How to convert PDF to HTML from URL asynchronously for PDF to HTML API in PowerShell using ByteScout Cloud API Server

Follow this simple tutorial to learn convert PDF to HTML from URL asynchronously to have PDF to HTML API in PowerShell

The sample source codes on this page will show you how to create PDF to HTML API in PowerShell. ByteScout Cloud API Server was designed to assist PDF to HTML API in PowerShell. 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 connnection 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!.

If you want to speed up the application's code writing then PowerShell code samples for PowerShell developers help to implement using ByteScout Cloud API Server. Open your PowerShell project and simply copy & paste the code and then run your app! Check PowerShell sample code examples to see if they respond to your needs and requirements for the project.

Trial version of ByteScout is available for free download from our website. This and other source code samples for PowerShell and other programming languages are available.

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

Source Code Files:

ConvertPdfToHtmlFromUrlAsynchronously.ps1

```
# Cloud API asynchronous "PDF To HTML" job example.
# Allows to avoid timeout errors when processing huge or scanned PDF documents.

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

# Direct URL of source PDF file.
$SourceFileUrl = "https://bytescout-com.s3.amazonaws.com/files/demo-files/cloud-api/pd
# Comma-separated list of page indices (or ranges) to process. Leave empty for all page
$Pages = ""
# PDF document password. Leave empty for unprotected documents.
$Password = ""
# Destination HTML file name
$DestinationFile = ".\result.html"
# Set to $true to get simplified HTML without CSS. Default is the rich HTML keeping the
$PlainHtml = $false
# Set to $true if your document has the column layout like a newspaper.
$ColumnLayout = $false
# (!) Make asynchronous job
$Async = $true


# Prepare URL for `PDF To HTML` API call
$query = "https://localhost/pdf/convert/to/html?name={0}&password={1}&pages={2}&simple=
    $(Split-Path $DestinationFile -Leaf), $Password, $Pages, $PlainHtml, $ColumnLayout
$query = [System.Uri]::EscapeUriString($query)

try {
    # Execute request
    $jsonResponse = Invoke-RestMethod -Method Get  -Uri $query

    if ($jsonResponse.error -eq $false) {
        # Asynchronous job ID
        $jobId = $jsonResponse.jobId
        # URL of generated HTML file that will available after the job completion
        $resultFileUrl = $jsonResponse.url

        # Check the job status in a loop.
        do {
            $statusCheckUrl = "https://localhost/job/check?jobid=" + $jobId
            $jsonStatus = Invoke-RestMethod -Method Get  -Uri $statusCheckUrl

            # Display timestamp and status (for demo purposes)
            Write-Host "$(Get-date): $($jsonStatus.status)"

            if ($jsonStatus.status -eq "success") {
                # Download HTML file
                Invoke-WebRequest  -OutFile $DestinationFile -Uri $resultFileUrl
                Write-Host "Generated HTML file saved as `"$($DestinationFile)`" file."
                break
            }
            elseif ($jsonStatus.status -eq "working") {
                # Pause for a few seconds
                Start-Sleep -Seconds 3
            }
            else {
                Write-Host $jsonStatus.status
                break
            }
        }
        while ($true)
    }
    else {
        # Display service reported error
        Write-Host $jsonResponse.message
    }
}
catch {
    # Display request error
    Write-Host $_.Exception
}
```

run.bat

```
@echo off

powershell -NoProfile -ExecutionPolicy Bypass -Command "& .\ConvertPdfToHtmlFromUrlAsy
echo Script finished with errorlevel=%errorlevel%

pause
```

VIDEO

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

ON-PREMISE OFFLINE SDK