# Automate document manipulation with Aspose.Words Product Family

## Introduction

Aspose.Words provides the most complete and efficient document-processing API with a particular focus on Microsoft Word and OpenOffice documents. We support both on-premise SDKs for [.NET](https://products.aspose.com/words/net), [Java](https://products.aspose.com/words/java) & [C++](https://products.aspose.com/words/cpp) to manipulate Word document on Windows, Linux, iOS & Android operating systems, and Cloud [API](https://apireference.aspose.cloud/words/) with a set of [libraries](https://products.aspose.cloud/words/family) for different programming languages, designed to perform a wide variety of operations with high scalability and speed on the Web.

**Note**: There are several Aspose.Words products: On Premise APIs, Cloud APIs and [Docker](https://hub.docker.com/r/aspose/words-cloud) container.

## Feature Set

You can easily integrate Word conversion, generation, and editing features into applications running on any modern OS. Aspose.Words allows you to work with a complete set of Word document components, such as main text, headers, footers, tables, sections, comments, drawing objects, fonts, hyperlinks, math objects, watermarks, and many others. Our SDKs assists in reading, editing, merging, splitting documents as well as converting Word documents to other supported file formats.

## Supported File Formats

**Read and write** formats: DOCX, DOC, RTF, DOTX, DOT, ODT, OTT, XML, HTML, MHTML, TXT, PDF.

**Write-only** formats are: XPS, OpenXPS, PS, JPEG, PNG, BMP, SVG, TIFF, EMF, PCL, EPUB.

# Convert PDF to DOCX in Python

from asposewordscloud import WordsApi  
from asposewordscloud.models.requests import ConvertDocumentRequest  
  
app\_sid = '####-####-####-####-####'  
app\_key = '##################'  
words\_api = WordsApi(app\_sid, app\_key)  
with open('sample.pdf', 'rb') as f:  
 request = ConvertDocumentRequest(f, format='docx')  
 result = words\_api.convert\_document(request)  
print('Output filename: {}'.format(result))

# How Aspose.Words works with fonts on different operating systems

## What could possibly go wrong?

When working with documents, one way or another you have to use different fonts, styles and sizes.

Documents can contain any number of fonts and these are not necessarily only fonts that are installed in the operating system. Fonts can be completely different, taken from other operating systems, purchased or created by the users themselves.

Sometimes information about the used fonts is embedded in the document, eliminating display problems during document transfer.

How does Aspose.Words find the right font?

In case the required font cannot be found, how to find a good replacement?

**Note**: At the moment Aspose.Words supports [OpenType](https://en.wikipedia.org/wiki/OpenType) and [TrueType](https://en.wikipedia.org/wiki/TrueType) font formats.

# Fonts on Windows systems

In most cases, while working in Windows environment, users don’t face significant problems with missed fonts or incorrect layout.

Aspose.Words goes through a document and when it encounters a font’s link it successfully fetches font’s data from the system folder.

On Windows Aspose.Words takes all available fonts from System.getenv("WINDIR")\\fonts folder first, then retrieves the additional fonts by their paths stored in the registry

HKEY\_LOCAL\_MACHINE\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Fonts

If a document contains embedded fonts, Aspose.Word can read their data from the document and use it to build up the document’s layout.

But documents could contain links to fonts that are not in the system folders, then several scenarios come to play:

Users can set up new font sources via *FontSettings* class;

Aspose.Words can try and replace the missed font with the similar one.

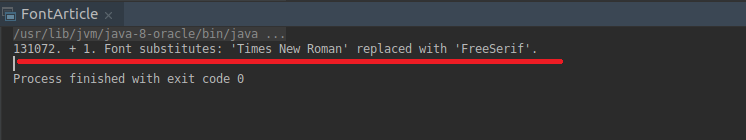
## How to recognize that the font was replaced

Sometimes it may be unclear why the layout has changed or some font does not look as expected. *IWarningCallback* interface comes to rescue. You can pass your own implementation of this interface via method loadOptions.setWarningCallback().

There are eleven types of warnings. We are interested in *FONT\_SUBSTITUTION*.

Here is the example how you can configure your own warning callback

|  |
| --- |
| **Java** |
| private static IWarningCallback createWarningCallback() {  return new IWarningCallback() {  @Override  public void warning(WarningInfo info) {  if (info.getWarningType() == WarningType.*FONT\_SUBSTITUTION*)  log.info(info.getWarningType() + ". + " + info.getSource() + ". " + info.getDescription());  }  };  } |
| **NET** |
| private class CustomWarningCallback : IWarningCallback  {  public void Warning(WarningInfo info)  {  if (info.WarningType == WarningType.FontSubstitution)  Console.WriteLine(info.WarningType + ". + " + info.Source + ". " + info.Description);  }  } |

As you can see in the output the “Times New Roman” font was replaced with “FreeSerif”

## Add a custom font source

If the document processed contains links to fonts that are not in the system, or you don't want to (or you can’t because of lack of permissions) add them to the system folder, then the best solution would be to add a folder with your own fonts along with the system ones using the setFontsSources method.

|  |
| --- |
| **Java** |
| FontSettings fontSettings = new FontSettings();  fontSettings.setFontsSources(  new FontSourceBase[]{  new FolderFontSource("/home/user/MyFonts", true, 1),  new SystemFontSource(0),  }  ); |
| **NET** |
| FontSettings fontSettings = new FontSettings();  fontSettings.SetFontsSources(  new FontSourceBase[]{  new FolderFontSource("/home/user/MyFonts", true, 1),  new SystemFontSource(0),  }  ); |

**Note priorities**: If there are fonts with the same familyName and style in different fontSources, then Aspose.Words will select a font from a source with a higher priority.

If you don’t want to use the system fonts at all, Aspose.Words allows you to ignore them and use your own fonts only.

|  |
| --- |
| **Java** |
| FontSettings fontSettings = new FontSettings();  fontSettings.setFontsFolder("/home/user/MyFonts", true); |