IRD PRINT API

Version 1.05

2/23/2017

COPYRIGHT

Revision History:

Revision	Author	Date	Notes
1.01	Toby Ball	09/01/16	Document created
1.02	Toby Ball	11/10/16	Added print to file for printers.
1.03	Toby Ball	1/13/2017	Fixed signature for 'Print' methods.
			Updated x9Device documentation for printing to file.
1.04	Toby Ball	2/22/2017	For x9IRD, x9CRD and x9RNL, • Added GenerateSemaphoreFiles • Added OutputToFileDestination • Moved MainTray, PrinterType, PrintDensity and SeparatorTray from device class. • Moved Offsets and Scaling from device class. Added x9String class for c++ SDK.
1.05	Toby Ball	2/23/2017	Fixed inconsistencies in example code.

TABLE OF CONTENTS

OVERVIEW

This document describes an API for printing US Image Replacement Documents (IRD), Canadian Image Replacement Documents (CRD), Return Notification Letters (RNL) from Image Cash Letter (ICL) files and custom formatted documents made up of images from files on the disk.

REQUIREMENTS

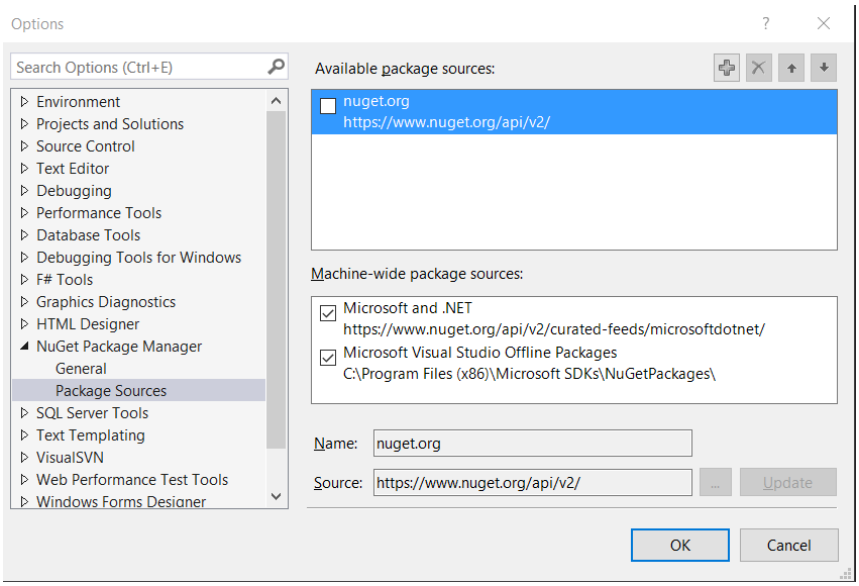
You will need Visual Studio 2010, 2012, 2013 or 2015 to build.

The libraries support both 32 and 64 bit platforms on Windows 7 and later.

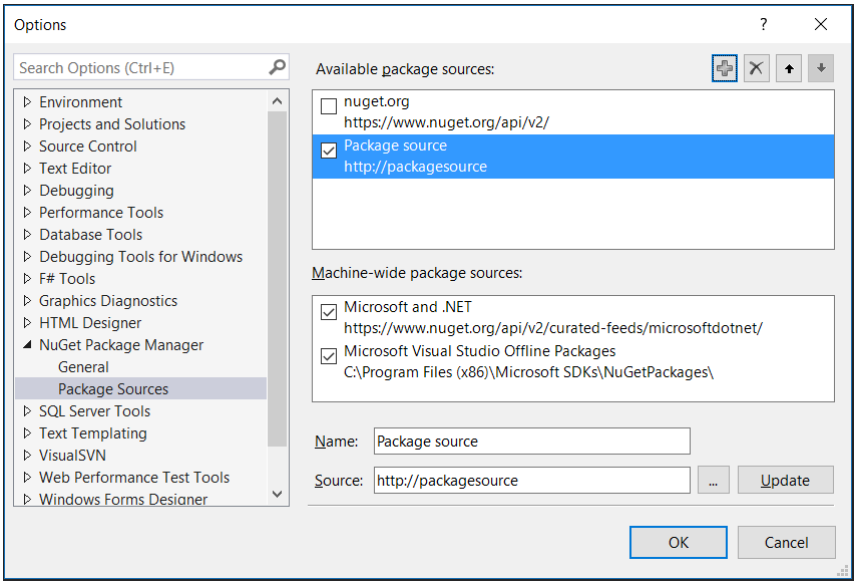
Note that native applications will need to be built against the statically linked runtime libraries. See this article, http://www.codeproject.com/Articles/85391/Microsoft-Visual-C-Static-and-Dynamic-Libraries, for instructions on how to do this.

SETUP

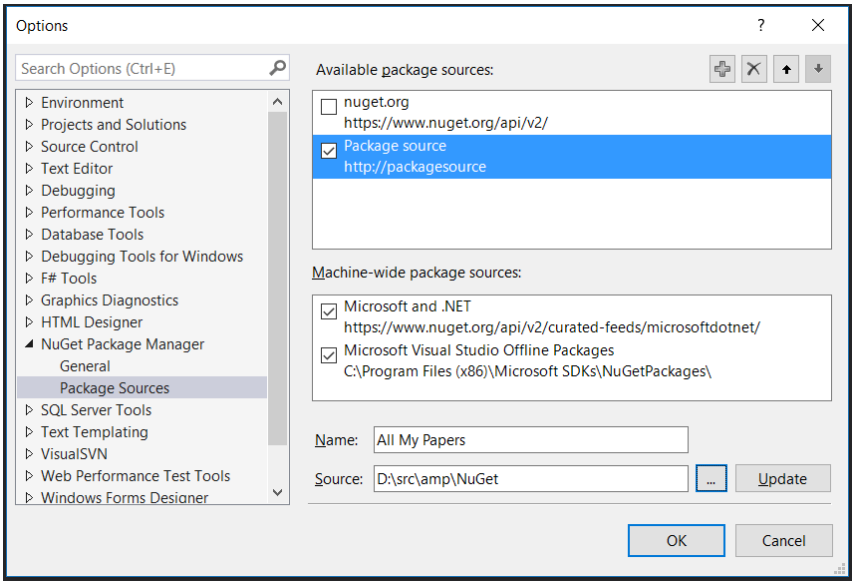
The libraries are provided as NuGet packages. There is one for the native libraries and one for the managed ones. To install the required package for your project, select Tools->Options.



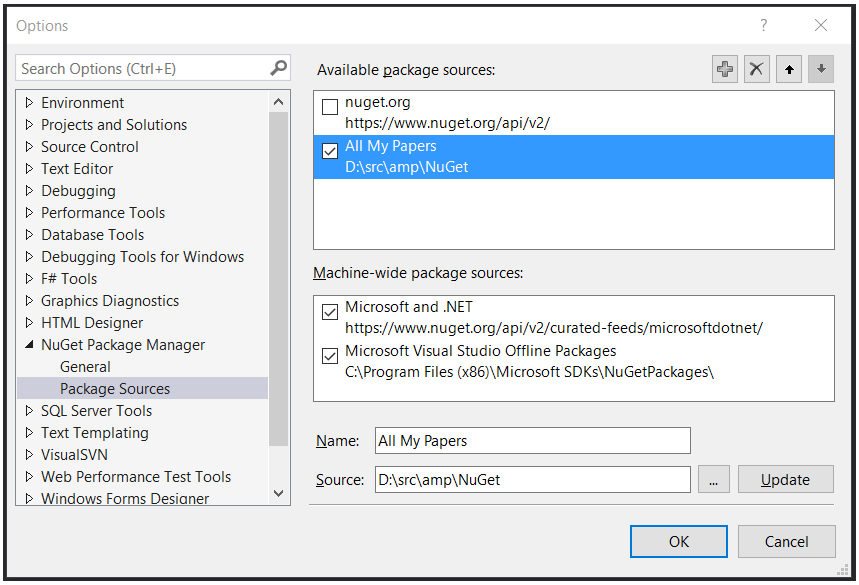
Click on the '+' icon in the upper right corner to add a new package source.



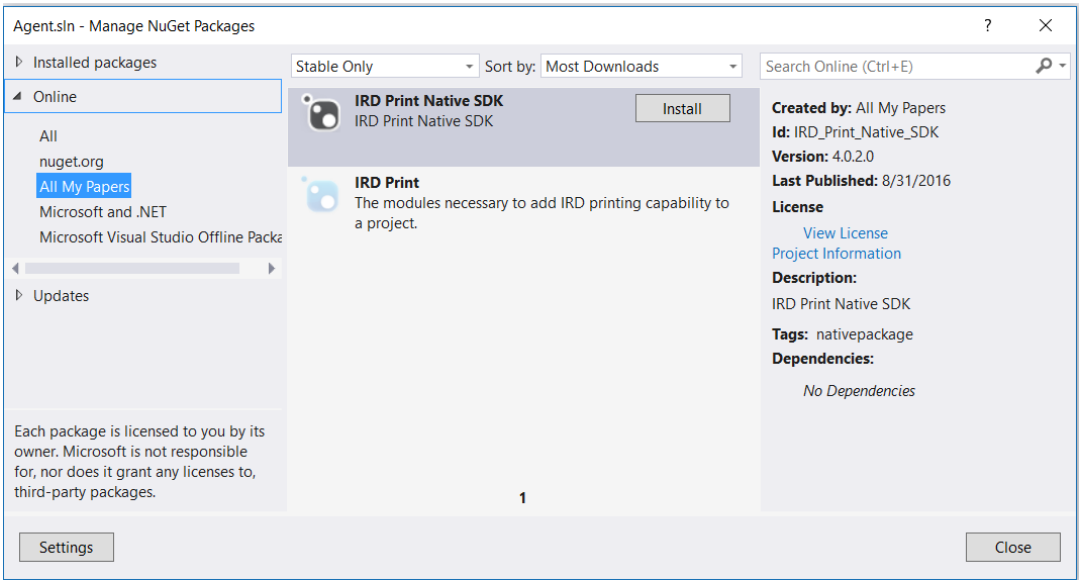
Now, name your source and add the path to the local directory where your NuGet packages are stored. In this case, I am using “All My Papers” as the name and “d:\src\amp\nuget” as the path.



Click the ‘update’ button.



Now click ‘OK’. If you now selection Tools->NuGet Package Manager->Manage NuGet Packages for Solution and select ‘All My Papers’ as the source, it should look something like this.



Select the appropriate package and click on the ‘Install’ button next to it. You will be presented with a dialog containing a list of the projects in the solution. Select the projects for which you want to install the package and then click ‘OK’. You are now all setup.

REDISTRIBUTABLES

Native mode applications are entirely self-contained and require no support modules. For managed applications, AllMyPapers.IRDPrint.dll is required to be in the same directory as the application.

API

IRD Print API

Notes:

- In C++, ‘string’ values are x9String values which may be interpreted as either MBCS or UNICODE.

DATETIME

This class is provided for C++ consumers as a substitute for the C# .NET System.DateTime structure in both API and functionality. Please reference the documentation at the following link https://msdn.microsoft.com/en-us/library/system.datetime(v=vs.110).aspx.

C++ Example:

```
#include "irdprint.h"

using namespace System;

void
Example (
)
{
    x9IRD ird;

    ird.CreationDate = DateTime::Now ();
    printf ("%s\n", ird.CreationDate.ToString ());
}
```

PATH

This class is provided for C++ consumers as a substitute for the C# .NET System.IO.Path structure in both API and functionality. Please reference the documentation at the following link https://msdn.microsoft.com/en-us/library/system.io.path(v=vs.110).aspx.

C++ Example:

```
#include "irdprint.h"

using namespace System::IO;

void
Example (
    __in x9String destinationPath
)
{
    auto outputDevices = x9Device::Installed;

    for (auto p = outputDevices.begin () ; p != outputDevices.end () ; p++)
    {
        auto& outputDevice = (*p);

        if (outputDevice->IsFile)
        {
            auto file = (x9File*)outputDevice.get ();

            break;
        }
    }
}
```

POINTF

This class is provided for C++ consumers as a substitute for the C# .NET System.Drawing.PointF structure in both API and functionality. Please reference the documentation at the following link https://msdn.microsoft.com/en-us/library/system.drawing.pointf(v=vs.110).aspx.

C++ Example:

```
#include "irdprint.h"

using namespace System;

void
Example (
)
{
    x9IRD ird;

    ird.BackSideOffsets = PointF (0, 0);
    ird.BackSideScaling = PointF (1, 1);
    ird.FrontSideOffsets = PointF (0, 0);
    ird.FrontSideScaling = PointF (1, 1);
}
```

RANGE

Represents a range of numbers.

Namespace: System

Constructors

	Name	Description
	Range (Int32, Int32)	Initializes a new instance of the Range class with the specified values.
	Range ()	Initializes a new instance of the Range class.

Properties

	Name	Description
	Begin	Gets a value indicating the start of the Range.
	End	Gets a value indicating the end of the Range.
	IsEmpty	Gets a value indicating whether this Range is empty.
	Length	Initializes a new instance of the Range class.

C++ Example:

```
#include "irdprint.h"

using namespace System;

void
Example (
)
```

```
{
    Range r (0, 1);

    printf ("Begin: %d\n", r.Begin);
    printf ("End: %d\n", r.End);
    printf ("IsEmpty: %s\n", r.IsEmpty ? "true" : "false");
    printf ("Length: %d\n", r.Length);
}
```

C# Example:

```
using System;

public
void
Example (
)
{
    var r = new Range (0, 1);

    Console.WriteLine ("Begin: {0}", r.Begin);
    Console.WriteLine ("End: {0}", r.End);
    Console.WriteLine ("IsEmpty: {0}", r.IsEmpty);
    Console.WriteLine ("Length: {0}", r.Length);
}
```

RANGE (INT32, INT32)

Initializes a new instance of the Range class with the specified values.

C++

```
Range::Range (
    __in int beg,
    __in int end
)
```

C#

```
public
Range (
    int beg,
    int end
)
```

Parameters

Name	Description
beg	The beginning of the range.
end	The end of the range.

RANGE ()

Initializes a new instance of the Range class with the default values.

C++

```
Range::Range (
)
```

C#

```
public
Range (
)
```

BEGIN

Gets or sets the beginning of the range.

C++

```
__declspec (property (get=get_Begin, put=set_Begin)) int Begin;

int
get_Begin (
)
    const;

void
set_Begin (
    __in int value
);
```

C#

```
public int Begin { get; set; }
```

Property Value

Type: integer
Default Value: 0

END

Gets or sets the end of the range.

C++

```
__declspec (property (get=get_End, put=set_End)) int End;

int
get_End (
)
    const;

void
set_End (
    __in int value
);
```

C#

```
public int End { get; set; }
```

Property Value

Type: integer

Default Value: 2^{32} - 1

ISEMPTY

Gets a value indicating whether this Range is empty.

C++

```
__declspec (property (get=get_IsEmpty)) bool IsEmpty;

bool
get_IsEmpty (
)
    const;
```

C#

```
public bool IsEmpty { get; }
```

Property Value

Type: bool
Value: **true** if Length <= 0; otherwise, **false**

LENGTH

Gets the length of the range defined by this Range object.

C++

```
__declspec (property (get=get_Length)) int Length;

int
get_Length (
)
    const;
```

C#

```
public int Length { get { return (End - Begin) + 1; } }
```

Property Value

Type: integer
Value: End - Begin

RECTANGLEF

This class is provided for C++ consumers as a substitute for the C# .NET System.Drawing.RectangleF structure in both API and functionality. Please reference the documentation at the following link https://msdn.microsoft.com/en-us/library/system.drawing.rectanglef(v=vs.110).aspx.

C++ Example:

```
#include "irdprint.h"
```

```
using namespace System::Drawing;
using namespace AllMyPapers::IRDPrint;

void
Example (
    __in DEVICE_PTR outputDevice,
    __in FORM_PTR form,
    __in std::wstring imageFile,
)
{

    x9DOC doc;

    doc.StartDocument (L"Test", form, jobID);
    doc.StartPage ();

    doc.SetBounds (2.1f, 0.1995f, 5.75f, 2.75f);

    doc.Image (imageFile.c_str (), 1, RectangleF (0, 0, 0, 0),
        ampSCALE::NS_BEST_FIT, PointF (1, 1), ampPOSITION::Center,
        ampROTATION::Angle0, 200, false, false);

    doc.EndPage ();
    doc.EndDocument ();
}
```

X9BASE

This is the base class for all document classes.

Namespace: AllMyPapers::IRDPrint

Properties

	Name	Description
	BackSideOffsets	Offsets to correct the positioning of the back side, in inches.
	BackSideScaling	Scale factors to correct the sizing of the back side.
	FrontSideOffsets	Offsets to correct the positioning of the front side, in inches.
	FrontSideScaling	Scale factors to correct the sizing of the front side.
	MainTray	An x9Tray object that describes the input tray for the job.
	OutputToFileDestination	The fully qualified path of the file when outputting to file.
	PrintDensity	The density of the toner (from 0 – 5).
	PrinterType	The type of printer (used for tracking internal fonts).
	SeparatorTray	An x9Tray object that describes the input tray for separator pages.

BACKSIDEOFFSETS

Get or sets a PointF object with X and Y values which are used to offset the back side of the page to compensate for any misplacement due to printer alignment.

C++

```
__declspec (property (get=get_BackSideOffsets, put=set_BackSideOffsets)) System::Drawing::PointF
BackSideOffsets;
```

```
System::Drawing::PointF
get_BackSideOffsets (
)
    const;

void
set_BackSideOffsets (
    __in System::Drawing::PointF& value
);
```

C#

```
public PointF BackSideOffsets { get; set; }
```

Property Value

```
Type: PointF
Value: page offsets, in inches
Default Value: PointF (0, 0)
```

BACKSIDESCALING

Get or sets a PointF object with X and Y values which are used to scale the back side of the page to compensate for any misplacement due to printer alignment. A scale factor of 1 does not affect the page, while 0.5 reduces the size by half and 2 doubles the size.

C++

```
__declspec (property (get=get_BackSideScaling, put=set_BackSideScaling)) System::Drawing::PointF
BackSideScaling;

System::Drawing::PointF
get_BackSideScaling (
)
    const;

void
set_BackSideScaling (
    __in System::Drawing::PointF& value
);
```

C#

```
public PointF BackSideScaling { get; set; }
```

Property Value

```
Type: PointF
Value: scale factors
Default Value: PointF (1, 1)
```

FRONTSIDEOFFSETS

Get or sets a PointF object with X and Y values which are used to offset the front side of the page to compensate for any misplacement due to printer alignment.

C++

```
__declspec (property (get=get_FrontSideOffsets, put=set_FrontSideOffsets)) System::Drawing::PointF
FrontSideOffsets;

System::Drawing::PointF
get_FrontSideOffsets (
)
    const;

void
set_FrontSideOffsets (
    __in System::Drawing::PointF& value
);
```

C#

```
public PointF FrontSideOffsets { get; set; }
```

Property Value

```
Type: PointF
Value: page offsets, in inches
Default Value: PointF (0, 0)
```

FRONTSIDESCALING

Get or sets a PointF object with X and Y values which are used to scale the front side of the page to compensate for any misplacement due to printer alignment. A scale factor of 1 does not affect the page, while 0.5 reduces the size by half and 2 doubles the size.

C++

```
__declspec (property (get=get_FrontSideScaling, put=set_FrontSideScaling)) System::Drawing::PointF
FrontSideScaling;

System::Drawing::PointF
get_FrontSideScaling (
)
    const;

void
set_FrontSideScaling (
    __in System::Drawing::PointF& value
);
```

C#

```
public PointF FrontSideScaling { get; set; }
```

Property Value

```
Type: PointF
Value: scale factors
Default Value: PointF (1, 1)
```

MAINTRAY

Gets or sets the x9Tray object that describes the input tray on the device from which paper for the document will be sourced.

C++

```
__declspec (property (get=get_MainTray, put=set_MainTray)) TRAY_PTR MainTray;

TRAY_PTR
get_MainTray (
)
    const;

void
set_MainTray (
    __in TRAY_PTR value
);
```

C#

```
public x9Tray MainTray { get; set; }
```

Property Value

```
Type: x9Tray
Default Value: x9Tray::Default
```

OUTPUTTOFILEDESTINATION

Gets or sets a fully qualified path that describes the output file when printing to file. If this value is set, the print operation will send data to a file instead of a device. In the case of x9File devices, this property is mandatory.

C++

```
__declspec (property (get=get_OutputToFileDestination, put=set_OutputToFileDestination)) x9String
OutputToFileDestination;

x9String
get_OutputToFileDestination (
)
    const;

void
set_OutputToFileDestination (
    __in x9String value
);
```

C#

```
public string OutputToFileDestination { get; set; }
```

Property Value

```
Type: string
Default Value: The fully qualified path of the output file.
```

PRINTDENSITY

Gets or sets a value dictating the relative density of printer toner that should be applied when printing.

C++

```
__declspec (property (get=get_PrintDensity, put=set_PrintDensity)) int PrintDensity;

int
get_PrintDensity (
)
    const;

void
set_PrintDensity (
    __in int value
);
```

C#

```
public int PrintDensity { get; set; }
```

Property Value

```
Type: int
Value: 0 to 5 where higher numbers are darker
Default Value: 0
```

PRINTERTYPE

Gets or sets a value indicating the type of printer. Certain printers have internal MICR fonts that are leveraged by the SDK. Set this value to ensure the best results.

C++

```
__declspec (property (get=get_PrinterType, put=set_PrinterType)) ePrinterType PrinterType;

ePrinterType
get_PrinterType (
)
    const;

void
set_PrinterType (
    __in ePrinterType value
);
```

C#

```
public PrinterType PrinterType { get; set; }
```

Property Value

```
Type: PrinterType (ePrinterType in C++)
Value: { GENERIC, OCE, Troy, Xerox }
Default Value: GENERIC
```

SEPARATORTRAY

Gets or sets the x9Tray object that describes the input tray on the device from which paper for the separators will be sourced.

```
__declspec (property (get=get_SeparatorTray, put=set_SeparatorTray)) TRAY_PTR SeparatorTray;
```

```
TRAY_PTR
get_SeparatorTray (
)
    const;

void
set_SeparatorTray (
    __in TRAY_PTR value
);
```

C#

```
public x9Tray SeparatorTray { get; set; }
```

Property Value

```
Type: x9Tray
Default Value: x9Tray::Default
```

X9CRD

The object used to output a Canadian Image Replacement Document (CRD).

Namespace: AllMyPapers::IRDPrint

Inheritance Heirarchy

AllMyPapers::IRDPrint::x9BASE
  AllMyPapers::IRDPrint::x9IRD
    AllMyPapers::IRDPrint::x9CRD

Constructors

	Name	Description
	x9CRD (string)	Initializes a new instance of the x9CRD class.

Properties

	Name	Description
	BundleRange	The range of bundles to print (See x9IRD).
	BackSideOffsets	Offsets to correct the positioning of the back side, in inches (See x9BASE).
	BackSideScaling	Scale factors to correct the sizing of the back side (See x9BASE).
	CashLetterRange	The range of cash letters to print (See x9IRD).
	CreationDate	The creation date of the document (See x9IRD).
	CreatorRoutingNumber	The routing number for the creating institution (See x9IRD).
	DPI	The image resolution, in dots per inch (See x9IRD).
	File	The name of the ICL file associated with the object (See x9IRD).
	FrontSideOffsets	Offsets to correct the positioning of the front side, in inches (See x9BASE).
	FrontSideScaling	Scale factors to correct the sizing of the front side (See x9BASE).
	GenerateSemaphoreFiles	If true, semaphore files are generated on output to file (See x9IRD).
	GenerateSeparatorPages	If true, separator pages are inserted between bundles (See x9IRD).

	IsQualifiedReturn	If true, returns are rendered as qualified returns (See x9IRD).
	ItemRange	The range of items to print (See x9IRD).
	MainTray	An x9Tray object that describes the input tray for the job (See x9BASE).
	nUp	The number of items per page (See x9IRD).
	OutputToFileDestination	The fully qualified path of the file when outputting to file (See x9BASE).
	PrintDensity	The density of the toner (from 0 – 5) (See x9BASE).
	PrinterType	The type of printer (used for tracking internal fonts) (See x9BASE).
	ReverseFrontAndBack	If true, the front and back sides of each page are switched (See x9IRD).
	SeparatorTray	An x9Tray object that describes the input tray for separator pages (See x9BASE).
	VOID	If true, a watermark is rendered over the front side images (See x9IRD).
	WorkDirectory	The directory for storing temp files during printing (See x9IRD).

C++ Example:

```
#include "irdprint.h"

using namespace System;
using namespace System::Drawing;
using namespace AllMyPapers::IRDPrint;

void
Example (
    __in DEVICE_PTR outputDevice,
    __in std::wstring iclFile
)
{
    x9CRD crd (L"test.937");

    crd.BundleRange = Range (0, MAXINT32);
    crd.CashLetterRange = Range (0, MAXINT32);
    crd.CreationDate = DateTime::Now ();
    crd.CreatorRoutingNumber = L"123456789";
    crd.DPI = 200;
    crd.GenerateSeparatorPages = false;
    crd.IsQualifiedReturn = false;
    crd.ItemRange = Range (0, MAXINT32);
    crd.nUp = 3;
    crd.ReverseFrontAndBack = false;
    crd.Void = false;

    if (outputDevice->IsPrinter)
    {
        auto printer = (x9Printer*)outputDevice.get ();

        crd.BackSideOffsets = PointF (0, 0);
        crd.BackSideScaling = PointF (1, 1);
        crd.FrontSideOffsets = PointF (0, 0);
        crd.FrontSideScaling = PointF (1, 1);
        crd.MainTray = x9Tray::get_Default ();
        crd.PrintDensity = 0;
        crd.PrinterType = GENERIC;
        crd.SeparatorTray = x9Tray::get_Default ();
    }
    else
    {
        auto file = (x9File*)outputDevice.get ();
```

```
        crd.OutputToFileDestination = std::wstring (L"test.") + file->Extension;
        crd.GenerateSemaphoreFiles = false;
    }

    try
    {
        outputDevice.Print (crd);
        printf ("success\n");
    }
    catch (x9Exception& e)
    {
        printf ("%ws\n", e.Message.c_str ());
    }
}
```

C# Example:

```
using System;
using System.Drawing;
using AllMyPapers.IRDPrint;

public
void
Example (
    x9Device outputDevice,
    string iclFile
)
{
    var crd = new x9CRD ("test.937");

    crd.BundleRange = new Point (0, int.MaxValue);
    crd.CashLetterRange = new Point (0, int.MaxValue);
    crd.CreationDate = DateTime.Now;
    crd.CreatorRoutingNumber = "123456789";
    crd.DPI = 200;
    crd.GenerateSeparatorPages = false;
    crd.IsQualifiedReturn = false;
    crd.ItemRange = new Point (0, int.MaxValue);
    crd.nUp = 3;
    crd.ReverseFrontAndBack = false;
    crd.Void = false;

    if (outputDevice is x9Printer)
    {
        var printer = outputDevice as x9Printer;

        crd.BackSideOffsets = new PointF (0, 0);
        crd.BackSideScaling = new PointF (1, 1);
        crd.FrontSideOffsets = new PointF (0, 0);
        crd.FrontSideScaling = new PointF (1, 1);
        crd.MainTray = x9Tray.Default;
        crd.PrintDensity = 0;
        crd.PrinterType = PrinterType.Generic;
        crd.SeparatorTray = x9Tray.Default;
    }
    else
    {
        var file = outputDevice as x9File;

        crd.OutputToFileDestination = "test." + file.Extension;
        crd.GenerateSemaphoreFiles = false;
    }

    try
    {
```

```
        outputDevice.Print (crd);
        Console.WriteLine ("success");
    }
    catch (x9Exception e)
    {
        Console.WriteLine (e.Message);
    }
}
```

X9CRD (STRING)

Initializes a new instance of the x9CRD class.

C++

```
x9CRD:: x9CRD (
    __in x9String file,
)
```

C#

```
public
x9CRD (
    string file
)
```

Parameters

Name	Description
file	The source ICL file.

X9DEVICE

An abstract class that defines an output device, be it a printer or a file, on or to which the data will be rendered.

Namespace: AllMyPapers::IRDPrint

Properties

	Name	Description
	CanWatermark	If true, the device is able to render watermarks.
	Description	Gets a textual description of the device (ie. “PDF File”).
	Extension	Gets the default extension used when printing to file.
	Forms	Gets a list of forms that are supported.
S	Installed	Gets a list of installed devices.
	IsFile	Specifies whether device is a file **(C++ only)**.
	IsPrinter	Specified whether device is a printer **(C++ only)**.

Methods

	Name	Description

	Print (x9IRD)	Renders the document on the device.

Notes

For C++, there are two definitions that are designed to simplify the use of these objects. The first provides an automatic pointer that will ensure that no memory gets leaked for x9Device objects and the second, a list object to contain multiple x9Device objects.

```
typedef std::shared_ptr<x9Device> DEVICE_PTR;

typedef std::list<DEVICE_PTR> DEVICE_LIST;
```

C++ Example:

```
#include "irdprint.h"

using namespace System;
using namespace System::Drawing;
using namespace AllMyPapers::IRDPrint;

void
Example (
    __in DEVICE_PTR outputDevice,
    __in std::wstring iclFile
)
{
    auto outputDevices = x9Device::Installed;

    for (auto p = outputDevices.begin () ; p != outputDevices.end () ; p++)
    {
        auto& outputDevice = (*p);

        printf ("%ws\n", (LPCWSTR)outputDevice->Description.c_strW ());
        printf ("  CanWatermark: %ws\n", outputDevice->CanWatermark ? L"true" : L"false");
        printf ("  Extension: %ws\n", (LPCWSTR)outputDevice->Extension.c_strW ());
        printf ("  IsPrinter: %ws\n", outputDevice->IsPrinter ? L"true" : L"false");
        printf ("  IsFile: %ws\n", outputDevice->IsFile ? L"true" : L"false");
        printf ("  Forms:\n");

        auto& forms = outputDevice->Forms;

        for (auto p = forms.begin (), p != forms.end () ; ++p)
        {
            auto& f = (*p);

            printf ("    %ws (%f x %f)\n", (LPCWSTR)p->Name.c_strW (), p->Width, p->Height);
        }

        if (outputDevice->IsPrinter)
        {
            auto printer = (x9Printer*)outputDevice.get ();
        }
        else
        {
            auto file = (x9File*)outputDevice.get ();
        }
    }
}
```

C# Example:

```
using System;
```

```
using System.Drawing;
using AllMyPapers.IRDPrint;

public
void
Example (
)
{
    var outputDevices = x9IRD.Installed;

    foreach (var outputDevice in outputDevices)
    {
        Console.WriteLine (outputDevice.Description);
        Console.WriteLine ("  CanWatermark: " + outputDevice.CanWatermark);
        Console.WriteLine ("  Extension: " + outputDevice.Extension);
        Console.WriteLine ("  Forms:");

        var forms = outputDevice.Forms;

        foreach (var f in forms)
        {
            Console.WriteLine ("    " + f.Name + " (" + f.Width + " x " + f.Height + ")");
        }

        if (outputDevice is x9Printer)
        {
            var printer = outputDevice as x9Printer;
        }
        else
        {
            var file = outputDevice as x9File;
        }
    }
}
```

CANWATERMARK

Gets a value indicating whether this Device supports watermarking. If true, setting the VOID property of an x9IRD derived object to true will result in a “VOID” watermark being rendered on top of the front image.

C++

```
__declspec (property (get=get_CanWatermark)) bool CanWatermark;

virtual
bool
get_CanWatermark (
)
    const = 0;
```

C#

```
abstract bool CanWatermark { get; }
```

Property Value

Type: bool
Value: **true** if the device supports rendering of watermarks on top of an image; otherwise **false**

DESCRIPTION

Gets a description of the Device (ie. “HP LaserJet 4050 Series PCL 5”).

C++

```
__declspec (property (get=get_Description)) x9String Description;

virtual
x9String
get_Description (
)
    const = 0;
```

C#

```
abstract string Description { get; }
```

Property Value

Type: string

EXTENSION

Gets a the default file extension used when printing to file.

C++

```
__declspec (property (get=get_Extension)) x9String Extension;

virtual
x9String
get_Extension (
)
    const = 0;
```

C#

```
abstract string Extension { get; }
```

Property Value

Type: string

FORMS

Get a list of x9Form objects describing the forms supported on the Device.

C++

```
__declspec (property (get=get_Forms)) FORM_LIST Forms;

virtual
FORM_LIST
get_Forms (
)
    const = 0;
```

C#

```
abstract List<x9Form> Forms { get; }
```

Property Value

Type: list of x9Form object (in C++, FORM_LIST; in C#, List<x9Form>)

INSTALLED

A static property that returns the list of recognized output devices.

C++

```
static
std::list<DEVICE_PTR>
get_Installed (
);
```

C#

```
public static List<x9Device> Installed { get; }
```

Property Value

Type: std::list<DEVICE_PTR> or List<x9Device>
Value: a list of recognized output devices

ISFILE

Gets a value that specifies whether this Device renders to a file and implements the x9file interface.

C++

```
__declspec (property (get=get_IsFile)) bool IsFile;

virtual
bool
get_IsFile (
)
    const = 0;
```

C#

```
abstract bool IsFile { get; }
```

Property Value

Type: bool
Value: **true** if the device is a file; otherwise **false**

ISPRINTER

Gets a value that specifies whether this Device renders to a file and implements the x9Printer interface.

C++

```
__declspec (property (get=get_IsPrinter)) bool IsPrinter;
```

```
virtual
bool
get_IsPrinter (
)
    const = 0;
```

C#

```
abstract bool IsPrinter { get; }
```

Property Value

```
Type: bool
Value: true if the device is a printer; otherwise false
```

PRINT (X9IRD)

Renders the Image Replacement Document (IRD) on the output device.

C++

```
void
Print (
    __in x9IRD outputDevice
);
```

C#

```
public
void
Print (
    x9IRD outputDevice
)
```

Parameters

Name	Description
ird	The Image Replacement Document (IRD) to render.

Note that there is no return value from this method. If the print operation fails, an exception will be thrown with an x9Exception that will contain information about the error.

X9EXCEPTION

An exception object that captures data related to an error condition within the library.

Namespace: AllMyPapers::IRDPrint

Constructors

	Name	Description
	x9Exception (string)	Initializes a new instance of the x9Exception class with a message.

Properties

	Name	Description
	Message	The message describing the error.

C++ Example:

```
#include "irdprint.h"

using namespace System;
using namespace System::Drawing;
using namespace AllMyPapers::IRDPrint;

void
Example (
    __in DEVICE_PTR outputDevice,
    __in std::wstring iclFile
)
{
    x9CRD crd (L"test.937");

    crd.BundleRange = Range (0, MAXINT32);
    crd.CashLetterRange = Range (0, MAXINT32);
    crd.CreationDate = DateTime::Now ();
    crd.CreatorRoutingNumber = L"123456789";
    crd.DPI = 200;
    crd.GenerateSeparatorPages = false;
    crd.IsQualifiedReturn = false;
    crd.ItemRange = Range (0, MAXINT32);
    crd.nUp = 3;
    crd.ReverseFrontAndBack = false;
    crd.Void = false;

    if (outputDevice->IsPrinter)
    {
        auto printer = (x9Printer*)outputDevice.get ();

        printer->BackSideOffsets = PointF (0, 0);
        printer->BackSideScaling = PointF (1, 1);
        printer->FrontSideOffsets = PointF (0, 0);
        printer->FrontSideScaling = PointF (1, 1);
        printer->MainTray = x9Tray::get_Default ();
        printer->PrintDensity = 0;
        printer->PrinterType = GENERIC;
        printer->SeparatorTray = x9Tray::get_Default ();
    }
    else
    {
        auto file = (x9File*)outputDevice.get ();

        crd.OutputToFileDestination = std::wstring (L"test.") + file->Extension;
        crd.GenerateSemaphoreFiles = false;
    }

    try
    {
        outputDevice.Print (crd);
        printf ("success\n");
    }
    catch (x9Exception& e)
    {
        printf ("%ws\n", e.Message.c_str ());
    }
}
```

C# Example:

```
using System;
using System.Drawing;
using AllMyPapers.IRDPrint;

public
void
Example (
    x9Device outputDevice,
    string iclFile
)
{
    var crd = new x9CRD ("test.937");

    crd.BundleRange = new Point (0, int.MaxValue);
    crd.CashLetterRange = new Point (0, int.MaxValue);
    crd.CreationDate = DateTime.Now;
    crd.CreatorRoutingNumber = "123456789";
    crd.DPI = 200;
    crd.GenerateSeparatorPages = false;
    crd.IsQualifiedReturn = false;
    crd.ItemRange = new Point (0, int.MaxValue);
    crd.nUp = 3;
    crd.ReverseFrontAndBack = false;
    crd.Void = false;

    if (outputDevice is x9Printer)
    {
        var printer = outputDevice as x9Printer;

        crd.BackSideOffsets = new PointF (0, 0);
        crd.BackSideScaling = new PointF (1, 1);
        crd.FrontSideOffsets = new PointF (0, 0);
        crd.FrontSideScaling = new PointF (1, 1);
        crd.MainTray = x9Tray.Default;
        crd.PrintDensity = 0;
        crd.PrinterType = PrinterType.Generic;
        crd.SeparatorTray = x9Tray.Default;
    }
    else
    {
        var file = outputDevice as x9File;

        crd.OutputToFileDestination = "test." + file.Extension;
        crd.GenerateSemaphoreFiles = false;
    }

    try
    {
        outputDevice.Print (crd);
        Console.WriteLine ("success");
    }
    catch (x9Exception e)
    {
        Console.WriteLine (e.Message);
    }
}
```

X9EXCEPTION (STRING)

Initializes an instance of the x9Exception class with the specified message.

C++

```
x9Exception (
    __in_z std::wstring message
);
```

C#

```
public
x9Exception (
    string message
)
```

Parameters

Name	Description
message	The error message.

MESSAGE

Returns the error message from the exception.

C++

```
__declspec (property (get=get_Message)) x9String Message;

x9String
get_Message (
)
    const;
```

C#

```
public string Message { get; }
```

Property Value

```
Type: string
Value: the error message
```

X9FILE

An interface to an output device that renders to a file.

Namespace: AllMyPapers::IRDPrint

Inheritance Heirarchy

AllMyPapers::IRDPrint::x9Device
  AllMyPapers::IRDPrint::x9File

Properties

	Name	Description
	CanWatermark	If true, the device is able to render watermarks (See x9Device).

	Description	Gets a textual description of the device (See x9Device).
	Extension	Gets the default extension used when printing to file(See x9Device).
	Forms	Gets a list of forms that are supported(See x9Device).
S	Installed	Gets a list of installed devices(See x9Device).
	IsFile	Returns true for x9File objects **(C++ only)** (See x9Device).
	IsPrinter	Returns false for x9File objects **(C++ only)** (See x9Device).

C++ Example:

```
#include "irdprint.h"

using namespace System;
using namespace System::Drawing;
using namespace AllMyPapers::IRDPrint;

public
void
Example (
    __in DEVICE_PTR outputDevice
)
{
    x9IRD ird;

    if (outputDevice->IsFile)
    {
        auto file = (x9File*)outputDevice.get ();

        printf ("%ws\n", (LPCWSTR)file->Description.c_strW ());
        printf ("  CanWatermark: %ws\n", file->CanWatermark ? L"true" : L"false");
        printf ("  Extension: %ws\n", (LPCWSTR)file->Extension.c_strW ());
        printf ("  IsPrinter: %ws\n", file->IsPrinter ? L"true" : L"false");
        printf ("  IsFile: %ws\n", file->IsFile ? L"true" : L"false");
    }

    outputDevice.Print (ird);
}
```

C# Example:

```
using System;
using System.Drawing;
using AllMyPapers.IRDPrint;

public
void
Example (
    x9Device outputDevice
)
{
    x9IRD ird = new x9IRD ();

    if (outputDevice is x9File)
    {
        var file = outputDevice as x9File;

        Console.WriteLine (file.Description);
        Console.WriteLine ("  CanWatermark: " + file.CanWatermark);
        Console.WriteLine ("  Extension: " + file.Extension);
    }
```

```
    outputDevice.Print (ird);
}
```

X9FORM

A read-only interface to a form object that describes a form that is available for rendering on an output device.

Namespace: AllMyPapers::IRDPrint

Properties

	Name	Description
	Height	The height, in inches, of the form.
	Name	A description of the form.
	Width	The width, in inches, of the form.

C++ Example:

```
#include "irdprint.h"

using namespace AllMyPapers::IRDPrint;

public
void
Example (
    __in DEVICE_PTR outputDevice,
)
{
    auto forms = outputDevice.Forms;

    for (auto form = forms.begin () ; form != forms.end () ; ++form)
    {
        printf ("%ws (%f x %f)\n", (LPCWSTR)form->Name.c_strW (), form->Width, form->Height);
    }
}
```

C# Example:

```
using AllMyPapers.IRDPrint;

public
void
Example (
    x9Device outputDevice
)
{
    var forms = outputDevice.Forms;

    foreach (var form in forms)
    {
        Console.WriteLine (form.Name + " (" + form.Width + " x " + form.Height + ")";
    }
}
```

HEIGHT

Gets the height of the form, in inches.

C++

```
__declspec (property (get=get_Height)) float Height;

virtual
float
get_Height (
)
    const = 0;
```

C#

```
abstract float Height { get; }
```

Property Value

```
Type: float
Value: the height, in inches
```

NAME

Gets the name of the form.

C++

```
__declspec (property (get=get_Name)) x9String Name;

virtual
x9String
get_Name (
)
    const = 0;
```

C#

```
abstract float Height { get; }
```

Property Value

```
Type: string
Value: the name of the form
```

WIDTH

Gets the width of the form, in inches.

C++

```
__declspec (property (get=get_Width)) float Height;

virtual
float
get_Width (
)
    const = 0;
```

C#

```
abstract float Width { get; }
```

Property Value

```
Type: float
Value: the width, in inches
```

X9IRD

The object used to output an Image Replacement Document (IRD).

Namespace: AllMyPapers::IRDPrint

Inheritance Heirarchy

AllMyPapers::IRDPrint::x9BASE
  AllMyPapers::IRDPrint::x9IRD

Constructors

	Name	Description
	x9IRD (string)	Initializes a new instance of the x9IRD class.

Properties

	Name	Description
	BundleRange	The range of bundles to print.
	BackSideOffsets	Offsets to correct the positioning of the back side, in inches (See x9BASE).
	BackSideScaling	Scale factors to correct the sizing of the back side (See x9BASE).
	CashLetterRange	The range of cash letters to print.
	CreationDate	The creation date of the document.
	CreatorRoutingNumber	The routing number for the creating institution.
	DPI	The image resolution, in dots per inch.
	File	The name of the ICL file associated with the object.
	FrontSideOffsets	Offsets to correct the positioning of the front side, in inches (See x9BASE).
	FrontSideScaling	Scale factors to correct the sizing of the front side (See x9BASE).
	GenerateSemaphoreFiles	If true, semaphore files are generated on output to file.
	GenerateSeparatorPages	If true, separator pages are inserted between bundles.
	IsQualifiedReturn	If true, returns are rendered as qualified returns.
	ItemRange	The range of items to print.
	MainTray	An x9Tray object that describes the input tray for the job (See x9BASE).
	nUp	The number of items per page.
	OutputToFileDestination	The fully qualified path of the file when outputting to file (See x9BASE).
	PrintDensity	The density of the toner (from 0 – 5) (See x9BASE).
	PrinterType	The type of printer (used for tracking internal fonts) (See x9BASE).
	ReverseFrontAndBack	If true, the front and back sides of each page are switched.

	SeparatorTray	An x9Tray object that describes the input tray for separator pages (See x9BASE).
	VOID	If true, a watermark is rendered over the front side images.
	WorkDirectory	The directory for storing temp files during printing.

C++ Example:

```
#include "irdprint.h"

using namespace System;
using namespace System::Drawing;
using namespace AllMyPapers::IRDPrint;

void
Example (
    __in DEVICE_PTR outputDevice,
    __in std::wstring iclFile
)
{
    x9IRD ird (L"test.937");

    ird.BundleRange = Range (0, MAXINT32);
    ird.CashLetterRange = Range (0, MAXINT32);
    ird.CreationDate = DateTime::Now ();
    ird.CreatorRoutingNumber = L"123456789";
    ird.DPI = 200;
    ird.GenerateSeparatorPages = false;
    ird.IsQualifiedReturn = false;
    ird.ItemRange = Range (0, MAXINT32);
    ird.nUp = 3;
    ird.ReverseFrontAndBack = false;
    ird.Void = false;

    if (outputDevice->IsPrinter)
    {
        auto printer = (x9Printer*)outputDevice.get ();

        ird.BackSideOffsets = PointF (0, 0);
        ird.BackSideScaling = PointF (1, 1);
        ird.FrontSideOffsets = PointF (0, 0);
        ird.FrontSideScaling = PointF (1, 1);
        ird.MainTray = x9Tray::get_Default ();
        ird.PrintDensity = 0;
        ird.PrinterType = GENERIC;
        ird.SeparatorTray = x9Tray::get_Default ();
    }
    else
    {
        auto file = (x9File*)outputDevice.get ();

        ird.OutputToFileDestination = std::wstring (L"test.") + file->Extension;
        ird.GenerateSemaphoreFiles = false;
    }

    try
    {
        outputDevice.Print (ird);
        printf ("success\n");
    }
    catch (x9Exception& e)
    {
        printf ("%ws\n", e.Message.c_str ());
    }
}
```

C# Example:

```
using System;
using System.Drawing;
using AllMyPapers.IRDPrint;

public
void
Example (
    x9Device outputDevice,
    string iclFile
)
{
    var ird = new x9IRD ("test.937");

    ird.BundleRange = new Point (0, int.MaxValue);
    ird.CashLetterRange = new Point (0, int.MaxValue);
    ird.CreationDate = DateTime.Now;
    ird.CreatorRoutingNumber = "123456789";
    ird.DPI = 200;
    ird.GenerateSeparatorPages = false;
    ird.IsQualifiedReturn = false;
    ird.ItemRange = new Point (0, int.MaxValue);
    ird.nUp = 3;
    ird.ReverseFrontAndBack = false;
    ird.Void = false;

    if (outputDevice is x9Printer)
    {
        var printer = outputDevice as x9Printer;

        ird.BackSideOffsets = new PointF (0, 0);
        ird.BackSideScaling = new PointF (1, 1);
        ird.FrontSideOffsets = new PointF (0, 0);
        ird.FrontSideScaling = new PointF (1, 1);
        ird.MainTray = x9Tray.Default;
        ird.PrintDensity = 0;
        ird.PrinterType = PrinterType.Generic;
        ird.SeparatorTray = x9Tray.Default;
    }
    else
    {
        var file = outputDevice as x9File;

        crd.OutputToFileDestination = "test." + file.Extension;
        ird.GenerateSemaphoreFiles = false;
    }

    try
    {
        outputDevice.Print (ird);
        Console.WriteLine ("success");
    }
    catch (x9Exception e)
    {
        Console.WriteLine (e.Message);
    }
}
```

X9IRD ()

Initializes a new instance of the x9IRD class.

C++

```
x9IRD:: x9IRD (
    __in x9String file,
)
```

C#

```
public
x9IRD (
    string file,
)
```

Parameters

Name	Description
file	The source ICL file.

BUNDLERANGE

Gets or sets the 1 based range of bundles that should be printed.

C++

```
__declspec (property (get=get_BundleRange, put=set_BundleRange)) Range BundleRange;

Range
get_BundleRange (
)
    const;

void
set_BundleRange (
    __in Range value
);
```

C#

```
public Range BundleRange { get; set; }
```

Property Value

Type: Range
Default Value: Range (1, 2^{32} – 1)

CASHLETTERRANGE

Gets or sets the 1 based range of cash letters that should be printed.

C++

```
__declspec (property (get=get_CashLetterRange, put=set_CashLetterRange)) Range CashLetterRange;

Range
get_CashLetterRange (
)
    const;

void
```

```
set_CashLetterRange (
    __in Range value
);
```

C#

```
public Range CashLetterRange { get; set; }
```

Property Value

Type: Range
Default Value: Range (1, 2^{32} – 1)

CREATIONDATE

Gets or sets a DateTime object that describes the creation date.

C++

```
__declspec (property (get=get_CreationDate, put=set_CreationDate)) DateTime CreationDate;

DateTime
get_CreationDate (
)
    const;

void
set_CreationDate (
    __in DateTime value
);
```

C#

```
public DateTime CreationDate { get; set; }
```

Property Value

Type: DateTime
Default Value: DateTime.Now

CREATORROUTINGNUMBER

Gets or sets the routing number for the creator institution.

C++

```
__declspec (property (get=get_CreatorRoutingNumber, put=set_CreatorRoutingNumber)) x9String
CreatorRoutingNumber;

x9String
get_CreatorRoutingNumber (
)
    const;

void
set_CreatorRoutingNumber (
    __in x9String value
);
```

C#

```
public string CreatorRoutingNumber { get; set; }
```

Property Value

```
Type: string
Value: A valid routing number.
```

DPI

Gets or sets the resolution to use when rendering images in dots per inch.

C++

```
__declspec (property (get=get_DPI, put=set_DPI)) int DPI;

int
get_DPI (
)
    const;

void
set_DPI (
    __in int value
);
```

C#

```
public int DPI { get; set; }
```

Property Value

```
Type: integer
Values: { 75, 100, 150, 200, 300, 600 }
Default Value: 200 dots per inch
```

FILE

Gets the name of the source ICL file that is associated with the object.

C++

```
__declspec (property (get=get_File)) string File;

x9String
get_File (
)
    const;
```

C#

```
public string File { get; }
```

Property Value

```
Type: string
```

GENERATESEMAPHOREFILES

If set to true and printing to file, a semaphore file will be created to mark the completion of the printing operation. This file will have the same name and be in the same location as that specified in OutputToFileDestination and will have the ".SEM" extension appended to it.

C++

```
__declspec (property (get=get_GenerateSemaphoreFiles, put=set_GenerateSemaphoreFiles)) bool
GenerateSemaphoreFiles;

bool
get_GenerateSemaphoreFiles (
)
    const;

void
set_GenerateSemaphoreFiles (
    __in bool value
);
```

C#

```
public bool GenerateSemaphoreFiles { get; set; }
```

Property Value

```
Type: bool
Default Value: false
```

GENERATESEPARATORPAGES

Gets or sets a value that determines whether separator pages will be inserted between bundles. If true, separator pages will be generated using the SeparatorTray from the x9Printer device. This value has no effect when the output device is a file.

C++

```
__declspec (property (get=get_GenerateSeparatorPages, put=set_GenerateSeparatorPages)) bool
GenerateSeparatorPages;

bool
get_GenerateSeparatorPages (
)
    const;

void
set_GenerateSeparatorPages (
    __in bool value
);
```

C#

```
public bool GenerateSeparatorPages { get; set; }
```

Property Value

```
Type: bool
```

Default Value: false

ISQUALIFIEDRETURN

Gets or sets a value that determines whether Image CashLetter Files that contain returns are printed as qualified returns or not.

C++

```
__declspec (property (get=get_IsQualifiedReturn, put=set_IsQualifiedReturn)) bool IsQualifiedReturn;

bool
get_IsQualifiedReturn (
)
    const;

void
set_IsQualifiedReturn (
    __in bool value
);
```

C#

```
public bool IsQualifiedReturn { get; set; }
```

Property Value

Type: bool
Default Value: false

ITEMRANGE

Gets or sets the 1 based range of items that should be printed.

C++

```
__declspec (property (get=get_ItemRange, put=set_ItemRange)) Range ItemRange;

Range
get_ItemRange (
)
    const;

void
set_ItemRange (
    __in Range value
);
```

C#

```
public Range ItemRange { get; set; }
```

Property Value

Type: Range
Default Value: Range (1, 2^{32} – 1)

NUP

Gets or sets the number of items that will be printed per page. This value, combined with the IsQualifiedReturn value determine the page size that is used for rendering.

If IsQualifiedReturn is false, the job is rendered on letter size paper; otherwise

nUp	Page Dimensions
1	8.5 x 4.292
2	8.5 x 8.5
3	8.5 x 13
4	8.5 x 17

C++

```
__declspec (property (get=get_nUp, put=set_nUp)) int nUp;

int
get_nUp (
)
    const;

void
set_nUp (
    __in int value
);
```

C#

```
public int nUp { get; set; }
```

Property Value

```
Type: integer
Default Value: 3
```

REVERSEFRONTANDBACK

Gets or sets a value that determines whether the front and back sides of the CRD pages will be reversed.

C++

```
__declspec (property (get=get_ReverseFrontAndBack, put=set_ReverseFrontAndBack)) bool ReverseFrontAndBack;

bool
get_ReverseFrontAndBack (
)
    const;

void
set_ReverseFrontAndBack (
    __in bool value
);
```

C#

```
public bool ReverseFrontAndBack { get; set; }
```

Property Value

```
Type: bool
Default Value: false
```

VOID

Gets or sets a value that determines whether a “VOID” watermark will be rendered on top of the front image for each item.

C++

```
__declspec (property (get=get_Void, put=set_Void)) bool Void;

bool
get_Void (
)
    const;

void
set_Void (
    __in bool value
);
```

C#

```
public bool Void { get; set; }
```

Property Value

```
Type: bool
Default Value: false
```

WORKDIRECTORY

Gets or sets a directory path which be used as a working directory when creating the CRD data.

C++

```
__declspec (property (get=get_WorkDirectory, put=set_WorkDirectory)) x9String WorkDirectory;

x9String
get_WorkDirectory (
)
    const;

void
set_WorkDirectory (
    __in x9String value
);
```

C#

```
public string WorkDirectory { get; set; }
```

Property Value

```
Type: string
Default Value: The user's temporary directory.
```

X9PRINTER

An interface to an output device that renders on a physical printer.

Namespace: AllMyPapers::IRDPrint

Inheritance Heirarchy

AllMyPapers::IRDPrint::x9Device
  AllMyPapers::IRDPrint::x9Printer

Properties

	Name	Description
	CanWatermark	If true, the device is able to render watermarks (See x9Device).
	Description	Gets a textual description of the device (See x9Device).
	Extension	Gets the default extension used when printing to file(See x9Device).
S	Default	Returns an x9Printer for the default system printer.
	Forms	Gets a list of forms that are supported(See x9Device).
S	Installed	Gets a list of installed devices(See x9Device).
	IsFile	Returns false for x9Printer objects **(C++ only)** (See x9Device).
	IsPrinter	Returns true for x9Printer objects **(C++ only)** (See x9Device).
	LeftMargin	Gets the physical left margin of the device, in inches.
	SupportsDuplex	If true, the device supports duplex output.
	TopMargin	Gets the physical top margin of the device, in inches.
	Trays	A list of x9Tray objects, one for each input tray on the device.

C++ Example:

```
#include "irdprint.h"

using namespace System;
using namespace System::Drawing;
using namespace AllMyPapers::IRDPrint;

void
Example (
)
{
    auto outputDevices = x9Device::Installed;

    for (auto p = outputDevices.begin () ; p != outputDevices.end () ; p++)
    {
        auto& outputDevice = (*p);

        if (outputDevice->IsPrinter)
        {
            auto printer = (x9Printer*)outputDevice.get ();

            printf ("%ws\n", (LPCWSTR) printer->Description.c_strW ());
```

```
            printf ("  CanWatermark: %ws\n", printer->CanWatermark ? L"true" : L"false");
            printf ("  Extension: %ws\n", (LPCWSTR) printer->Extension.c_strW ());
            printf ("  IsPrinter: %ws\n", printer->IsPrinter ? L"true" : L"false");
            printf ("  IsFile: %ws\n", printer->IsFile ? L"true" : L"false");
            printf ("  LeftMargin: %f\n", printer->LeftMargin);
            printf ("  SupportsDuplex: %ws\n", printer->SupportsDuplex ? L"true" : L"false");
            printf ("  TopMargin: %f\n", printer->TopMargin);
            printf ("  Forms:\n");

            auto& forms = printer->Forms;

            for (auto p = forms.begin (), p != forms.end () ; ++p)
            {
                auto& f = (*p);

                printf ("    %ws (%f x %f)\n", (LPCWSTR)p->Name.c_strW (), p->Width, p->Height);
            }
        }
    }
}
```

C# Example:

```
using System;
using System.Drawing;
using AllMyPapers.IRDPrint;

public
void
Example (
)
{
    var outputDevices = x9Device.Installed;

    foreach (var outputDevice in outputDevices)
    {
        if (outputDevice is x9Printer)
        {
            var printer = outputDevice as x9Printer;

            Console.WriteLine (printer.Description);
            Console.WriteLine ("  CanWatermark: " + printer.CanWatermark);
            Console.WriteLine ("  Extension: " + printer.Extension);
            Console.WriteLine ("  LeftMargin: " + printer.LeftMargin);
            Console.WriteLine ("  SupportsDuplex: " + printer.SupportsDuplex);
            Console.WriteLine ("  TopMargin: " + printer.TopMargin);
            Console.WriteLine ("  Forms:");

            var forms = printer.Forms;

            foreach (var f in forms)
            {
                Console.WriteLine ("    " + f.Name + " (" + f.Width + " x " + f.Height + ")");
            }
        }
    }
}
```

DEFAULT

A static property that gets the default x9Printer on the machine.

C++

```
static
DEVICE_PTR
get_Default (
);
```

C#

```
static x9Printer Default { get; }
```

Property Value

Type: x9Printer

LEFTMARGIN

Gets the physical left margin defined by the device, in inches.

C++

```
__declspec (property (get=get_LeftMargin)) float LeftMargin;

virtual
float
get_LeftMargin (
)
        const = 0;
```

C#

```
abstract float LeftMargin { get; }
```

Property Value

Type: float
Value: left margin, in inches

SUPPORTSDUPLEX

Gets a value that specifies whether the device supports duplex operations.

C++

```
__declspec (property (get=get_SupportsDuplex)) bool SupportsDuplex;

virtual
bool
get_SupportsDuplex (
)
        const = 0;
```

C#

```
abstract bool SupportsDuplex { get; }
```

Property Value

Type: bool

Value: **true** if the device supports duplex; otherwise **false**

TOPMARGIN

Gets the physical top margin defined by the device, in inches.

C++

```
__declspec (property (get=get_TopMargin)) float TopMargin;

virtual
float
get_TopMargin (
)
        const = 0;
```

C#

```
abstract float TopMargin { get; }
```

Property Value

Type: float
Value: top margin, in inches

TRAYS

Gets the list of input paper trays on the device.

C++

```
__declspec (property (get=get_Trays)) TRAY_LIST Trays;

virtual
TRAY_LIST
get_Trays (
)
    const = 0;
```

C#

```
abstract List<x9Tray> Trays { get; }
```

Property Value

Type: List<x9Tray> (C#) or TRAY_LIST (C++)

X9RNL

The object used to output a Return Notification Letter (RNL).

Namespace: AllMyPapers::IRDPrint

Inheritance Heirarchy

AllMyPapers::IRDPrint::x9BASE

AllMyPapers::IRDPrint::x9IRD
AllMyPapers::IRDPrint::x9RNL

Constructors

	Name	Description
	x9RNL (string)	Initializes a new instance of the x9RNL class.

Properties

	Name	Description
	BundleRange	The range of bundles to print (See x9IRD).
	BackSideOffsets	Offsets to correct the positioning of the back side, in inches (See x9BASE).
	BackSideScaling	Scale factors to correct the sizing of the back side (See x9BASE).
	CashLetterRange	The range of cash letters to print (See x9IRD).
	CreationDate	The creation date of the document (See x9IRD).
	CreatorRoutingNumber	The routing number for the creating institution (See x9IRD).
	DPI	The image resolution, in dots per inch (See x9IRD).
	File	The name of the ICL file associated with the object (See x9IRD).
	FrontSideOffsets	Offsets to correct the positioning of the front side, in inches (See x9BASE).
	FrontSideScaling	Scale factors to correct the sizing of the front side (See x9BASE).
	GenerateSemaphoreFiles	If true, semaphore files are generated on output to file (See x9IRD).
	GenerateSeparatorPages	If true, separator pages are inserted between bundles (See x9IRD).
	IsQualifiedReturn	If true, returns are rendered as qualified returns (See x9IRD).
	ItemRange	The range of items to print (See x9IRD).
	MainTray	An x9Tray object that describes the input tray for the job (See x9BASE).
	nUp	The number of items per page (See x9IRD).
	OutputToFileDestination	The fully qualified path of the file when outputting to file (See x9BASE).
	PrintDensity	The density of the toner (from 0 – 5) (See x9BASE).
	PrinterType	The type of printer (used for tracking internal fonts) (See x9BASE).
	ReverseFrontAndBack	If true, the front and back sides of each page are switched (See x9IRD).
	SeparatorTray	An x9Tray object that describes the input tray for separator pages (See x9BASE).
	VOID	If true, a watermark is rendered over the front side images (See x9IRD).
	WorkDirectory	The directory for storing temp files during printing (See x9IRD).

C++ Example:

```
#include "irdprint.h"

using namespace System;
using namespace System::Drawing;
using namespace AllMyPapers::IRDPrint;

void
Example (
    __in DEVICE_PTR outputDevice,
    __in std::wstring iclFile
```

```
)
{
    x9RNL rnl (L"test.937");

    rnl.BundleRange = Range (0, MAXINT32);
    rnl.CashLetterRange = Range (0, MAXINT32);
    rnl.CreationDate = DateTime::Now ();
    rnl.CreatorRoutingNumber = L"123456789";
    rnl.DPI = 200;
    rnl.GenerateSeparatorPages = false;
    rnl.IsQualifiedReturn = false;
    rnl.ItemRange = Range (0, MAXINT32);
    rnl.nUp = 3;
    rnl.ReverseFrontAndBack = false;
    rnl.Void = false;

    if (outputDevice->IsPrinter)
    {
        auto printer = (x9Printer*)outputDevice.get ();

        rnl.BackSideOffsets = PointF (0, 0);
        rnl.BackSideScaling = PointF (1, 1);
        rnl.FrontSideOffsets = PointF (0, 0);
        rnl.FrontSideScaling = PointF (1, 1);
        rnl.MainTray = x9Tray::get_Default ();
        rnl.PrintDensity = 0;
        rnl.PrinterType = GENERIC;
        rnl.SeparatorTray = x9Tray::get_Default ();
    }
    else
    {
        auto file = (x9File*)outputDevice.get ();

        rnl.OutputToFileDestination = std::wstring (L"test.") + file->Extension;
        rnl.GenerateSemaphoreFiles = false;
    }

    try
    {
        outputDevice.Print (rnl);
        printf ("success\n");
    }
    catch (x9Exception& e)
    {
        printf ("%ws\n", e.Message.c_str ());
    }
}
```

C# Example:

```
using System;
using System.Drawing;
using AllMyPapers.IRDPrint;

public
void
Example (
    x9Device outputDevice,
    string iclFile
)
{
    var rnl = new x9RNL ("test.937");

    rnl.BundleRange = new Point (0, int.MaxValue);
    rnl.CashLetterRange = new Point (0, int.MaxValue);
```

```
    rnl.CreationDate = DateTime.Now;
    rnl.CreatorRoutingNumber = "123456789";
    rnl.DPI = 200;
    rnl.GenerateSeparatorPages = false;
    rnl.IsQualifiedReturn = false;
    rnl.ItemRange = new Point (0, int.MaxValue);
    rnl.nUp = 3;
    rnl.ReverseFrontAndBack = false;
    rnl.Void = false;

    if (outputDevice is x9Printer)
    {
        var printer = outputDevice as x9Printer;

        rnl.BackSideOffsets = new PointF (0, 0);
        rnl.BackSideScaling = new PointF (1, 1);
        rnl.FrontSideOffsets = new PointF (0, 0);
        rnl.FrontSideScaling = new PointF (1, 1);
        rnl.MainTray = x9Tray.Default;
        rnl.PrintDensity = 0;
        rnl.PrinterType = PrinterType.Generic;
        rnl.SeparatorTray = x9Tray.Default;
    }
    else
    {
        var file = outputDevice as x9File;

        rnl.OutputToFileDestination = "test." + file.Extension;
        rnl.GenerateSemaphoreFiles = false;
    }

    try
    {
        outputDevice.Print (rnl);
        Console.WriteLine ("success");
    }
    catch (x9Exception e)
    {
        Console.WriteLine (e.Message);
    }
}
```

X9RNL ()

Initializes a new instance of the x9RNL class.

C++

```
x9RNL:: x9RNL (
    __in x9String file,
)
```

C#

```
public
x9RNL (
    string file,
)
```

Parameters

Name	Description
file	The source ICL file.

X9TRAY

A read-only class that describes an input paper tray on a printer.

Namespace: AllMyPapers::IRDPrint

Properties

	Name	Description
	Default	The default tray.
	Description	A description of the tray.

C++ Example:

```
#include "irdprint.h"

using namespace AllMyPapers::IRDPrint;

public
void
Example (
    __in DEVICE_PTR outputDevice
)
{
    if (outputDevice->IsPrinter)
    {
        auto printer = (x9Printer*)outputDevice.get ();
        auto trays = printer->Trays;

        for (auto p = trays.begin () ; p != trays.end () ; ++p)
        {
            printf (“%hs\n”, (LPCSTR)p->Description.c_strA ());
        }
    }
}
```

C# Example:

```
using AllMyPapers.IRDPrint;

public
void
Example (
    x9Device outputDevice
)
{
    if (outputDevice is x9Printer)
    {
        auto printer = outputDevice as x9Printer;
        auto trays = printer.Trays;

        foreach (var p in trays)
        {
            Console.WriteLine (p.Description);
        }
    }
 }
```

DEFAULT

Gets a x9Tray object that describes the default input tray on the printer.

C++

```
static
TRAY_PTR
get_Default (
);
```

C#

```
static x9Tray Default { get; }
```

Property Value

Type: x9Tray

DESCRIPTION

Gets a description of the tray.

C++

```
__declspec (property (get=get_Description)) x9String Description;

virtual
x9String
get_Description (
)
    const = 0;
```

C#

```
abstract string Description { get; }
```

Property Value

Type: string

APPENDIX

AMPALIGNMENT

Describes the alignment of text elements either within the specified bounding rectangle or relative to other text elements.

C++

```
enum
class
ampALIGNMENT
{
    None        = 0,
    Near        = 1,
    Center      = 2,
    Far         = 3
```

```
};
```

C#

```
public
enum
ampALIGNMENT
{
    None        = 0,
    Near        = 1,
    Center      = 2,
    Far         = 3
}
```

Values

Name	Description
None	The value has not been set.
Near	The element will be aligned on the near side of the bounding rectangle. In a left-to-right situation, the near position is left.
Center	The element will be centered in the bounding rectangle.
Far	The element will be aligned on the far side of the bounding rectangle. In a left-to-right situation, the far position is right.

AMPFONT_ORIENTATION

Describes the orientation of text elements.

C++

```
enum
class
ampFONT_ORIENTATION
{
    Portrait        = 1,
    Landscape       = 2
};
```

C#

```
public
enum
ampFONT_ORIENTATION
{
    Portrait        = 1,
    Landscape       = 2
}
```

Values

Name	Description
Portrait	The text is oriented vertically.
Landscape	The text is oriented horizontally.

AMPFONT_SPACING

Describes the spacing of the characters within the text element.

C++

```
enum
class
ampFONT_SPACING
{
    Fixed           = 1,
    Proportional    = 2
};
```

C#

```
public
enum
ampFONT_SPACING
{
    Fixed           = 1,
    Proportional    = 2
}
```

Values

Name	Description
Fixed	The characters are spaced evenly.
Proportional	The characters are spaced according to their actual width.

AMPFONT_STROKE

Describes the weight of the font.

C++

```
enum
class
ampFONT_STROKE
{
    Light           = 1,
    Medium          = 2,
    Dark            = 3
};
```

C#

```
public
enum
ampFONT_STROKE
{
    Light           = 1,
    Medium          = 2,
    Dark            = 3
}
```

Values

Name	Description
Light	The characters are lighter than normal.

Medium	This is the default.
Dark	The characters are bold.

AMPFONT_STYLE

Describes the angle of the text.

C++

```
enum
class
ampFONT_STYLE
{
    Upright         = 1,
    Italic          = 2
};
```

C#

```
public
enum
ampFONT_STYLE
{
    Upright         = 1,
    Italic          = 2
}
```

Values

Name	Description
Upright	The characters are rendered normally.
Italic	The characters are italicized.

AMPPOSITION

Describes the position of the image within the bounding rectangle.

C++

```
enum
class
ampPOSITION
{
    NONE = -1,

    UpperLeft,
    UpperRight,
    Center,
    LowerLeft,
    LowerRight
};
```

C#

```
public
enum
ampPOSITION
{
    NONE = -1,
```

```
    UpperLeft,
    UpperRight,
    Center,
    LowerLeft,
    LowerRight
}
```

Values

Name	Description
NONE	The setting was not set.
UpperLeft	The image is positioned in the upper left.
UpperRight	The image is positioned in the upper right.
Center	The image is centered.
LowerLeft	The image is positioned in the lower left.
LowerRight	The image is positioned in the lower right.

AMPROTATION

Describes the rotation of the image.

C++

```
enum
class
ampROTATION
{
    Angle0 = 0,
    Angle90 = 90,
    Angle180 = 180,
    Angle270 = 270
};
```

C#

```
public
enum
ampROTATION
{
    Angle0 = 0,
    Angle90 = 90,
    Angle180 = 180,
    Angle270 = 270
}
```

Values

Name	Description
Angle0	The image is not rotated.
Angle90	The image is rotated 90 degrees counter clockwise.
Angle180	The image is rotated 180 degrees.
Angle270	The image is rotated 90 degrees clockwise.

AMPSCALE

Describes the scaling of the image.

C++

```
enum
class
ampSCALE
{
    NONE = -1,

    OS_PRESERVE = 0,
    OS_BEST_FIT = 1,
    OS_FIT_HEIGHT = 2,
    OS_FIT_WIDTH = 3,
    OS_CUSTOM = 4,
    OS_NONE = 5,
    OS_COMBO = 6,

    NS_NONE = 256,
    NS_BEST_FIT = 257,
    NS_FIT_HEIGHT = 258,
    NS_FIT_WIDTH = 259,
    NS_CUSTOM = 260,
    NS_FIT_BOTH = 261
};
```

C#

```
public
enum
ampSCALE
{
    NONE = -1,

    OS_PRESERVE = 0,
    OS_BEST_FIT = 1,
    OS_FIT_HEIGHT = 2,
    OS_FIT_WIDTH = 3,
    OS_CUSTOM = 4,
    OS_NONE = 5,
    OS_COMBO = 6,

    NS_NONE = 256,
    NS_BEST_FIT = 257,
    NS_FIT_HEIGHT = 258,
    NS_FIT_WIDTH = 259,
    NS_CUSTOM = 260,
    NS_FIT_BOTH = 261
}
```

Values

Name	Description
OS_PRESERVE	Preserve the linear dimension. If the image is larger than the bounding rectangle, the image will be centered and the excess clipped to fit; otherwise the image will be anchored in the upper left corner.
OS_BEST_FIT	The image will be scaled so that it completely spans the bounding rectangle in one dimension and the aspect ratio will be preserved. The resulting image will be anchored in the upper left corner.

OS_FIT_HEIGHT	Fit the height of the image to the height of the bounding rectangle and preserve the aspect ratio. Any excess image will be clipped at the right edge. The resulting image will be anchored in the upper left corner.
OS_FIT_WIDTH	Fit the width of the image to the width of the bounding rectangle and preserve the aspect ratio. Any excess image will be clipped at the bottom edge. The resulting image will be anchored in the upper left corner.
OS_CUSTOM	Specifies that custom scale factors will be provided. The resulting image will be anchored in the upper left corner and any excess will be clipped at the right and bottom edges.
OS_NONE	Preserve the linear dimension and anchor the image in the upper left corner. If necessary, the excess will be clipped at the right and bottom edges.
OS_COMBO	Preserve the linear dimension if the image fits in the bounding rectangle; otherwise scale to best fit so that it completely spans the bounding rectangle in one dimension and the aspect ratio gets preserved.
NS_NONE	Do not scale.
NS_BEST_FIT	The image will be scaled so that it completely spans the bounding rectangle in one dimension and the aspect ratio will be preserved. The image's position will be specified separately.
NS_FIT_HEIGHT	Fit the height of the image to the height of the bounding rectangle and preserve the aspect ratio. The image's position will be specified separately.
NS_FIT_WIDTH	Fit the width of the image to the width of the bounding rectangle and preserve the aspect ratio. The image's position will be specified separately.
NS_CUSTOM	Specifies that custom scale factors will be provided. The image's position will be specified separately.
NS_FIT_BOTH	Stretch the height and width of the image to fit the bounding rectangle without preserving the aspect ratio. The image's position will be specified separately.

PRINTERTYPE

Describes the type of printer. This enumeration is used to assist the library in understanding which fonts might already be installed on the printer.

C++

```
enum
class
ePrinterType
{
    GENERIC,
    TROY,
    OCE,
    XEROX
};
```

C#

```
public
enum
PrinterType
{
    GENERIC,
    TROY,
    OCE,
```

```
    XEROX
}
```

Values

Name	Description
GENERIC	This is the default.
TROY	Set for Troy printers.
OCE	Set for OCE printers.
XEROX	Set for XEROX printers.