www.bytescout.com

blood test results to JSON in C# with ByteScout Document Parser SDK

What is ByteScout Document Parser SDK? It is the robost offline data extraction platform for template based data extraction and processing. Supports high load with millions of documents as input. Templates can be quickly created and updated with no special technical knowledge required.

FOR MORE INFORMATION AND FREE TRIAL:

Download Free Trial SDK (on-premise version)

Read more about ByteScout Document Parser SDK

Explore API Documentation

Get Free Training for ByteScout Document Parser SDK

Get Free API key for Web API

visit www.ByteScout.com

Source Code Files:

BloodTestResultsToJson.csproj

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="15.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props" Condition="Exists('$
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{A73776C6-D2B2-4E37-B852-06C6454D1B5B}</ProjectGuid>
    <OutputType>Exe</OutputType>
    <RootNamespace>BloodTestResultsToJson</RootNamespace>
    <AssemblyName>BloodTestResultsToJson</AssemblyName>
    <TargetFrameworkVersion>v4.0</TargetFrameworkVersion>
    <FileAlignment>512</FileAlignment>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
```

```
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="ByteScout.DocumentParser, Version=1.0.0.100, Culture=neutral, PublicKeyToken=f7dd1bd9
      <SpecificVersion>False</SpecificVersion>
      <HintPath>c:\Program Files\ByteScout Document Parser SDK\net40\ByteScout.DocumentParser.dll</HintPath>
    </Reference>
    <Reference Include="Newtonsoft.Json, Version=12.0.0.0, Culture=neutral, PublicKeyToken=30ad4fe6b2a6aeed, pr
      <HintPath>packages\Newtonsoft.Json.12.0.3\lib\net40\Newtonsoft.Json.dll</HintPath>
    </Reference>
    <Reference Include="System" />
    <Reference Include="System.Core" />
    <Reference Include="System.Xml.Linq" />
    <Reference Include="System.Data" />
    <Reference Include="System.Xml" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="Program.cs" />
  </ItemGroup>
  <ItemGroup>
    <None Include="..\..\SampleBloodReport.pdf">
      <Link>SampleBloodReport.pdf</Link>
      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    </None>
    <None Include="..\..\_Sample Templates\SampleBloodReport.yml">
      <Link>Templates\SampleBloodReport.yml</Link>
      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    </None>
    <None Include="packages.config" />
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
</Project>
```

BloodTestResultsToJson.sln

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 15
VisualStudioVersion = 15.0.27703.2018
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "BloodTestResultsToJson", "BloodTestResultsToJson.cs
EndProject
Global
	GlobalSection(SolutionConfigurationPlatforms) = preSolution
		Debug|Any CPU = Debug|Any CPU
		Release|Any CPU = Release|Any CPU
	EndGlobalSection
	GlobalSection(ProjectConfigurationPlatforms) = postSolution
		{A73776C6-D2B2-4E37-B852-06C6454D1B5B}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
		{A73776C6-D2B2-4E37-B852-06C6454D1B5B}.Debug|Any CPU.Build.0 = Debug|Any CPU
		{A73776C6-D2B2-4E37-B852-06C6454D1B5B}.Release|Any CPU.ActiveCfg = Release|Any CPU
```

```
		{A73776C6-D2B2-4E37-B852-06C6454D1B5B}.ReleaseIAny CPU.Build.0 = ReleaseIAny CPU
	EndGlobalSection
	GlobalSection(SolutionProperties) = preSolution
		HideSolutionNode = FALSE
	EndGlobalSection
	GlobalSection(ExtensibilityGlobals) = postSolution
		SolutionGuid = {7E6DAA79-020B-421A-844A-5FE05EFC9B15}
	EndGlobalSection
EndGlobal
```

Program.cs

```
using ByteScout.DocumentParser;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Threading;

namespace BloodTestResultsToJson
{
    class Program
    {
        static void Main(string[] args)
        {
            // Sample document containing blood test report
            // Report consists of sample patient information such as patinet name, DOB, patientID
            // Report information such as report type, collection date and time
            // Test components such as complete blood count, Hemoglobin, White Blood Cell (WBC), Red Blood Cell (RBC
            string sampleDocument = @".\SampleBloodReport.pdf";

            // Sample template which primarily extracts patient name, report date and test result table.
            string sampleTemplate = @".\Templates\SampleBloodReport.yml";

            // Perform parsing with SDK
            ParseWithSDK(sampleDocument, sampleTemplate);

            // Perform parsing with PDF.co api
            // ParseWithPDFCoApi(sampleDocument, sampleTemplate);

            Console.WriteLine();
            Console.WriteLine("Press any key to continue...");
            Console.ReadLine();
        }

        /// <summary>
        /// Parse with Document Parser SDK
        /// </summary>
        static void ParseWithSDK(string sampleDocument, string sampleTemplate)
        {
            // Create DocumentParser instance
            using (DocumentParser documentParser = new DocumentParser("demo", "demo"))
            {
                // Add sample template
                documentParser.AddTemplate(sampleTemplate);

                // Parse document data in JSON format
                string jsonString = documentParser.ParseDocument(sampleDocument, OutputFormat.JSON);

                string resultFile = "result.json";

                // Write output file
                File.WriteAllText(resultFile, jsonString);
```

```
            Console.WriteLine({code}quot;Generated JSON file saved as {resultFile} file");
        }
    }

    /// <summary>
    /// Parse with PDF.Co API
    /// </summary>
    static void ParseWithPDFCoApi(string SourceFile, string sampleTemplate)
    {
        // The authentication key (API Key).
        // Get your own by registering at https://app.pdf.co/documentation/api
        const String API_KEY = "**********************************";

        // PDF document password. Leave empty for unprotected documents.
        const string Password = "";

        // Destination TXT file name
        const string DestinationFile = @".\result.json";

        // (!) Make asynchronous job
        const bool Async = true;

        // Template text. Use Document Parser SDK (https://bytescout.com/products/developer/documentparsersdk/in
        // to create templates.
        // Read template from file:
        String templateText = File.ReadAllText(sampleTemplate);

        // Create standard .NET web client instance
        WebClient webClient = new WebClient();

        // Set API Key
        webClient.Headers.Add("x-api-key", API_KEY);

        // 1. RETRIEVE THE PRESIGNED URL TO UPLOAD THE FILE.
        // * If you already have a direct file URL, skip to the step 3.

        // Prepare URL for `Get Presigned URL` API call
        string query = Uri.EscapeUriString(string.Format(
            "https://api.pdf.co/v1/file/upload/get-presigned-url?contenttype=application/octet-stream&name={0}",
            Path.GetFileName(SourceFile)));

        try
        {
            // Execute request
            string response = webClient.DownloadString(query);

            // Parse JSON response
            JObject json = JObject.Parse(response);

            if (json["error"].ToObject<bool>() == false)
            {
                // Get URL to use for the file upload
                string uploadUrl = json["presignedUrl"].ToString();
                string uploadedFileUrl = json["url"].ToString();

                // 2. UPLOAD THE FILE TO CLOUD.

                webClient.Headers.Add("content-type", "application/octet-stream");
                webClient.UploadFile(uploadUrl, "PUT", SourceFile); // You can use UploadData() instead if your file is by
                webClient.Headers.Remove("content-type");

                // 3. PARSE UPLOADED PDF DOCUMENT

                // URL for `Document Parser` API call
                query = Uri.EscapeUriString(string.Format(
                    "https://api.pdf.co/v1/pdf/documentparser?url={0}&async={1}",
                    uploadedFileUrl,
                    Async));

                Dictionary<string, string> requestBody = new Dictionary<string, string>();
                requestBody.Add("template", templateText);

                // Execute request
                response = webClient.UploadString(query, "POST", JsonConvert.SerializeObject(requestBody));

                // Parse JSON response
                json = JObject.Parse(response);
```

```
                if (json["error"].ToObject<bool>() == false)
                {
                    // Asynchronous job ID
                    string jobId = json["jobId"].ToString();
                    // Get URL of generated JSON file
                    string resultFileUrl = json["url"].ToString();

                    // Check the job status in a loop.
                    // If you don't want to pause the main thread you can rework the code
                    // to use a separate thread for the status checking and completion.
                    do
                    {
                        string status = CheckJobStatus(webClient, jobId); // Possible statuses: "working", "failed", "aborted",

                        // Display timestamp and status (for demo purposes)
                        Console.WriteLine(DateTime.Now.ToLongTimeString() + ": " + status);

                        if (status == "success")
                        {
                            // Download JSON file
                            webClient.DownloadFile(resultFileUrl, DestinationFile);

                            Console.WriteLine("Generated JSON file saved as \"{0}\" file.", DestinationFile);
                            break;
                        }
                        else if (status == "working")
                        {
                            // Pause for a few seconds
                            Thread.Sleep(3000);
                        }
                        else
                        {
                            Console.WriteLine(status);
                            break;
                        }
                    }
                    while (true);
                }
                else
                {
                    Console.WriteLine(json["message"].ToString());
                }
            }
            else
            {
                Console.WriteLine(json["message"].ToString());
            }
        }
        catch (WebException e)
        {
            Console.WriteLine(e.ToString());
        }

        webClient.Dispose();
    }

    /// <summary>
    /// Check PDF.co job status
    /// </summary>
    static string CheckJobStatus(WebClient webClient, string jobId)
    {
        string url = "https://api.pdf.co/v1/job/check?jobid=" + jobId;

        string response = webClient.DownloadString(url);
        JObject json = JObject.Parse(response);

        return Convert.ToString(json["status"]);
    }


}
}
```

packages.config

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Newtonsoft.Json" version="12.0.3" targetFramework="net40" />
</packages>
```

VIDEO

https://www.youtube.com/watch?v=MG5FfTWWSVE

ON-PREMISE OFFLINE SDK

60 Day Free Trial or Visit ByteScout Document Parser SDK Home Page
Explore ByteScout Document Parser SDK Documentation
Explore Samples
Sign Up for ByteScout Document Parser SDK Online Training

ON-DEMAND REST WEB API

Get Your API Key
Explore Web API Docs
Explore Web API Samples

visit www.ByteScout.com

visit www.PDF.co