

[www.bytescout.com](http://www.bytescout.com)

## How to parse from URL for document parser API in C# and ByteScout Cloud API Server

Learn in simple ways: How to parse from URL for document parser API in C#

The sample source codes on this page will show you how to create document parser API in C#. ByteScout Cloud API Server was designed to assist document parser API in C#. ByteScout Cloud API Server is the ready to deploy Web API Server that can be deployed in less than thirty minutes into your own in-house Windows server (no Internet connection is required to process data!) or into private cloud server. Can store data on in-house local server based storage or in Amazon AWS S3 bucket. Processing data solely on the server using built-in ByteScout powered engine, no cloud services are used to process your data!.

The SDK samples displayed below explain how to quickly make your application do document parser API in C# with the help of ByteScout Cloud API Server. For implementation of this functionality, please copy and paste the code below into your app using code editor. Then compile and run your app. Writing C# application mostly includes various stages of the software development so even if the functionality works please check it with your data and the production environment.

Our website provides free trial version of ByteScout Cloud API Server that gives source code samples to assist with your C# project.

FOR MORE INFORMATION AND FREE TRIAL:

[Download Free Trial SDK \(on-premise version\)](#)

[Read more about ByteScout Cloud API Server](#)

[Explore API Documentation](#)

[Get Free Training for ByteScout Cloud API Server](#)

[Get Free API key for Web API](#)

[visit \[www.ByteScout.com\]\(http://www.ByteScout.com\)](#)

Source Code Files:

## ByteScoutWebApiExample.csproj

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="12.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
<Import Project="$(MSBuildExtensionsPath)$(MSBuildToolsVersion)\Microsoft.Common.props" Condition="Exists('$(MSBuildExtensionsPath)$(MSBuildToolsVersion)\Microsoft.Common.props')"/>
<PropertyGroup>
<Configuration Condition=" '$(Configuration)' == "" >Debug</Configuration>
<Platform Condition=" '$(Platform)' == "" >AnyCPU</Platform>
<ProjectGuid>{1E1C2C34-017E-4605-AE2B-55EA3313BE51}</ProjectGuid>
<OutputType>Exe</OutputType>
<RootNamespace>ByteScoutWebApiExample</RootNamespace>
<AssemblyName>ByteScoutWebApiExample</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) | $(Platform)' == 'Release|AnyCPU' ">
<PlatformTarget>AnyCPU</PlatformTarget>
<DebugType>pdbonly</DebugType>
<Optimize>true</Optimize>
<OutputPath>bin\Release\</OutputPath>
<DefineConstants>TRACE</DefineConstants>
<ErrorReport>prompt</ErrorReport>
<WarningLevel>4</WarningLevel>
</PropertyGroup>
<ItemGroup>
<Reference Include="Newtonsoft.Json, Version=10.0.0.0, Culture=neutral, PublicKeyToken=30ad4fe6b2a6aeed, processorArchitecture=MSIL">
<HintPath>packages\Newtonsoft.Json.10.0.3\lib\net40\Newtonsoft.Json.dll</HintPath>
<Private>True</Private>
</Reference>
<Reference Include="System" />
<Reference Include="System.Core" />
<Reference Include="System.Xml.Linq" />
<Reference Include="System.Data" />
<Reference Include="System.Xml" />
</ItemGroup>
<ItemGroup>
<Compile Include="Program.cs" />
</ItemGroup>
<ItemGroup>
<None Include="MultiPageTable-template1.yml">
<CopyToOutputDirectory>Always</CopyToOutputDirectory>
</None>
<None Include="packages.config" />
</ItemGroup>
<Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
<!-- To modify your build process, add your task inside one of the targets below and uncomment it.
     Other similar extension points exist, see Microsoft.Common.targets.
<Target Name="BeforeBuild">
</Target>
<Target Name="AfterBuild">
</Target>
-->
</Project>
```

## ByteScoutWebApiExample.sln

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 2013
VisualStudioVersion = 12.0.40629.0
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "ByteScoutWebApiExample", "ByteScoutWebApiExample"
EndProject
Global
    GlobalSection(SolutionConfigurationPlatforms) = preSolution
        Debug|Any CPU = Debug|Any CPU
        Release|Any CPU = Release|Any CPU
    EndGlobalSection
    GlobalSection(ProjectConfigurationPlatforms) = postSolution
        {1E1C2C34-017E-4605-AE2B-55EA3313BE51}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
        {1E1C2C34-017E-4605-AE2B-55EA3313BE51}.Debug|Any CPU.Build.0 = Debug|Any CPU
        {1E1C2C34-017E-4605-AE2B-55EA3313BE51}.Release|Any CPU.ActiveCfg = Release|Any CPU
        {1E1C2C34-017E-4605-AE2B-55EA3313BE51}.Release|Any CPU.Build.0 = Release|Any CPU
    EndGlobalSection
    GlobalSection(SolutionProperties) = preSolution
        HideSolutionNode = FALSE
    EndGlobalSection
EndGlobal
```

## MultiPageTable-template1.yml

```
---
# Template that demonstrates parsing of multi-page table using only
# regular expressions for the table start, end, and rows.
# If regular expression cannot be written for every table row (for example,
# if the table contains empty cells), try the second method demonstrated
# in `MultiPageTable-template2.yml` template.
templateVersion: 2
templatePriority: 0
sourceId: Multipage Table Test
detectionRules:
    keywords:
        - Sample document with multi-page table
fields:
    total:
        expression: TOTAL {{DECIMAL}}
tables:
    - name: table1
        start:
            # regular expression to find the table start in document
            expression: Item\s+Description\s+Price\s+Qty\s+Extended Price
        end:
            # regular expression to find the table end in document
            expression: TOTAL\s+\d+\.\d\d
        row:
            # regular expression to find table rows
            expression: ^\s*(?<itemNo>\d+)\s+(?<description>.+)\s+(?<price>\d+\.\d\d)\s+(?<qty>\d+)\s+(?<extPrice>\d+\.\d\d)
        columns:
            - name: itemNo
                type: integer
            - name: description
                type: string
            - name: price
                type: decimal
            - name: qty
                type: integer
            - name: extPrice
```

**type:** decimal  
**multipage:** true

## Program.cs

```
using System;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Text;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;

// Please NOTE: In this sample we're assuming Cloud API Server is hosted at "https://localhost".
// If it's not then please replace this with your hosting url.

namespace ByteScoutWebApiExample
{
    class Program
    {
        // Source PDF file
        const string SourceFileUrl = "https://bytescout-com.s3.amazonaws.com/files/demo-files/cloud-api/document-parser/multi-page-table.pdf";

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

        static void Main(string[] args)
        {
            // Template text. Use Document Parser SDK (https://bytescout.com/products/developer/documentparsersdk/instruction) to create templates.
            // Read template from file:
            String templateText = File.ReadAllText(@"..\MultiPageTable-template1.yml");

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

            try
            {
                // PARSE UPLOADED PDF DOCUMENT

                // URL for `Document Parser` API call
                string query = Uri.EscapeUriString(string.Format(
                    "https://localhost/pdf/documentparser?url={0}",
                    SourceFileUrl));

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

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

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

                if (json["error"].ToObject<bool>() == false)
                {
                    // Get URL of generated JSON file
                    string resultImageUrl = json["url"].ToString();

                    // Download JSON file
                    webClient.DownloadFile(resultImageUrl, DestinationFile);

                    Console.WriteLine("Generated JSON file saved as \"{0}\" file.", DestinationFile);
                }
                else
                {

```

```
        Console.WriteLine(json["message"].ToString());
    }
}
catch (WebException e)
{
    Console.WriteLine(e.ToString());
}

webClient.Dispose();

Console.WriteLine();
Console.WriteLine("Press any key...");
Console.ReadKey();
}
}
```

## packages.config

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Newtonsoft.Json" version="10.0.3" targetFramework="net40" />
</packages>
```

---

## VIDEO

<https://www.youtube.com/watch?v=NEwNs2b9YN8>

## ON-PREMISE OFFLINE SDK

[60 Day Free Trial](#) or [Visit ByteScout Cloud API Server Home Page](#)  
[Explore ByteScout Cloud API Server Documentation](#)  
[Explore Samples](#)  
[Sign Up for ByteScout Cloud API Server Online Training](#)

## ON-DEMAND REST WEB API

[Get Your API Key](#)  
[Explore Web API Docs](#)  
[Explore Web API Samples](#)

[visit www.ByteScout.com](http://www.ByteScout.com)

[visit www.PDF.co](http://www.PDF.co)

[www.bytescout.com](http://www.bytescout.com)