How to PDF text search API in PowerShell and PDF.co Web API

This code in PowerShell shows how to PDF text search API with this how to tutorial

On this page you will learn from code samples for programming in PowerShell.Writing of the code to PDF text search API in PowerShell can be done by developers of any level using PDF.co Web API. PDF.co Web API: the Web API with a set of tools for documents manipulation, data conversion, data extraction, splitting and merging of documents. Includes image recognition, built-in OCR, barcode generation and barcode decoders to decode bar codes from scans, pictures and pdf. It can PDF text search API in PowerShell.

PowerShell code samples for PowerShell developers help to speed up coding of your application when using PDF.co Web API. This PowerShell sample code is all you need for your app. Just copy and paste the code, add references (if needs to) and you are all set! Code testing will allow the function to be tested and work properly with your data.

ByteScout free trial version is available for download from our website. It includes all these programming tutorials along with source code samples.

PowerShell - PDFTextSearchFromUrlAsynchronously.ps1

```
# The authentication key (API Key).
# Get your own by registering at https://app.pdf.co/documentation/api
$API_KEY = "**********************************"

# Direct URL of PDF file to get information
$SourceFileURL = "https://bytescout-com.s3.amazonaws.com/files/demo-files/cloud-
api/pdf-to-text/sample.pdf"

# Comma-separated list of page indices (or ranges) to process. Leave empty for all
pages. Example: '0,2-5,7-'.
$Pages = ""

# PDF document password. Leave empty for unprotected documents.
$Password = ""

# Search string.
$SearchString = '\d{1,}\.\d\d' #Regular expression to find numbers like '100.00'

# Enable regular expressions (Regex)
$RegexSearch = 'True'

# (!) Make asynchronous job
$Async = $true

# Prepare URL for PDF text search API call.
```

```
# See documentation: https://app.pdf.co/documentation/api/1.0/pdf/find.html
$query = "https://api.pdf.co/v1/pdf/find?
password=$($Password)&pages=$($Pages)&url=$($SourceFileURL)&searchString=$($SearchStri

$query = [System.Uri]::EscapeUriString($query)

try {
    # Execute request
    $jsonResponse = Invoke-RestMethod -Method Get -Headers @{ "x-api-key" = $API_KEY
} -Uri $query

    if ($jsonResponse.error -eq $false) {
        # Asynchronous job ID
        $jobId = $jsonResponse.jobId

        # URL of generated JSON file with search result that will available after the
job completion
        $resultFileUrl = $jsonResponse.url

        # 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 {
            $statusCheckUrl = "https://api.pdf.co/v1/job/check?jobid=" + $jobId
            $jsonStatus = Invoke-RestMethod -Method Get -Headers @{ "x-api-key" =
$API_KEY } -Uri $statusCheckUrl

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

            if ($jsonStatus.status -eq "success") {
                # Get JSON for search result
                $jsonSearchResult = Invoke-RestMethod -Method Get -Headers @{ "x-api-
key" = $API_KEY } -Uri $resultFileUrl

                # Display found result in console
                foreach ($item in $jsonSearchResult)
                {
                    Write-Host "Found text $($item.text) at coordinates
$($item.left), $($item.top)"
                }
                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
}
```

PowerShell - run.bat

```
@echo off

powershell -NoProfile -ExecutionPolicy Bypass -Command "&
.\PDFTextSearchFromUrlAsynchronously.ps1"
echo Script finished with errorlevel=%errorlevel%

pause
```
