Capture Software Development Kit

OmniPage Capture SDKRelease Notes

Release Notes

Version: 21.0.0

Date: 2019-10-01

KOFAX

Table of Contents

Chapter 1

System requirements

System requirements for computers used to develop a Capture SDK-enabled application

Supported operation systems:

- Windows Server 2008 R2
- Windows 7 x86/x64
- Windows Server 2012/R2
- Windows 8/8.1 x86/x64
- Windows 10 x86/x64
- Windows Server 2016
- Windows Server 2019

Note CSDK runs on Server type operating systems when the Server license feature is present in the CSDK license.

Windows RT is not supported in this release.

- Intel Core (higher CPU is recommended)
- 4 GB minimum RAM (6 GB recommended, more for working with grayscale or color images and more for multithreaded applications)
- 4 GB free disk space

System requirements for computers used to run a Capture SDK-enabled application

Supported operation systems:

- Windows Server 2008 R2
- Windows 7 x86/x64
- Windows Server 2012/R2, also Core
- Windows 8/8.1 x86/x64
- Windows 10 x86/x64
- Windows Server 2016, also Core in Docker containers

Note CSDK runs on Server type operating systems when the Server license feature is present in the CSDK license.

Windows RT is not supported in this release.

- Intel Core (higher CPU is recommended

- 2 GB minimum RAM (4 GB recommended, more for working with grayscale or color images and more for multithreaded applications)
- 450 MB free hard disk space (less if not all recognition modules are distributed)

Chapter 2

Getting started

Read the Licensing changes in General Information help OPLicMgr

RecAPI help

Objects help

IWR guide

Chapter 3

User development tools

The following development tools are considered as 'fully supported' by the Capture SDK.

- Microsoft Visual Studio 2010-SP1
- Microsoft Visual Studio 2012
- Microsoft Visual Studio 2013.4
- Microsoft Visual Studio 2015.3
- **Microsoft Visual Studio 2017 (used to produce the Capture SDK)**
- Microsoft Visual Studio 2019

Other (not tested) development environments with COM and / or .NET Client 4.6 support should also work with Capture SDK.

Chapter 4

Known issues

- On Windows 7, the OS may show a window about the CSDK setup.exe as being from Unknown publisher. It is a Windows 7 issue.
- Windows 7 Update KB4284842 required

  The fix is a part of the June 21, 2018—KB4284842 – which is already available through windows update as an optional package. https://support.microsoft.com/en-us/help/4284842/windows-7-update-kb4284842

Chapter 5

Version 21.0

After the setup, start OPLicMgrUI.exe from the Start Menu - OmniPage and use it to activate your license.

Changes and new features

Screen capture OCR accuracy Improvement

CSDK contains a new preprocessing algorithm for recognizing screen captures.If the images to be processed can include screen capture you need to set Kernel.Img.ScreenCapture setting to SCR_AUTO or SCR_YES.SRC_AUTO is suggested because in this case CSDK analyzes the image and uses the optimal algorithm.

Objects API for .NET Core to support Docker container environments on Windows (and later Linux)

Customers should use Kofax.OmniPageCSDK.NETCore.Objects.dll Kofax.OmniPageCSDK.NETCore.CAPI.dll, Kofax.OmniPageCSDK.NETCore.RecPDF.dll assemblies in their .NET Core applications. .NET Core v2.1.

For CSDK usage in Docker, check the Technical Note container section.

RecAPI for Java

JNI implementation of the RecAPI. The current implementation is based on Java version 8.

csdk_omnipage.jar and csdk_omnipage.dll are located in bin folder.

The complete Java Sample are located in `Samples\JavaSamples`. Build.cmd builds the sample, use Admin command prompt and run.cmd -x 1 for sample #1.

Essay, New Output Format support

Introduced support for styles and heading levels.

Improved Headers/Footers.

Enable these features and select one of the new Essay mode converter shortcuts (Converters.Text.DocXEssay, .Rtf2000Essay, .Html50Essay) when exporting a document, with the flexibility of already existing export flags.

Essay mode concept is about exporting layout metadata unified at document-level, allowing the user to easily process the exported document further.

Improvement in accuracy

Significant accuracy improvements in mixed alphanumeric text recognition.

Form Template Datarules

Customer can assign data rules (wordlists, regular expressions or logical expressions) to form fields. After the recognition process the form fields will be flagged if they suffice the data rules or not. Customer can quickly verify the erroneous field. The reliability and the reparability of forms increases dramatically with this feature.

Form Template Adaptation

Form Template Editor (FTE) now supports adapting existing form templates to new (empty or filled) form pages with somewhat modified field layout, when the user wants to create a new template based on an existing one. The adaptation process tries to find the new, modified positions of the original fields of the existing template (based on similar field label texts), thus reduces the amount of the required manual editing. The result of this adaptation can still be imperfect, requiring further manual adjustment or finalization. (The adaptation process gives status-info feedback about the failure of finding new position for some individual field-zones in the message-list displayed by the ‘Check Template’ operation.)

Changes in CSDK Licensing

CSDK Installer does not install License Service for CSDK. After the install, you need to use OPLicMgr.exe or OPLicMgrUI.exe from the bin folder to license the CSDK. Seat and OEM licenses should be managed using these tools.

The previous central licensing has been replaced by the new OmniPage Licensing Agent component. OPLA works similar to NLS/NCLS and it is used for page based licenses.

Please study the new licensing information in the documentation.

The default Omnifont OCR engine configuration change to accurate one

The default OCR recognition engine is RM_AUTO. This Engine is a variable and it means 2-way vote OCR on Latin based input images. From v21 it is changed to 3-way vote where it is possible. The default OCR engine is more accurate and slightly slower. If an application requires the previous 2-way voting, it can be changed by setting "Kernel.OcrMgr.PreferAccurateEngine" to false.

The default value of Kernel.Img.Resolution.Restore changed (from FALSE to TRUE)

Assembly and Namespace changes in .NET APIs

In this version .NET assemblies and its namespaces were changed due to the rebranding to Kofax Inc.

The new names are descriptive and easy to use. Application must be changed a bit to refere to the assemblies and namespaces.

Generally Nuance.OmniPage.CSDK.xxx has been changed to Kofax.OmniPageCSDK.xxx. Please check the Samples.

Old IProPlus .NET support API and its Samples are retired and not supported

There is a newer version. The Applications should use the actual IProPlus .NET API, Kofax.OmniPageCSDK.IproPlus.dll and Samples\HowCouldWeUseIt.

Revoked binaries:

```
IproPlus.NET.dll
IproPlus.SRV.NET.dll
MediumWeightVisualsLib.dll
MediumWeightVisuals.NET.dll
VisualsLib.dll
Visuals.NET.dll
```

Note There is a possibility to recreate these old dlls on the customer side like so:

```
tlbimp IproPlus.dll /out:IproPlus.NET.dll /sysarray /namespace:IproPlus
tlbimp IproPlus.exe /out:IproPlus.SRV.NET.dll /sysarray /namespace:IproPlus
tlbimp MediumWeightVisuals.dll /out:MediumWeightVisualsLib.dll /sysarray
aximp MediumWeightVisuals.dll /out:MediumWeightVisuals.NET.dll /
rcw:MediumWeightVisualsLib.dll
tlbimp Visuals.dll /out:VisualsLib.dll /sysarray
aximp Visuals.dll /out:Visuals.NET.dll /rcw:VisualsLib.dll
```

Old RecAPI P/Invoke API (CAPI_PInvoke.dll) is retired
The applications should use the new assemblies, Kofax.OmniPageCSDK.CAPI.dll and Kofax.OmniPageCSDK.ArgTypes.dll.

Old ISAppMFC and VSAppMFC samples are retired

Please study the Samples\HowDoWeUseIt. It is a complete sample for C/C++ applications.

MAX image file support is retired

V21 is not able to load MAX image files.

Chapter 6

Technical notes

The following issues are added to the documentation.

Issue with installing eariler CSDK v20.3 when 21 is already installed

Here is a **workaround** to install CSDK20, when CSDK21 is already installed:

1. Before installing CSDK20, open the file AUTORUN.ini (in the build root) with a text editor for editing.
2. Navigate to the section "[@PREREQUISITE5@]", which is responsible for installing "Visual C++ Redistributable 2015 (x86)".
3. Within the section, navigate to the line starting with "RequireOn=" and delete all the characters in the line AFTER the equal sign.
4. Close the file and launch SETUP.exe again.

.NET Core version of Objects API uses one Nuget package

https://www.nuget.org/packages/System.Drawing.Common/

Partial manual for container

Server core

1. Install Docker on machine that has Hyper-V enabled.
   - On Windows 10 – download and install Docker CE.
   - On windows server – run the following commands in elevated powershell.

     ```
     Install-Module DockerMsftProvider -Force
     Install-Package Docker -ProviderName DockerMsftProvider -Force
     ```
2. Restart the PC.
3. Verify that docker is in Windows mode – run command `docker info` to make sure that `OSType: Windows`
4. Verify that docker is working properly by running command `docker run hello-world:nanoserver`

5. Download base images: `docker pull microsoft/nanoserver` and `docker pull microsoft/windowsservercore`
6. Prepare a folder from where you will create an image. It should contain:
   - Program folder, which contains file the set created by Distribution Wizard (64-bit) after selecting all necessary APIs and modules
     - Inside Program there should be your application and inputs
   - Dockerfile:

```
# escape=`
FROM mcr.microsoft.com/windows/servercore:1903
RUN powershell md c:\Program
COPY vcredist_x64.exe c:\
RUN powershell.exe -Command Start-Process C:\vcredist_x64.exe -ArgumentList '/
quiet' -Wait
COPY Program c:\Program
CMD C:\Program\yourprogramname.exe
```

   - Licensing.config file, pointing to OPLA licensing server
   - Vcredist_x64.exe – latest visual C redistributable
7. Open powershell, create a new image by navigating to your folder and running this command (with dot): `Docker build -t csdkcore .`
8. After image is created – run it by issuing command: `Docker run -it csdkcore`

Nano server

On Windows Server Nano, CSDK operates at KernelAPI level. Application can use the Engine with invoking Engine.Init("YourCompany", "YourProduct", **false**), to initialize the Engine only KernelAPI functionality.

1. Install Docker on machine that has Hyper-V enabled.
   - On Windows 10 – download and install Docker CE.
   - On windows server – run the following commands in elevated powershell.

```
Install-Module DockerMsftProvider -Force
Install-Package Docker -ProviderName DockerMsftProvider -Force
```

2. Restart the PC.
3. Verify that docker is in Windows mode – run command `docker info` to make sure that `OSType: Windows`
4. Verify that docker is working properly by running command `docker run hello-world:nanoserver`
5. Download base images: `docker pull mcr.microsoft.com/dotnet/core/runtime:2.2`
6. Prepare a folder from where you will create an image. It should contain Program folder, which contains:
   - Files exported by 64-bit Distribution Wizard (having selected all items but all Visuals, all Scanner, all IPRO and IWR)
   - Your .Net Core application as dll and input files for it
   - Licensing.config file, pointing to OPLA licensing server
   - Dockerfile:

```
# escape=`
FROM mcr.microsoft.com/dotnet/core/runtime:2.2
```

```
RUN md c:\Program
COPY Program c:\Program
COPY CrtForNano_CSDK c:\Program
WORKDIR Program
# CMD dotnet YourProgramName.dll
```

- Windows CRT package in `C:\Program Files (x86)\Microsoft Visual Studio \2017\Enterprise\VC\Redist\MSVC\14.16.27012\onecore\x64`

  List of CRT files needed for Windows Server Nano:

  - concrt140.dll
  - msvcp140.dll
  - msvcp140_1.dll
  - msvcp140_2.dll
  - ucrtbase.dll
  - vccorlib140.dll
  - vcomp140.dll
  - vcruntime140.dll

7. Open PowerShell, create a new image by navigating to your folder and running this command (with dot): `Docker build -t ImageName .`
8. After image is created – run it by issuing command: `Docker run -it ImageName`

V20 IWR .ocrjob files are not compatible with this version

Customers should export their workflows to WorkflowXML format in the Workflow Designer v20. This WorkflowXMLs can be loaded to the Designer V21.Another option is to change Nuance.OmniPage.CSDK to Kofax.OmniPageCSDK in the ocrjob files (XAML).

Vocalizer support is retired

In this release the voice output (Converters.Text.Audiomp3) is not supported.Converters.Text.Audio is available for mp3 output. This export filter utilizes Windows Speech API.

DCTool additional information

- 1. modify and get_def_file commands - project definition file format

  A project definition file(‘definition - file’ parameter in case of the ‘modify’ and ‘get_def_file’ commands) is a text file of JSON format, describing the actual project properties (classes, documents, stopwords, metawords, and so on). Actually the following project - data is serialized (using a struct Project instance as a root - object):

```
struct POINT {
 long x;
 long y;
};
```

```
struct SIZE {
 int cx;
 int cy;
};

// see KernelApi.h!
// NOTE: enum values are serialized now using their 'int' representation!
enum LANGUAGES {
 // ...
};

// NOTE: enum values are serialized now using their 'int' representation!
enum Ngram_weights {
 NGW_NON = -1,
 NGW_PROHIBITED,
 NGW_NORMAL,
 NGW_LEVEL1,
 NGW_LEVEL2,
 NGW_MANDATORY
};

// NOTE: enum values are serialized now using their 'int' representation!
enum DCProjectType {
 DCPT_OPEN = 0,
 DCPT_CLOSED
};

struct MetaWord {
 string pattern;
 string value;
};

typedef string StopWord;

struct Phrase {
 string text;
 string value;
};

struct ClassPhrase : public Phrase {
 Ngram_weights weight;
 POINT location; // serialized as location.x, location.y
 SIZE size; // serialized as size.cx, size.cy
 SIZE drift; // serialized as drift.cx, drift.cy
 int groupId = -1;
};

struct Document {
 string imagePath;
 unsigned page = 0;
 bool isHidden = false;
};

typedef vector<ClassPhrase> ClassPhraseVect;
typedef map<LANGUAGES, ClassPhraseVect> ClassPhrasesMap;

struct Class {
 string name;
 bool isHidden = false;
 vector<Document> documents;
 ClassPhrasesMap classPhrasesMap;
};

typedef set<LANGUAGES> LangSet;
```

```
typedef vector<MetaWord> MetaWordVect;
typedef map<LANGUAGES, MetaWordVect> MetaWordsMap;
typedef vector<StopWord> StopWordVect;
typedef map<LANGUAGES, StopWordVect> StopWordsMap;

struct Project {
 string projectPath;
 DCProjectType projectType = DCPT_CLOSED;
 LangSet langSet;
 vector<Class> classes;
 MetaWordsMap metaWordsMap;
 StopWordsMap stopWordsMap;
};
```

A simplified project definition file sample:

```
{
 "project": {
  "projectPath": "e:\\dc\\test_project\\test.dcproj",
  "projectType" : 1,
  "langSet" : [ 0 ],
  "classes" : [ {
   "name": "BusinessCard",
   "isHidden" : false,
   "documents" : [ {
    "imagePath": "e:\\dc\\test_input\\BusinessCard\\02.jpg",
    "page" : 0,
    "isHidden" : false },{
    "imagePath": "e:\\dc\\test_input\\BusinessCard\\10.jpg",
    "page" : 0,
    "isHidden" : false } ],
   "classPhrasesMap": [ {
    "key": 0,
    "value" : [] } ] },
  {
   "name": "Receipt",
   "isHidden" : false,
   "documents" : [ {
    "imagePath": "e:\\dc\\test_input\\Receipt\\03.jpg",
    "page" : 0,
    "isHidden" : false },{
    "imagePath": "e:\\dc\\test_input\\Receipt\\08.jpg",
    "page" : 0,
    "isHidden" : false } ],
   "classPhrasesMap": [ {
    "key": 0,
    "value" : [] } ] }
  ],
  "metaWordsMap": [
   {
    "key": 0,
    "value" : [
     {
      "pattern": "fax|facsimile",
      "value" : "fax"
     },
     {
      "pattern": "tel|telephone|phone|mobil[e]",
      "value" : "tel"
     }]
   }
  ],
  "stopWordsMap": [
   {
```

```
    "key": 0,
    "value" : [
     "a",
     "about",
     "above"
    ]
   }
  ]
 }
}
```

- 2. test command - optional CSV output

  Using the –csv parameter of the ‘test’ command, we can create a CSV (Comma Separated Values) output file, containing the test result information similar to the format displayed by the DCAssistant tool.We can directly open this CSV file using Excel or other spreadsheet tools.

  Excel and Windows – the role of the configured default list separator character

  Excel uses the default list separator character to separate columns in the CSV - file.The default list separator can be configured here:

  1. Click the Windows Start menu.
  2. Click Control Panel.
  3. Open the Regional and Language Options dialog box.
  4. Click the Formats Tab.
  5. Click Additional settings.
  6. Check the actual value or type a new separator in the List separator box.
  7. Click OK twice.

     The default list - separator used by DCTool is the ‘, ’(comma) character.

  If you want to specify an alternative list separator character for DCTool(e.g.matching to the default list separator character configured in the Control Panel), you can use the –csv_separator parameter of the ‘test’ command.

- 3. test command - optional JSON output

  Using the ‘–json <json - output - file>’ option of the ‘test’ command we can create the test - output data using JSON format (as an alternative to CSV - format).

  Actually the following test - output data is serialized(using a struct TesOutputData instance as a root - object):

```
struct StatData {
 unsigned count = 0;
 float percent = 0.0;
};

struct FileMatchInfo {
 string docName;
 string targetClass;
 string predictedClass;
 int confidence;
 bool isConfident;
 string result;
};

struct ConfusionMatrix {
 struct Row {
```

```
  string className;
  vector<unsigned> numOfDocsVect;
 };
 vector<string> header;
 vector<Row> matrix;
};

struct TotalStatistics {
 unsigned alienNum = 0;
 float falsePosPercentInAliens;
 StatData falsePos;
 StatData falseNegOrRejected; // open project: false negative; closed project:
 rejected
 StatData misclassified;
 StatData totalError;
 StatData correct;
};

struct BestConfidenceThresholdInfo {
 float falseNegativeOrRejectedWeight;
 float falsePositiveWeight;
 float misclassifiedWeight;
 unsigned bestConfidenceThreshold;
};

struct TestOutputData {
 vector<FileMatchInfo> fileMatchInfoVect;
 ConfusionMatrix confusionMatrix;
 TotalStatistics totalStatistics;
 BestConfidenceThresholdInfo bestConfidenceThresholdInfo;
};
```

A simplified JSON test - output sample:

```
{
 "testOutputData": {
  "fileMatchInfoVect": [
   {
    "docName": "02.jpg#1",
    "targetClass" : "BusinessCard",
    "predictedClass" : "BusinessCard",
    "confidence" : 100,
    "isConfident" : true,
    "result" : "Correct"
   },
   {
    "docName": "10.jpg#1",
    "targetClass" : "BusinessCard",
    "predictedClass" : "BusinessCard",
    "confidence" : 100,
    "isConfident" : true,
    "result" : "Correct"
   },
   {
    "docName": "03.jpg#1",
    "targetClass" : "Receipt",
    "predictedClass" : "Receipt",
    "confidence" : 100,
    "isConfident" : true,
    "result" : "Correct"
   },
   {
    "docName": "08.jpg#1",
    "targetClass" : "Receipt",
```

```
   "predictedClass" : "Receipt",
   "confidence" : 100,
   "isConfident" : true,
   "result" : "Correct"
  }
 ],
 "confusionMatrix": {
  "header": [
   "BusinessCard",
   "Receipt",
   "<Rejected>"
  ],
  "matrix" : [
   {
    "className": "BusinessCard",
    "numOfDocsVect" : [
     2,
     0,
     0
    ]
   },
   {
    "className": "Receipt",
    "numOfDocsVect" : [
     0,
     2,
     0
    ]
   },
   {
    "className": "Alien documents",
    "numOfDocsVect" : [
     0,
     0,
     0
    ]
   }
  ]
 },
 "totalStatistics": {
  "alienNum": 0,
  "falsePosPercentInAliens" : 0.0,
  "falsePos" : {
   "count": 0,
   "percent" : 0.0
  },
  "falseNegOrRejected" : {
   "count": 0,
   "percent" : 0.0
  },
  "misclassified" : {
   "count": 0,
   "percent" : 0.0
  },
  "totalError" : {
   "count": 0,
   "percent" : 0.0
  },
  "correct" : {
   "count": 4,
   "percent" : 100.0
  }
 },
  "bestConfidenceThresholdInfo": {
```

```
   "falseNegativeOrRejectedWeight": 1.0,
   "falsePositiveWeight" : 1.0,
   "misclassifiedWeight" : 1.0,
   "bestConfidenceThreshold" : 49
  }
 }
}
```

User Guide for Intelligent Workflow Runner (OCRService)

See in the https://docshield.kofax.com/OmniPageCaptureSDK/en_US/21.0.0_y3b7m3tg1v/help/IntelligentWorkflowRunner/OmniPageCapture_SDKintelligentworkflowrunner/c_overview.html site for details.The typical usage of the OCRService:

- Build the workflow using AssistantApp.exe (or iTest or OmniPage Ultimate 19).
- Save the workflow into a .xwf file.
- Test its running capabilities in iTest (or OmniPage Ultimate 19).
- Convert the xwf file into JobXML using the XWFToXML.exe command line tool.

You should create workflows without naming input and output files. In this case the workflow displays dialog boxes (custom open and save dialogs) to request the actual file names. In the save dialog you can set all output converter specific settings (Converters.Text.<convertername>.<settingname>). Here you can save these settings into a setting file for the xwf to JobXML conversion. After that, use the XWFToXML converter tool and the setting file to create the JobXML file.You can place any additional CSDK settings to the setting file. If /p command line parameters are specified for XWFToXML.exe, the given input and output files are replaced by macros $INPUT_FILE$, $OUTPUT_FILE$.

- Place these JobXML files somewhere in the application folder structure.
- In the Application change the runtime input ($INPUT_FILE$), output ($OUTPUT_FILE$) and response XML ($OUTPUT_XML$) file macros to the actual file names.
- Read JobXML into a string and pass it to the Run() function and wait for the Done event and handle the response XML.

A similar way to use WorkflowXMLDesigner.exe for creating WorkflowXMLs directly is saving them into templated WorkflowXML files. Check the IWR_Designer.wmv.

From a C++ program use `#import "OCRService.exe" named_guids to access OCRService COM`.

Simple .NET C# Application for IWR usage

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace ConsoleApplication1
{
 class Program
 {
  static OCRServiceLib.OCRService OCRService = null;
  static System.Threading.Semaphore sem;
```

```
  static void Main(string[] args)
  {
   try
   {
    Console.WriteLine("OCRServiceLib.OCRService()");
    OCRService = new OCRServiceLib.OCRService();
    // OEM type licensing
    //OCRService.SetLicense("License file", "Key");

    OCRService.Done += OCRService_Done;

    ((OCRServiceLib.IOCRServiceEvents_Event)OCRService).Ping += OCRService_HeartBeat;

    OCRService.Ping(false);

    string WorkflowXml = "";
    WorkflowXml = File.ReadAllText(@"C:\Temp\WorkflowXMLFile1.xml");
    // Delete the output file, naming in the WorkflowXMLFile1.xml
    try { File.Delete(@"C:\Temp\Test1.txt"); } catch (Exception) { };

    sem = new System.Threading.Semaphore(0, 1);

    OCRService.Run(WorkflowXml, "1"); // Asynchronous call

    sem.WaitOne(); // wait for finishing the Job

    OCRService.Done -= OCRService_Done;
    ((OCRServiceLib.IOCRServiceEvents_Event)OCRService).Ping -= OCRService_HeartBeat;
    OCRService = null;

    string output = System.IO.File.ReadAllText(@"C:\Temp\Test1.txt");
    Console.WriteLine(output);
   }
   catch (Exception e)
   {
    Console.WriteLine(e.ToString());
   }
  }

  static void OCRService_Done(string ID, int StartError, Array StartErrors, Array
 RuntimeErrors, Array RuntimeErrorDesciptions, Array ResponseXMLFiles)
  {
   sem.Release(1);
   return;
  }

  static void OCRService_HeartBeat()
  {
   return;
  }
 }
}
```

Simple C++ Application for IWR

```
// OCRJob.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <atlbase.h>
#include <atlcom.h>
#include <atlstr.h>
```

```
#include <msxml2.h>
#pragma comment(lib, "msxml2")

#import "libid:0D6304DA-A877-4E51-AD1F-19C0248B9715" no_namespace named_guids
 no_smart_pointers raw_interfaces_only

class ATL_NO_VTABLE CServiceWithEvents :
 public CComObjectRootEx<CComMultiThreadModel>,
 public IOCRServiceEvents
{
public:
 CServiceWithEvents() : m_hEvent(NULL), m_hr(S_OK)
 {
 }

 BEGIN_COM_MAP(CServiceWithEvents)
  COM_INTERFACE_ENTRY(IOCRServiceEvents)
  COM_INTERFACE_ENTRY_AGGREGATE(IID_IMarshal, m_pUnkMarshaler.p)
 END_COM_MAP()

 DECLARE_GET_CONTROLLING_UNKNOWN()
 HRESULT FinalConstruct()
 {
  return S_OK;
 }
 void FinalRelease()
 {
 }
 STDMETHOD(Done)(BSTR strGUID, long lResult, SAFEARRAY* psaResults, SAFEARRAY*,
 SAFEARRAY *, SAFEARRAY *)
 {
  if (m_strGUID != strGUID)
   return S_OK;
  if (lResult || psaResults)
   m_hr = E_FAIL;
  SetEvent(m_hEvent);
 return S_OK;
 }
 STDMETHOD(Ping)(void)
 {
  return S_OK;
 }

 HRESULT Run(TCHAR* strFile)
 {
  GUID guid = GUID_NULL;
  HRESULT hr = S_OK;
  CComPtr<IXMLDOMDocument2> pXMLDOMDocument;
  CComPtr<IOCRService> pOCRSrvice;
  VARIANT_BOOL bSuccess = VARIANT_TRUE;
  CComBSTR strXML;
  DWORD dwCookie = 0;

  m_strGUID.Empty();
  m_hr = S_OK;
  m_hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
  if (!m_hEvent)
   hr = AtlHresultFromLastError();
  if (SUCCEEDED(hr))
  {
   hr = CoCreateGuid(&guid);
   m_strGUID = CComBSTR(guid);
  }
  if (SUCCEEDED(hr))
```

```
   hr = CoCreateFreeThreadedMarshaler(GetControllingUnknown(), &m_pUnkMarshaler.p);
  if (SUCCEEDED(hr))
   hr = pXMLDOMDocument.CoCreateInstance(CLSID_DOMDocument);
  if (SUCCEEDED(hr))
   hr = pXMLDOMDocument->load(CComVariant(strFile), &bSuccess);
  if (SUCCEEDED(hr))
   hr = pXMLDOMDocument->get_xml(&strXML);
  if (SUCCEEDED(hr))
   hr = pOCRSrvice.CoCreateInstance(CLSID_OCRService, NULL, CLSCTX_LOCAL_SERVER);
  if (SUCCEEDED(hr))
   hr = AtlAdvise(pOCRSrvice, GetControllingUnknown(), IID_IOCRServiceEvents,
 &dwCookie);
  if (SUCCEEDED(hr))
   hr = pOCRSrvice->Run(strXML, CComBSTR(guid));
  if (SUCCEEDED(hr))
  {
   AtlWaitWithMessageLoop(m_hEvent);
   hr = m_hr;
  }
  if (dwCookie)
   AtlUnadvise(pOCRSrvice, IID_IOCRServiceEvents, dwCookie);
  if (m_hEvent)
   CloseHandle(m_hEvent);
   m_pUnkMarshaler.Release();
   return hr;
  }

  CComPtr<IUnknown> m_pUnkMarshaler;
  HANDLE m_hEvent;
  HRESULT m_hr;
  CComBSTR m_strGUID;
};

int _tmain(int argc, _TCHAR* argv[])
{
 if (argc < 2)
  return E_FAIL;
 HRESULT hr = S_OK;
 CoInitialize(NULL);
 {
  CComObjectStack<ExCServiceWithEvents> service;
  hr = service.Run(argv[1]);
 }
 CoUninitialize();

 return hr;
}
```

Redistributable Microsoft file sets for deployments

In each case, the deployment for your own Capture SDK-enabled application requires you to include one of the redistributable file sets from Microsoft. The Distribution Wizard in the Capture SDK provides hints on which merge modules contain the redistributable file set necessary for your particular deployment. This section tries to simplify your options.

As the Capture SDK was built with Microsoft VS 2017, the Distribution Wizard will advise several merge modules of VS 2017. Microsoft Visual C++ 2017 Redistributable Package (x86/x64) –

`vcredist_x86.exe/vcredist_x64.exe xx.x.xxxx.x`, covers all the necessary Microsoft binaries from those merge modules. It can be downloaded from Microsoft internet sites

User written custom workflows in Workflow Runner

Create custom job item (C#)

1. Create a new C# project with 'Class Library' template. Set Target framework to '.NET Framework 4.6'.
2. With Configuration Manager, create new solution platform; 'x86' or 'x64' depending on your deployed CSDK.
3. Add OmniPageSDK.IproPlus.JobItem.dll and OmniPageSDK.IproPlus.dll as reference.
4. Use OmniPage.CSDK.IproPlus namespace.
5. Derive your class from JobItem and implement abstract class.
6. You can find an implementation of Ping and GetResult method in CustomJobItemCS sample.
7. Implement Run method as you wish. You can evaluate your job request accessing XElement public property.

   You can access IproPlus Engine and Document through public properties, do not dispose them.

   Fire a `Progress` event periodically. If you have a lot of work to do, fire `NewObject` event occasionally.

   When your task is finished, fire the `Done` event.
8. Make your class COM visible by adding attributes `ComVisible(true), Guid(" "), ClassInterface(ClassInterfaceType.None).`
9. Implement a static Register function to change COM threading model to apartment model. You can find a sample in CustomJobItemCS sample.
10. Unload project and edit it to add a platform specific post build event.

```
<PropertyGroup Condition="'$(Platform)' == 'x86'">
 <PostBuildEvent>
  call "$(DevEnvDir)..\..\vc\bin\vcvars32.bat"
  regasm "$(TargetPath)" /codebase
 </PostBuildEvent>
</PropertyGroup>
<PropertyGroup Condition="'$(Platform)' == 'x64'">
 <PostBuildEvent>
  call "$(DevEnvDir)..\..\vc\bin\amd64\vcvars64.bat"
  regasm "$(TargetPath)" /codebase
 </PostBuildEvent>
</PropertyGroup>
```

    Do not use 'Register for COM interop' option.
11. Build your project.

Integrate into WorkflowXMLDesigner

1. Create a new C# 'Class Library' project, targeting to '.NET Framework 4.6' or use the project containing your JobItem.
2. Add System.Activities.dll and OmniPageSDK.IproPlus.JobDesign.dll as reference.

3. Add a new class, use OmniPage.CSDK.IproPlus.JobDesign namespace and derive your class from JobItem.
4. In constructor, set clsid field to '{<YOURGUID>}'.
5. Add necessary public properties to your class, these will be edited in WorkflowXMLDesigner.
6. Override CacheMetadata method to check your properties and add validation error if necessary.
7. Override InputXMLContainer property to compose a proper XML snippet, it will be passed to your custom job item.
8. If you want a custom UI, you can implement an ActivityDesigner class and add Designer attribute to your class.
9. Build your project.
10. In WorkflowXMLDesigner select Options/Custom Job Items... and add your assembly to the collection.

Debug your custom jobitem designer

1. Simply set WorkflowXMLDesigner as external program to start debugging.
2. Check composed XML.

Debug your custom jobitem

1. In Task Manager, terminate all OCRServer.exe instances.
2. Set OCRServer.exe to the external program to start debugging.
3. Start debugging.
4. Launch WorkflowXMLDesigner, create a job using your custom job item, and run the job.

Sample WorkflowXML

This Workflow XML has been created by WorkflowXMLDesigner. Comments were added later.

```
<? xml version="1.0" encoding="utf-8"?>
<OCRJob>
 <JobItem clsid = "{14ABDDC9-0DF7-4D1C-9687-0AE5F2AD1C3D}" <!-- Workflow Job Type, load
 recognize, export -->
  id="{fc02eeb3-c985-4bff-965f-6f1464872599}">
  <Type type = "integer" value="IWFT_WFS_DIRECT|CWFF_LOADIMG| CWFF_RECOGNIZE|
 CWFF_EXPORTDOC" />
  <Inputs type = "array" >
   <Input type="string" value="$inputfile1$" />
  </Inputs>
  <Parameters>
   <Parameter parametername = "SP_LDI_DESKEW" type="boolean" value="false" />
   <Parameter parametername = "SP_LDI_PAGEROTATION" type="integer" value="ROT_NO" />
   <Parameter parametername = "SP_RCI_PROFDICT" type="array">
    <Parameter type = "string" value="English Medical Dictionary" />
    <Parameter type = "string" value="English Financial Dictionary" />
    <Parameter type = "string" value="English Legal Dictionary" />
   </Parameter>
  </Parameters>
  <Settings> <!-- Any CSDK Setting could go here -->
```

```
   <Setting settingname = "Kernel.OcrMgr.DefaultRecognitionModule" type="integer"
 value="RM_OMNIFONT_PLUS3W" />
  </Settings>
  <Output type = "string" value="$outputfile1$" />
  <Converter value = "Converters.Text.PDFImageOnText" >
   <Properties >
    <!-- Any CSDK Converter Setting could go here -->
    <Property propertyname = "ColorQuality" type="integer"
 value="R2ID_PDFCOLORQUALITY_LOSLESS" />
    <Property propertyname = "Compatibility" type="integer" value="R2ID_PDF17" />
    <Property propertyname = "Linearized" type="boolean" value="true" />
    <Property propertyname = "UseMRC" type="integer" value="R2ID_PDFMRC_LOSLESS" />
    <Property propertyname = "Compression.UseJBIG2" type="integer" value="true" />
    <Property propertyname = "Compression.UseJPEG2000" type="integer" value="true" />
   </Properties>
  </Converter>
 </JobItem>
 <JobItem clsid = "{69490304-CC87-476C-90C3-58B200F6303F}" <!-- Statistics Job Type,
 depending from Workflow Job Type -->
  id="{66b6e5e1-8fa8-46e0-91f9-6232dedbfa3c}" dependency="{fc02eeb3-
c985-4bff-965f-6f1464872599}">
  <OutputXML type = "string" value="$xmlfile1$" />
 </JobItem>
</OCRJob>
```

How to setup CSDK scanning sub-system in the redistribution

There are three new functions in RSD to be called by the install/uninstall process as delayed custom actions: RsdInstallMsi(), RsdRepairMsi() and RsdUninstallMsi().

All these functions are located in RnRSDu.dll. These functions have to be called from the installed instance of RnRSDu.dll in the folder where it is installed, and **NOT** from the temporary one in the temp folder , otherwise other RSD modules will not be found by RnRSDU.dll.

These functions replace all activities of the installer/uninstaller in regard to all registry entries and files mentioned below. The installer should simply install all files mentioned below in the same folder where the binaries of RSD are installed. All other activities including maintenance of reference counts are the task of these functions. All files belonging to RSD (including the files mentioned below) should be installed in the same folder, but other files of the product might be installed in other folders including the 'main app'. E.g. in case of PPDF the binaries of RSD are located in the **install** folder, while the main app is installed in the `install\bin` folder, therefore delayed custom actions should be called from **RnRSDu.dll** in the `install` folder and **<full path of main app>** should point to the main application in the `install\bin` folder.

int WINAPI RsdInstallMsi(MSIHANDLE hMsi)

It can be called after all binaries of RSD are installed to the destination folder. This function performs all necessary actions for RSD which can only be done in Admin Mode at install time.

The **CustomActionData** should be composed as follows: **<product name>;<full path of main app>;<path of scanner.ini>;<registry key>;<language>**, where

- **<product name>** is the name of the product as the Wizard will show (e.g. "Power PDF Advanced")
- **<full path of main app>** is what it really means (for example, "D:\Program Files (x86)\...\...\bin\....exe")

- **<path of scanner.ini>** can be absolute path, but usually it is a relative path, using one of the following prefixes:
  - “**ROAMING**\” is the user’s roaming data folder
  - “**LOCAL**\” is the user’s local data folder
  - “**COMMON**\” is the common data folder

  (for example, "**LOCAL\Kofax\PDF\V1\Scanner.ini**")
- **<registry key>** is the path of registry entries relative to HKLM\SOFTWARE if it is different from <product name>, otherwise it can be empty (e.g. "Kofax\PDF\V1")
- **<language>** is the 3-letter suffix of one of the RnRsdWizRes_*.dll files. It is treated “ENG” if it is empty.

  Currently the following languages are available: BRA, BUL, CHS, CHT, CZH, DAN, DUT, ENG, FIN, FRE, GER, HUN, ITA, JPN, KOR, NOR, POL, ROM, RUS, SLO, SPA, SWE, TUR, but the list of available languages depends on the file set just installed.

  For example, “ **Power PDF Advanced;D:\Program Files (x86)\...\...\bin\....exe;LOCAL\Nuance\PDF\V1\Scanner.ini;Nuance\PDF\V1;ENG** ”

  The separator character between the components of CustomActionData can be ‘;’ (semicolon) or ‘|’ (vertical bar)

This function performs the following operations:

- Creates the following registry key: HKLM\SOFTWARE\ <product name> or HKLM\SOFTWARE \<registry key> if <registry key> is not empty
- Creates the following values under the key above:
  - BinPath = **<full path of RSD>** without the name of the file (e.g. “ D:\Program Files (x86)\Kofax\Power PDF 20\bin\ ”)
  - DataPath = **<path of scanner.ini>**
  - Language = **<language>**
  - ProductName = **<product name>** if it does not exist already and differs from **<registry key>**
  - _RSD_ = **<number>** for private use of RSD
- Creates the following folder: `<Common AppData>\Kofax\SWizard`
- Copies the following files from the installed binaries to the folder above: **SWizard.xml**, **SWizard.xsd** with version control and reference count (creates or increases reference count)
- Creates the following folder: `<Windows>\PIXTRAN` if ISIS is supported by the product (i.e. **RnISISu.rsd** is installed)
- Copies the **RsdScan.chn** file from the installed binaries with reference count (creates or increases reference count) to the folder above
- Creates the following registry key: HKLM\SOFTWARE\PFU\ScanSnap Extension\<product name> if Fujitsu ScanSnap is supported by the product (when **RnScanSnapU.rsd** is installed)
- Creates the following values under the key above:
  - <Default> = **<full path of main app>**
  - Config = **<path of ScanSnap.ini>** in the same folder as <path of scanner.ini> if it does not contain any prefix above, otherwise the prefix is changed to <Common AppData>.
  - Path = **<full path of main app>** without the name of the file (for example “`D:\Program Files (x86)\Kofax\Power PDF 20\bin\`”)

int WINAPI RsdRepairMsi(MSIHANDLE hMsi)

It can be called when 'Repair' is selected in the installer. This function performs all necessary actions for RSD, which can only be done in Admin Mode at repair time.

The CustomActionData should be composed exactly the same way as for **RsdInstallMsi()**.

This function performs the same operations as **RsdInstallMsi()** except it does not increase reference counts.

int WINAPI RsdUninstallMsi(MSIHANDLE hMsi)

It can be called when 'Uninstall' is selected in the installer before the installed binaries are removed. This function performs all necessary actions for RSD which can only be done in Admin Mode at uninstall time.

The **CustomActionData** should be composed exactly the same way as **RsdInstallMsi()** except it should contain <delete custom setting> instead of <language>:<product name>;<full path of main app>;<path of scanner.ini>;<registry key>;<delete custom settings>, where

<delete custom settings> is **"0"** or empty if custom settings should be preserved, or **"1"** if custom settings should be deleted. Currently it applies only to the **WizConf.xml** file of the Wizard, for example "Power PDF Advanced;D:\Program Files (x86)\Kofax\Power PDF 20\bin\PPdf.exe;LOCAL\Kofax\PDF \V1\Scanner.ini;Kofax\PDF\V1;1"

This function performs the following operations:

- Deletes the following values under the key below:
  BinPath
  DataPath
  Language
  ProductName if it has been created by **RsdInstallMsi()**
  RSD
- Deletes the following registry key: HKLM\SOFTWARE\ **<product name>** or HKLM\SOFTWARE **<registry key>** if it has been created by **RsdInstallMsi()**
- Removes the following files from the folder below: SWizard.xml , SWizard.xsd with reference count (decreases the reference count and deletes the files and the reference count itself if it becomes 0)
- Removes the following file from the folder below: **WizConf.xml** along with **SWizard.xml** if it was removed, and <delete custom settings> is "1"
- Removes the following folder: <Common AppData>\Kofax\SWizard if it is empty
- Removes the following folder: <Common AppData>\Kofax if it is empty
- Removes the following file from the folder below: **RsdScan.chn** with reference count (decreases the reference count, and deletes the file and the reference count itself, if it becomes 0)
- Removes the following folder: <Windows>\PIXTRAN if it is empty
- Removes the following file: **ScanSnap.ini** from the folder where the Config key of HKLM\SOFTWARE \PFU\ScanSnap Extension\ **<product name>** points
- Removes the following folder: the folder where the Config key of HKLM\SOFTWARE\PFU\ScanSnap Extension**<product name>** points if it is empty

- Deletes the following registry key: HKLM\SOFTWARE\PFU\ScanSnap Extension**<product name>** with all of its values