www.bytescout.com

How to parse and generate hl7 output in C# and ByteScout Document Parser SDK

Learn to code in C# to parse and generate hl7 output with this step-by-step tutorial

The sample source codes on this page shows how to parse and generate hl7 output in C#. ByteScout Document Parser SDK can parse and generate hl7 output. It can be applied from C#. ByteScout Document Parser SDK is the template based data extraction engine can process thousands and millions of documents per day based on templates. Can work online and offline for privacy focused tasks. Templates can be supported and updated without any programming or technical knowledge required. Output is generated in JSON, CSV, XML and custom format if required.

Want to save time? You will save a lot of time on writing and testing code as you may just take the C# code from ByteScout Document Parser SDK for parse and generate hl7 output below and use it in your application. Follow the instructions from scratch to work and copy the C# code. This basic programming language sample code for C# will do the whole work for you to parse and generate hl7 output.

You can download free trial version of ByteScout Document Parser SDK from our website with this and other source code samples for C#.

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:

HL7CreationFromJson.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>HL7CreationFromJson</RootNamespace>
    <AssemblyName>HL7CreationFromJson</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=f7dd1bd9c
      <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.12.0.2\lib\net40\Newtonsoft.Json.dll</HintPath>
      <SpecificVersion>False</SpecificVersion>
    </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="HI7Helper.cs" />
    <Compile Include="JsonHL7Fields.cs" />
    <Compile Include="JsonParserHelper.cs" />
    <Compile Include="Program.cs" />
    <Compile Include="Src\Component.cs" />
    <Compile Include="Src\Encoding.cs" />
    <Compile Include="Src\Field.cs" />
    <Compile Include="Src\HL7Exception.cs" />
    <Compile Include="Src\Message.cs" />
    <Compile Include="Src\MessageElement.cs" />
    <Compile Include="Src\MessageHelper.cs" />
    <Compile Include="Src\Segment.cs" />
    <Compile Include="Src\SubComponent.cs" />
  </ItemGroup>
  <ItemGroup>
    <None Include="InputData\TestReportFormat.yml">
      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    </None>
    <None Include="InputData\Test_Report_Format.pdf">
      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    </None>
```

```
    <None Include="packages.config" />
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
</Project>
```

HL7CreationFromJson.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}") = "HL7CreationFromJson", "HL7CreationFromJson.csproj
EndProject
Global
	GlobalSection(SolutionConfigurationPlatforms) = preSolution
		DebugIAny CPU = DebugIAny CPU
		ReleaseIAny CPU = ReleaseIAny CPU
	EndGlobalSection
	GlobalSection(ProjectConfigurationPlatforms) = postSolution
		{A73776C6-D2B2-4E37-B852-06C6454D1B5B}.DebugIAny CPU.ActiveCfg = DebugIAny CPU
		{A73776C6-D2B2-4E37-B852-06C6454D1B5B}.DebugIAny CPU.Build.0 = DebugIAny CPU
		{A73776C6-D2B2-4E37-B852-06C6454D1B5B}.ReleaseIAny CPU.ActiveCfg = ReleaseIAny 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
```

HL7Helper.cs

```
using HL7.Dotnetcore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HL7CreationFromJson
{
    class Hl7Helper
    {
        /// <summary>
        /// Get HL7 format representation from input model
        /// </summary>
        public static string GetHL7Format(JsonHL7Fields inpModel)
        {
            // https://github.com/Efferent-Health/HL7-dotnetcore
            // http://www.j4jayant.com/2013/05/hl7-parsing-in-csharp.html

            Message oHl7Message = new Message();

            // Add MSH Segment
```

```
            Segment mshSegment = new Segment("MSH", new HL7Encoding());
            mshSegment.AddNewField("SendingApp", 3);
            mshSegment.AddNewField(inpModel.LabName ?? "", 4);
            mshSegment.AddNewField(DateTime.Now.ToString("yyyymmddhhMMss"), 7);
            mshSegment.AddNewField("ORM", 9); // Message type
            mshSegment.AddNewField("2.3", 12); // Message version
            oHl7Message.AddNewSegment(mshSegment);

            // Add PID Segment
            Segment pidSegment = new Segment("PID", new HL7Encoding());
            pidSegment.AddNewField("1", 1);
            pidSegment.AddNewField(inpModel.PatientChartNo ?? "", 2); // Patient ID
            pidSegment.AddNewField(inpModel.PatientChartNo ?? "", 4); // Alternate Patient ID
            pidSegment.AddNewField({code}quot;{inpModel.PatientLastName ?? ""}^{inpModel.PatientFirstName ?? ""}", 
            pidSegment.AddNewField(inpModel.PatientDOB ?? "", 7); // Patient DOB
            pidSegment.AddNewField(inpModel.PatientGender ?? "", 8); // Patient Gender
            pidSegment.AddNewField(inpModel.PatientAddress ?? "", 11); // Patient Address
            pidSegment.AddNewField(inpModel.PatientPhoneHome ?? "", 13); // Patient Home Phone number
            pidSegment.AddNewField(inpModel.PatientSSN ?? "", 19); // Patient SSN Number
            oHl7Message.AddNewSegment(pidSegment);

            // Add PV1 Segment
            Segment pv1Segment = new Segment("PV1", new HL7Encoding());
            pv1Segment.AddNewField({code}quot;{inpModel.PhysicianNpi ?? ""}^{inpModel.PhysicianName}", 7); // Physi
            oHl7Message.AddNewSegment(pv1Segment);

            // Add IN1 Segment
            Segment in1Segment = new Segment("IN1", new HL7Encoding());
            in1Segment.AddNewField("1", 1);
            in1Segment.AddNewField(inpModel.InsuranceName ?? "", 4); // Insurance Name
            in1Segment.AddNewField(inpModel.InsuranceGroup ?? "", 8); // Insurance Group Name
            in1Segment.AddNewField(inpModel.InsuredName ?? "", 16); // Insured Name
            in1Segment.AddNewField(inpModel.RelationToPatient ?? "", 17); // Insured Relatino
            in1Segment.AddNewField(inpModel.InsuredDob ?? "", 18); // Insured Date of Birth
            in1Segment.AddNewField(inpModel.InsurancePolicy ?? "", 36); // Insurance Policy Number
            oHl7Message.AddNewSegment(in1Segment);

            // Add ORC Segment
            Segment orcSegment = new Segment("ORC", new HL7Encoding());
            orcSegment.AddNewField("NW", 1); // New Order
            orcSegment.AddNewField(inpModel.CollectionDateTime ?? "", 9); // Date/Time of Transaction
            orcSegment.AddNewField({code}quot;{inpModel.PhysicianNpi ?? ""}^{inpModel.PhysicianName ?? ""}", 12); //
            oHl7Message.AddNewSegment(orcSegment);

            // Add OBR Segment
            Segment obrSegment = new Segment("OBR", new HL7Encoding());
            obrSegment.AddNewField(inpModel.CollectionDateTime ?? "", 7); // Date/Time of Transaction
            obrSegment.AddNewField({code}quot;{inpModel.PhysicianNpi ?? ""}^{inpModel.PhysicianName ?? ""}", 16); //
            oHl7Message.AddNewSegment(obrSegment);

            // Add Diagnosis
            for (int i = 0; i < inpModel.Icd10Codes.Count; i++)
            {
                Segment dg1Segment = new Segment("DG1", new HL7Encoding());
                dg1Segment.AddNewField((i + 1).ToString(), 1);
                dg1Segment.AddNewField("I10", 2); // Icd Type
                dg1Segment.AddNewField(inpModel.Icd10Codes[i], 3); // Icd Code
                oHl7Message.AddNewSegment(dg1Segment);
            }

            // Add OBX
            for (int i = 0; i < inpModel.QuestionAnswer.Count; i++)
            {
                Segment obxSegment = new Segment("OBX", new HL7Encoding());
                obxSegment.AddNewField((i + 1).ToString(), 1);
                obxSegment.AddNewField("ST", 2); // Value Type
                obxSegment.AddNewField(inpModel.QuestionAnswer[i].Key, 3); // Question
                obxSegment.AddNewField(inpModel.QuestionAnswer[i].Value, 5); // Answer
                oHl7Message.AddNewSegment(obxSegment);
            }

            string oRetMessage = oHl7Message.SerializeMessage(false);

            return oRetMessage;
        }
    }
}
```

JsonHL7Fields.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HL7CreationFromJson
{
    public class JsonHL7Fields
    {
        public JsonHL7Fields()
        {
            Icd10Codes = new List<string>();
            QuestionAnswer = new List<KeyValuePair<string, string>>();
        }

        public string LabName { get; set; }

        public string PatientLastName { get; set; }
        public string PatientFirstName { get; set; }
        public string PatientSSN { get; set; }
        public string PatientDOB { get; set; }
        public string PatientPhoneHome { get; set; }
        public string PatientPhoneWork { get; set; }
        public string PatientChartNo { get; set; }
        public string PatientGender { get; set; }
        public string PatientAddress { get; set; }

        public string PhysicianName { get; set; }
        public string PhysicianAccountNo { get; set; }
        public string PhysicianNpi { get; set; }

        public string InsuranceName { get; set; }
        public string InsurancePolicy { get; set; }
        public string InsuranceGroup { get; set; }
        public string InsuredName { get; set; }
        public string InsuredSSN { get; set; }
        public string InsuredDob { get; set; }
        public string RelationToPatient { get; set; }

        public string CollectionDateTime { get; set; }

        public List<string> Icd10Codes { get; set; }
        public List<KeyValuePair<string, string>> QuestionAnswer { get; set; }

    }
}
```

JsonParserHelper.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using Newtonsoft.Json.Linq;

namespace HL7CreationFromJson
{
    class JsonParserHelper
    {
        /// <summary>
        /// Parse Json Fileds in class format
        /// </summary>
        public static JsonHL7Fields ParseJsonHL7Fields(string jsonData)
        {
            // Get Object data from input file
            JObject jsonObj = JObject.Parse(jsonData);

            var oRet = new JsonHL7Fields();

            oRet.LabName = Convert.ToString(jsonObj["fields"]["labName"]["value"]);

            oRet.PatientLastName = Convert.ToString(jsonObj["fields"]["patientLastName"]["value"]);
            oRet.PatientFirstName = Convert.ToString(jsonObj["fields"]["patientFirstName"]["value"]);
            oRet.PatientSSN = Convert.ToString(jsonObj["fields"]["patientSSN"]["value"]);
            oRet.PatientDOB = Convert.ToString(jsonObj["fields"]["patientDOB"]["value"]);
            oRet.PatientPhoneHome = Convert.ToString(jsonObj["fields"]["patientHomePhone"]["value"]);
            oRet.PatientPhoneWork = Convert.ToString(jsonObj["fields"]["patientWorkPhone"]["value"]);
            oRet.PatientAddress = Convert.ToString(jsonObj["fields"]["patientAddress"]["value"]);
            oRet.PatientChartNo = Convert.ToString(jsonObj["fields"]["patientChartNo"]["value"]);

            string patGenderMaleSelectedVal = Convert.ToString(jsonObj["fields"]["patientGenderMale"]["value"]);
            string patGenderFemaleSelectedVal = Convert.ToString(jsonObj["fields"]["patientGenderFemale"]["value"]);

            if (!string.IsNullOrEmpty(patGenderMaleSelectedVal))
            {
                oRet.PatientGender = "M";
            }
            else if (!string.IsNullOrEmpty(patGenderFemaleSelectedVal))
            {
                oRet.PatientGender = "F";
            }

            oRet.PhysicianName = Convert.ToString(jsonObj["fields"]["physicianName"]["value"]);
            oRet.PhysicianAccountNo = Convert.ToString(jsonObj["fields"]["physicianAccountName"]["value"]);
            oRet.PhysicianNpi = Convert.ToString(jsonObj["fields"]["physicianNPI"]["value"]);

            oRet.InsuranceName = Convert.ToString(jsonObj["fields"]["insuranceName"]["value"]);
            oRet.InsurancePolicy = Convert.ToString(jsonObj["fields"]["insurancePolicy"]["value"]);
            oRet.InsuranceGroup = Convert.ToString(jsonObj["fields"]["insuranceGroup"]["value"]);
            oRet.InsuredName = Convert.ToString(jsonObj["fields"]["insuredName"]["value"]);
            oRet.InsuredSSN = Convert.ToString(jsonObj["fields"]["insuredSSN"]["value"]);
            oRet.InsuredDob = Convert.ToString(jsonObj["fields"]["insuredDOB"]["value"]);

            string relToPatIsSelf = Convert.ToString(jsonObj["fields"]["relationToPatientIsSelf"]["value"]);
            string relToPatIsSpouse = Convert.ToString(jsonObj["fields"]["relationToPatientIsSpouse"]["value"]);
            string relToPatIsDependent = Convert.ToString(jsonObj["fields"]["relationToPatientIsDependent"]["value"]);

            if (!string.IsNullOrEmpty(relToPatIsSelf))
            {
                oRet.RelationToPatient = "Self";
            }
            else if (!string.IsNullOrEmpty(relToPatIsSpouse))
            {
                oRet.RelationToPatient = "Spouse";
            }
            else if (!string.IsNullOrEmpty(relToPatIsDependent))
            {
                oRet.RelationToPatient = "Dependent";
            }

            // Add Collection Date/Time
            string colDate = Convert.ToString(jsonObj["fields"]["collectionDate"]["value"]);
            string colTime = Convert.ToString(jsonObj["fields"]["collectionTime"]["value"]);
            string colTimeIsAm = Convert.ToString(jsonObj["fields"]["collectionTimeIsAM"]["value"]);
            string colTimeIsPm = Convert.ToString(jsonObj["fields"]["collectionTimeIsPM"]["value"]);

            string colTimeAmPm = "";
            if (!string.IsNullOrEmpty(colTimeIsAm))
            {
                colTimeAmPm = "AM";
            }
```

```
            else if (!string.IsNullOrEmpty(colTimeIsPm))
            {
                colTimeAmPm = "PM";
            }

            oRet.CollectionDateTime = {code}quot;{colDate} {colTime} {colTimeAmPm}";


            // Add ICD Codes
            string IcdCodes = Convert.ToString(jsonObj["fields"]["icD10DxCodes"]["value"]);
            if (!string.IsNullOrEmpty(IcdCodes))
            {
                var arrIcdCodes = IcdCodes.Split(',');

                foreach (var itmIcd in arrIcdCodes)
                {
                    oRet.Icd10Codes.Add(itmIcd.Trim());
                }
            }

            // Add Question/Answers
            string Ques_ClinicalHistoryIsRoutinePap = string.IsNullOrEmpty(Convert.ToString(jsonObj["fields"]["clinicalHis
            string Ques_ClinicalHistoryIsAbnormalBleeding = string.IsNullOrEmpty(Convert.ToString(jsonObj["fields"]["clin

            oRet.QuestionAnswer.Add(new KeyValuePair<string, string>("Is Routine PAP?", Ques_ClinicalHistoryIsRoutir
            oRet.QuestionAnswer.Add(new KeyValuePair<string, string>("Is Abnormal Bleeding?", Ques_ClinicalHistoryIs

            return oRet;
        }

    }
}
```

Program.cs

```
using System;
using System.Diagnostics;
using ByteScout.DocumentParser;

// This example demonstrates document data parsing to JSON, YAML and XML formats.

namespace HL7CreationFromJson
{
    class Program
    {
        static void Main(string[] args)
        {
            // Step 1: Generate Parse PDF File With Template and Generate Json
            string inputPDF = "InputData/Test_Report_Format.pdf";
            string template = "InputData/TestReportFormat.yml";

            // Create Document Parser Instance
            DocumentParser docParser = new DocumentParser("demo", "demo");

            // Add Template
            docParser.AddTemplate(template);

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

            // Step 2: Parse Json fileds in class format
            var oInpModel = JsonParserHelper.ParseJsonHL7Fields(jsonString);

            // Step 3: Get Data in HL7 Format
            var oHL7Format = Hl7Helper.GetHL7Format(oInpModel);
```

```
            // Store HL7 File and open with default associated program
            var outputFile = "outputHl7.txt";
            System.IO.File.WriteAllText(outputFile, oHL7Format);
            Process.Start(outputFile);
        }
    }
}
```

packages.config

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Newtonsoft.Json" version="12.0.2" 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