PublicRuntimeSettings` like below:

Code snippet in C:

```
void *hBarcode = NULL;
char sError[512];
TextResultArray* pResult = NULL;
PublicRuntimeSettings runtimeSettings;
hBarcode = DBR_CreateInstance();
// Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
DBR_InitLicense(hBarcode, "<Put your license key here>");

//Change the runtime settings to read barcodes with inverted color
DBR_GetRuntimeSettings(hBarcode, &runtimeSettings);
runtimeSettings.furtherModes.grayscaleTransformationModes[0] = GTM_INVERTED;
DBR_UpdateRuntimeSettings(hBarcode, &runtimeSettings,sError,512);

//Replace "<Put the path of your file here>" with your own file path
DBR_DecodeFile(hBarcode,"<Put your file path here>","");
DBR_GetAllTextResults(hBarcode, &pResult);
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
```

Code snippet in C++

```
char sError[512];
TextResultArray* paryResult = NULL;
PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
CBarcodeReader* reader = new CBarcodeReader();
//Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
reader->InitLicense("<Put your license key here>");
```

```
//Change the runtime settings to read barcodes with inverted color
reader->GetRuntimeSettings(runtimeSettings);
runtimeSettings->furtherModes.grayscaleTransformationModes[0] = GTM_INVERTED;
reader->UpdateRuntimeSettings(runtimeSettings, sError, 512);

//Replace "<Put your file path here>" with your own file path
reader->DecodeFile("<Put your file path here>", "");
// If succeeds
reader->GetAllTextResults(&paryResult);
CBarcodeReader::FreeTextResults(&paryResult);
delete runtimeSettings;
delete reader;
```

How to Filter out Unwanted Barcode Results

Dynamsoft Barcode Reader SDK is able to read multiple barcodes at once and return results of all the decoded barcodes. However, you may not want all the results. For example, you may need only the results of a specific barcode format, or you may need only the barcodes with a certain length (number of barcode data bytes). The SDK provides settings to help you filter out the barcode results by barcode formats, confidence and text length.

Filtering out the barcode results based on the barcode format is shown in Specify Which Barcode Type to Read.

The following code shows how to filter out the unwanted barcode results based on the barcode confidence and barcode text length.

- Filtering using Barcode Confidence
- Filtering using Barcode Result Length

Filtering using Barcode Confidence

The barcode confidence means the probablity that the barcode result is reconigzed correctly. You can use `minResultConfidence` to filter out the results with low confidence.

Code snippet in C:

```
void *hBarcode = NULL;
char sError[512];
TextResultArray* pResult = NULL;
PublicRuntimeSettings runtimeSettings;
hBarcode = DBR_CreateInstance();
// Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
DBR_InitLicense(hBarcode, "<Put your license key here>");

// Obtain the barcode results with confidence above 35
DBR_GetRuntimeSettings(hBarcode, &runtimeSettings);
runtimeSettings.minResultConfidence = 35;  //It is recommended to set the confidence above 35
DBR_UpdateRuntimeSettings(hBarcode, &runtimeSettings,sError,512);

// Replace "<Put the path of your file here>" with your own file path
DBR_DecodeFile(hBarcode,"<Put your file path here>","");
DBR_GetAllTextResults(hBarcode, &pResult);
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
```

Code snippet in C++

```
char sError[512];
TextResultArray* paryResult = NULL;
PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
CBarcodeReader* reader = new CBarcodeReader();
// Initialize license prior to any decoding
// Replace "<Put your license key here>" with your own license
reader->InitLicense("<Put your license key here>");

// Obtain the barcode results with confidence above 35
reader->GetRuntimeSettings(runtimeSettings);
runtimeSettings->minResultConfidence = 35;    //It is recommended to set the confidence above 35
reader->UpdateRuntimeSettings(runtimeSettings, sError, 512);

// Replace "<Put your file path here>" with your own file path
reader->DecodeFile("<Put your file path here>", "");
// If succeeds
reader->GetAllTextResults(&paryResult);
CBarcodeReader::FreeTextResults(&paryResult);
delete runtimeSettings;
delete reader;
```

Filtering using Barcode Result Length

The barcode length is calculated in bytes. The length of the barcode text does not necessarily mean the actual length of the barcode bytes. You can use `minBarcodeTextLength` to filter out some invalid or unwanted results.

Code snippet in C:

```c
void *hBarcode = NULL;
char sError[512];
TextResultArray* pResult = NULL;
PublicRuntimeSettings runtimeSettings;
hBarcode = DBR_CreateInstance();
// Initialize license prior to any decoding
// Replace "<Put your license key here>" with your own license
DBR_InitLicense(hBarcode, "<Put your license key here>");

// Decode the barcodes with its length longer than 10 bytes
DBR_GetRuntimeSettings(hBarcode, &runtimeSettings);
runtimeSettings.minBarcodeTextLength = 10; //The length is calculated in bytes
DBR_UpdateRuntimeSettings(hBarcode, &runtimeSettings,sError,512);

// Replace "<Put your file path here>" with your own file path
DBR_DecodeFile(hBarcode,"put your file path here","");
DBR_GetAllTextResults(hBarcode, &pResult);
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
```

Code snippet in C++:

```cpp
char sError[512];
TextResultArray* paryResult = NULL;
PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
CBarcodeReader* reader = new CBarcodeReader();
// Initialize license prior to any decoding
// Replace "Put your license key here" with your own license
reader->InitLicense("Put your license key here");

// Decode the barcodes with its length longer than 10 bytes
reader->GetRuntimeSettings(runtimeSettings);
runtimeSettings->minBarcodeTextLength = 10;
reader->UpdateRuntimeSettings(runtimeSettings, sError, 512);

// Replace "Put the path of your file here" with your own file path
reader->DecodeFile("Put your file path here", "");
// If succeeds
reader->GetAllTextResults(&paryResult);
CBarcodeReader::FreeTextResults(&paryResult);
delete runtimeSettings;
delete reader;
```

How to Get Intermediate Results

Besides getting the results like barcode format, value, location, Dynamsoft Barcode Reader also provides APIs for you to get the intermediate results like original image, transformed grayscale image, binarized image, text zone, etc. for further analysis.

To learn more about what intermediate results you can get, see enum IntermediateResultType.

Here we take the original image as example and show how to get the original image and save it to your file system.

```
    int iRet = -1;
    hBarcode = DBR_CreateInstance();
    DBR_InitLicense(hBarcode, "t0068MgAAAGotN***"); // Set the trial license

    DBR_GetRuntimeSettings(hBarcode,&runtimeSettings);
    runtimeSettings.intermediateResultTypes = IRT_ORIGINAL_IMAGE;
    // Here we save it to the file system. You can also save it to memory according to your needs.
    runtimeSettings.intermediateResultSavingMode = IRSM_FILESYSTEM;
    iRet = DBR_UpdateRuntimeSettings(hBarcode,&runtimeSettings,szErrorMsg,256);
    // Set the folder path which stores the intermediate result (original image). Please make sure you have write permission to this
folder.
    iRet = DBR_SetModeArgument(hBarcode, "IntermediateResultSavingMode", 0, "FolderPath", "D:\DBRLogs", szErrorMsg, 256);
    if (iRet != DBR_OK) // If error occurs
    {
        printf("Error code: %d. Error message: %s\n", iRet, szErrorMsg);
        return -1;
    }
    DBR_DecodeFile(hBarcode, "<Your Image Path here>","");
    IntermediateResultArray* pResults;
    DBR_GetIntermediateResults(hBarcode, &pResults);

    DBR_DestroyInstance(hBarcode);
```

How to Turn On or Off Text Filter

In some cases, the image may be filled with a lot of characters and numbers. This could affect the barcode recognition. With text filter on, it can filter and remove the characters and numbers from the image and improve the barcode recognition accuracy. However, you may want to turn off text filter when barcodes are the only content of the image or when the speed is the prority.

The text filter is on by default.

The following code snippet shows how to set text filter function by changing the `TextFilterModes` parameter.

Code snippet in C:

```
void *hBarcode = NULL;
char sError[512];
TextResultArray* pResult = NULL;
PublicRuntimeSettings runtimeSettings;
hBarcode = DBR_CreateInstance();
// Initialize license prior to any decoding
//Replace "<Put your license key here>" with your own license
DBR_InitLicense(hBarcode, "<Put your license key here>");

DBR_GetRuntimeSettings(hBarcode, &runtimeSettings);
// Turn off the text filter function. The enumeration includes TFM_GENERAL_CONTOUR,
// which is used to turn on the function.
runtimeSettings.furtherModes.textFilterModes[0] = TFM_SKIP;
DBR_UpdateRuntimeSettings(hBarcode, &runtimeSettings,sError,512);

// Replace "<Put your file path here>" with your own file path
DBR_DecodeFile(hBarcode,"put your file path here","");
DBR_GetAllTextResults(hBarcode, &pResult);
DBR_FreeTextResults(&pResult);
DBR_DestroyInstance(hBarcode);
```

Code snippet in C++:

```
char sError[512];
TextResultArray* paryResult = NULL;
PublicRuntimeSettings* runtimeSettings = new PublicRuntimeSettings();
CBarcodeReader* reader = new CBarcodeReader();
// Initialize license prior to any decoding
// Replace "Put your license key here" with your own license
reader->InitLicense("Put your license key here");

reader->GetRuntimeSettings(runtimeSettings);
// Turn off the text filter function. The enumeration includes TFM_GENERAL_CONTOUR,
// which is used to turn on the function, and TFM_AUTO.
runtimeSettings->furtherModes.textFilterModes[0] = TFM_SKIP;
reader->UpdateRuntimeSettings(runtimeSettings, sError, 512);

//Replace "Put the path of your file here" with your own file path
reader->DecodeFile("Put your file path here", "");
// If succeeds
reader->GetAllTextResults(&paryResult);
CBarcodeReader::FreeTextResults(&paryResult);
delete runtimeSettings;
delete reader;
```

How to Scan in Multiple Threads

To scan barcodes in multiple threads using Dynamsoft Barcode Reader, you need to create multiple instances of `BarcodeReader` and run separate instance in each thread. Dynamsoft Barcode Reader SDK is non-thread safe. Please don't have multiple threads access the same `BarcodeReader` object.

We have a sample that shows how to use multiple threads to read barcodes from images using the C API.

Get the sample: Multi-thread scanning in C >

Note: The sample is for Windows Edition, but the C code is the same as the Linux Edition. You can refer to the source code for reference and build your own application.

Licensing and Distributing

This section covers the following topics:

Use a Trial Key

1. Get a trial license key.

   You can log in the customer portal and request for a trial extension online.

   Note: If the trial license expires or it is missing, you can still get barcode reading results but partial of the result will be masked with "".*

2. Update the license key in source code.

   You can use `initLicense()` or `ProductKeys` to set the license.

   Code snippet in C:

   ```
   void *hBarcode = NULL;
   hBarcode = DBR_CreateInstance();
   DBR_InitLicense(hBarcode, "t0068NQAAAI8+mMcYRNwmijAzExhq******");
   DBR_DestroyInstance(hBarcode);
   ```

   Code snippet in C++:

   ```
   CBarcodeReader reader = new CBarcodeReader();
   reader.InitLicense("t0068NQAAAI8+mMcYRNwmijAzExhq******");
   ```

   Code snippet in Java:

   ```
   BarcodeReader mBarcodeReader;
   mBarcodeReader = new BarcodeReader("t0068NQAAAI8+mMcYRNwmijAzExhq******");
   ```

   Code snippet in Python:

   ```
   def initLicense(license):
       dbr.initLicense(license)

   initLicense("t0068NQAAAI8+mMcYRNwmijAzExhq******")
   ```

3. Save and rebuild your application.

Use a Development License

Different methods are used for setting trial and full license keys. In the demo or sample applications, we use `.InitLicense()` or `.ProductKeys` to set trial license keys. For the purchased version, you need to use `initLicenseFromServer()` or `initLicenseFromLicenseContent()` to complete the license registration.

You can use a development license by following the steps below:

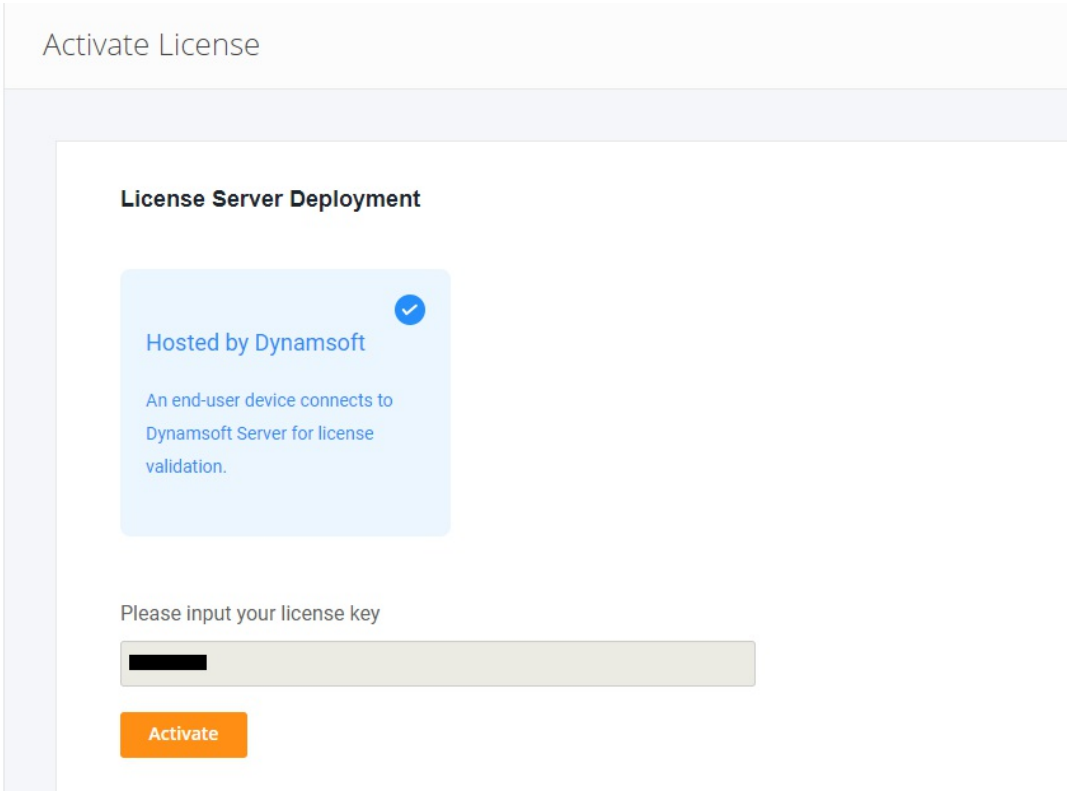
1. Activate a development license
2. Register the development license key

Activate a development license

Once you obtain a Development License, you can find your license information at **Customer Portal** > **License Center** > **Barcode Reader SDK**.

To activate a development license (8-digit key), click the **Activate Now** link beside it.

On the following page, click the **Activate** button.



Then you can see the status, quota, and expiration date of the activated license key.

You can repeat the above steps to activate other license keys.

Register the development license key

Here are the possible options to register a development license:

- Connect to Dynamsoft server once and use the SDK offline
- Always connect to Dynamsoft server for license verification
- No Internet connection

Connect to Dynamsoft server once and use the SDK offline

If you wish to use the SDK offline after activating, please follow the steps below:

1. Use `initLicenseFromServer()` to connect to Dynamsoft Hosted server (or your own server) to obtain the license information. The machine needs Internet connection to complete the device registration for the first time.
2. Use `outputLicenseToString()` to get the information of the license and store the information to the development machine.
3. Use `initLicenseFromLicenseContent()` API to register the license.

Code snippet in C:

```
void* _br = NULL;
int iLicMsg = -1;
char info[512];
_br = DBR_CreateInstance();
FILE *file;
// Check if there is a license file on the local machine. If yes, use the local license file; Otherwise, connect to Dynamsoft Ho
sted server to verify the license.
if ((file = fopen("license.txt", "r")) == NULL)
{
  // Connect to the Dynamsoft server to verify the license
  iLicMsg = DBR_InitLicenseFromServer(_br, "", "licenseKey1;licenseKey2");
  // The second parameter is the IP of the license server. Leaving it  empty ("") means it will connect to Dynamsoft License Ser
ver for online  verification automatically.
  // If error occurs to the license
  if (iLicMsg != DBR_OK) {
     printf("Failed to initialize the license: %d\r\n%s\r\n", iLicMsg, DBR_GetErrorString(iLicMsg));
     return iLicMsg;
  }
  DBR_OutputLicenseToString(_br, info, 512);
  // If you wish to use SDK offline, store the license information in txt or other format
  FILE *fp = fopen("license.txt", "w");
  if (fp == 0){
     printf("can't open file\n");
     return 0;
  }
  fwrite(info, sizeof(char) * 512, 1, fp);
  fclose(fp);
}
else{
  // Use the local license file and use Dynamsoft Barcode Reader SDK offline
  FILE *fp = fopen("license.txt", "r");
  fscanf(fp, "%s", &info);
  fclose(fp);
  iLicenMsg = DBR_InitLicenseFromLicenseContent(_br, "licenseKey1;licenseKey2",  info);
  // If error occurs to the license
  if (iLicMsg != DBR_OK) {
     printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, DBR_GetErrorString(iLicMsg));
     return iLicMsg;
  }
}

// Barcode decoding happens here
//....
DBR_DestroyInstance(_br);
```

Code snippet in C++:

```
CBarcodeReader* reader = new CBarcodeReader();
int iLicMsg = -1;
char info[512];
string filePath= "license.txt";
// To be able to use the license key offline, you need to store the license file obtained from Dynamsoft server once you use the
 API, InitLicenseFromServer.
fstream licenseFile;
licenseFile.open(filePath, ios::in);
// Check if there is a license file on the local machine. If not, connect to Dynamsoft Hosted server to verify the license. Othe
rwise, use the local license file.
if (!licenseFile)
{
   // Connect to Dynamsoft server to verify the license.
```

```
    iLicMsg = reader->InitLicenseFromServer("", "licenseKey1;licenseKey2");
    // The first parameter is the IP of the license server. Leaving it empty ("") means it will connect to Dynamsoft License Server for online verification automatically.

    // If error occurs to the license
    if (iLicMsg != DBR_OK)
    {
        printf("Failed to initialize the license: %d\r\n%s\r\n", iLicMsg, CBarcodeReader::GetErrorString(iLicMsg));
        return iLicMsg;
    }
    // If you wish to use SDK offline, store the license information in TXT or other format
    reader->OutputLicenseToString(info, 512);
    ofstream licFileOut(filePath);
    licFileOut << info;
    licFileOut.close();
}
else
{
    // Use the local license file and use Dynamsoft Barcode Reader SDK offline
    ifstream licFileIn(filePath);
    licFileIn >> info;
    licFileIn.close();
    iLicMsg = reader->InitLicenseFromLicenseContent("licenseKey1;licenseKey2", info);
    // If error occurs to the license
    if (iLicMsg != DBR_OK)
    {
        printf("Failed to initialize the license successfully: %d\r\n%s\r\n", iLicMsg, CBarcodeReader::GetErrorString(iLicMsg));
        return iLicMsg;
    }
}
// Decode barcodes happens here
//....
delete reader;
```

Code snippet in Java:

```
int iLicMsg = -1;
File file = new File("Put your file path here");
BarcodeReader reader = new BarcodeReader();
// Check if there is a license file on the local machine. If not, connect to Dynamsoft Hosted server to verify the license. Otherwise, use the license file.
if (!file.exists()){
    // Connect to Dynamsoft server to verify the license.
    iLicMsg = reader.initLicenseFromServer("", "licenseKey1;licenseKey2");// The first parameter is the string of the license server. Leaving it empty ("") means it will connect to Dynamsoft License Server for online verification.
    if(iLicMsg != 0)
    {
        System.out.println("License error Code:"+iLicMsg);
        return;
    }
    // If you wish to use SDK offline, store the license information as txt format or in other format
    String license = reader.outputLicenseToString();
    PrintWriter pw = new PrintWriter(file);
    pw.print(license);
    pw.close();
}
else{
    // Use the local license file and use Dynamsoft Barcode Reader SDK
    byte[] encoded = Files.readAllBytes(file.toPath());
    String license = new String(encoded, "utf-8");
    iLicMsg = reader.initLicenseFromLicenseContent("licenseKey1;licenseKey2",license);
    if(iLicMsg != 0)
    {
        System.out.println("License error Code:"+iLicMsg);
        return;
    }
}
```

Code snippet in Python:

```
def initLicenseFromServer(pLicenseServer,pLicenseKey):
    dbr.initLicenseFromServer(pLicenseServer,pLicenseKey)
def initLicenseFromLicenseContent(pLicenseKey,pLicenseContent):
    dbr.initLicenseFromLicenseContent(pLicenseKey,pLicenseContent)
def outputLicenseToString():
```

```
    content = dbr.outputLicenseToString()
    return content

#Check if there is a license file on the local machine. If not, connect to Dynamsoft Hosted server to verify the license. Otherw
ise, use the local license file.
if os.path.exists(license.txt):
   #Use the local license file to activate the SDK
   with open("license.txt","r") as f:
      pLicenseContent = f.read()
   initLicenseFromLicenseContent("licenseKey1",pLicenseContent)
else:
   initLicenseFromServer("","licenseKey1")
   #If you wish to use SDK offline, store the license information as .txt or other format
   content=outputLicenseToString()
   with open("license.txt","w") as f:
      f.write(content)
```

Note:

- The license verification process on the development machine can be a one-time process. Once it is registered, the registration file for this specific device can be returned and stored to the machine.
- If you need to increase the quota of your existing developer license key, please contact us.

Always connect to Dynamsoft server for license verification

If your development machine can access Internet all the time, you can use the `initLicenseFromServer()` method to register the development license. It will connect to Dynamsoft server for license verification each time you use the SDK.

Code snippet in C:

```
void* _br = NULL;
int iLicMsg = -1;
_br = DBR_CreateInstance();
// Connect to the Dynamsoft server to verify the license
iLicMsg = DBR_InitLicenseFromServer(_br, "", "licenseKey1;licenseKey2");
// If error occurs to the license
if (iLicMsg != DBR_OK) {
   printf("Failed to initialize the license: %d\r\n%s\r\n", iLicMsg, DBR_GetErrorString(iLicMsg));
   return iLicMsg;
}
// Barcode decoding happens here
//....
DBR_DestroyInstance(_br);
```

Code snippet in C++:

```
CBarcodeReader* reader = new CBarcodeReader();
int iLicMsg = -1;
// Connect to Dynamsoft server to verify the license.
iLicMsg = reader->InitLicenseFromServer("", "licenseKey1;licenseKey2");
// If error occurs to the license
if (iLicMsg != DBR_OK)
{
   printf("Failed to initialize the license: %d\r\n%s\r\n", iLicMsg, CBarcodeReader::GetErrorString(iLicMsg));
   return iLicMsg;
}
// Decode barcodes happens here
//....
delete reader;
```

Code snippet in Java:

```
int iLicMsg = -1;
BarcodeReader reader = new BarcodeReader();
// Connect to Dynamsoft server to verify the license.
iLicMsg = reader.initLicenseFromServer("", "licenseKey1;licenseKey2");
if(iLicMsg != 0)
{
   System.out.println("License error Code:"+iLicMsg);
   return;
```

```
}
```

Code snippet in Python:

```
def initLicenseFromServer(pLicenseServer,pLicenseKey):
    dbr.initLicenseFromServer(pLicenseServer,pLicenseKey)
#Check if there is a license file on the local machine. If not, connect to Dynamsoft Hosted server to verify the license. Otherwise,
 use the local license file.
if os.path.exists(license.txt):
   #Use the local license file to activate the SDK
   with open("license.txt","r") as f:
      pLicenseContent = f.read()
   initLicenseFromLicenseContent("licenseKey1",pLicenseContent)
else:
   initLicenseFromServer("","licenseKey1")
   #If you wish to use SDK offline, store the license information as .txt or other format
   content=outputLicenseToString()
   with open("license.txt","w") as f:
      f.write(content)
```

No Internet connection

If your machine is not allowed to connect to Internet, please contact us for other options.

Use a Runtime License

For Desktop Applications

The procedure for setting a desktop runtime license is the same as using a development license. You can use `initLicenseFromServer()` or `initLicenseFromLicenseContent()` to complete the license registration. Please refer to Use a Development License section for more information.

For Web Applications

The procedure for setting a server runtime license is the same as using a trial license. You can use `initLicense()` or `ProductKeys` to set the license. Please refer to Use a Trial License section for more information.

Distribution

Licensing

Once you finish the development and testing of your application with a Development License, you'll need a Runtime License to distribute your application. For more information on how to set a runtime license, please refer to Use a Runtime License.

Distributing

Distribute the appropriate assembly files with the application using Dynamsoft Barcode Reader SDK.

Programming language	Files for distribution or deployment
C/C++	*libDynamsoftBarcodeReader.so*, *libDynamLicenseClient.so*, *libDynamicPdf.so* (optional)
Java	*DynamsoftBarcodeReader_Linux.jar*

Note: If you want to read barcodes from PDF files through C/C++ API, please put *libDynamicPdf.so* in the same folder as the core assembly file (*libDynamsoftBarcodeReader.so*). Otherwise, this is not required.

Additional Resources

Demos and Samples

- Read barcodes from images online demo
- Code gallery
- Github repositories

API Reference

- Windows Edition
- Linux Edition
- JavaScript Edition
- iOS Edition
- Android Edition

Release Notes

Release Notes - Dynamsoft Barcode Reader SDK

Get Support

If you need any help, please feel free to contact us via email, telephone, live chat etc.

Contact Dynamsoft >