



DEVELOPER GUIDE

Foxit[®] WebPDF SDK

Microsoft[®] Partner
Gold Independent Software Vendor (ISV)

Table of Contents

1	Introduction to Foxit WebPDF SDK	1
1.1	Why Foxit WebPDF SDK is your choice	1
1.2	Audience and Scope	1
1.3	Your Web Application	2
1.4	Features.....	2
1.5	Evaluation.....	2
1.6	License.....	2
2	Getting Started	4
2.1	System Requirements	4
2.2	What's in the package.....	4
2.2.1	For Windows.....	4
2.2.2	For Linux	5
2.3	Installation.....	5
2.3.1	For Windows.....	5
2.3.1.1	Installation	5
2.3.1.2	Uninstallation.....	9
2.3.1.3	Run WebPDF Reader.....	11
2.3.2	For Linux	12
2.3.2.1	Installation	12
2.3.2.2	Uninstallation.....	12
2.3.2.3	Run WebPDF Reader.....	12
2.3.3	Docker container mode deployment	13
2.3.3.1	Prepare a CentOS or Ubuntu server (physical or virtual machine).....	13
2.3.3.2	Install docker.....	13

2.3.3.3	Obtain the docker image	13
2.3.3.4	Run the docker image	13
2.3.3.5	Run WebPDF Reader	13
2.3.3.6	Supported Docker versions.....	13
2.3.4	Administrator management	14
2.3.4.1	Apply a license	14
2.3.4.2	Set a watermark.....	15
2.3.4.3	Database Setting	16
2.3.4.4	DPI settings	17
2.3.4.5	Manage the Cache	17
3	Database.....	18
4	Working with SPI	18
4.1	Connect PDF files with WebPDF SDK (Document Adapter Plug-in)	18
4.1.1	Create a Document Adapter Plug-in project	19
4.1.2	Run the Document Plugin.....	21
4.1.2.1	Get a PDF file.....	21
4.1.2.2	Export a PDF file.....	22
4.2	Integrate Digital Signature (Signature Plug-in)	22
4.2.1	Create a Signature Plug-in project.....	22
4.2.2	Run the Signature Plug-in	22
4.3	Integrate Document Management System User Accounts (System User Plug-in)	23
4.3.1	Create a System User Plug-in	23
4.3.2	Run the System User Plug-in	23
4.3.2.1	Ink signature template list	24
4.4	Import/export form data (XML) to/from a remote server (Form Plug-in).....	24

4.4.1	Create a Form Plug-in project.....	25
4.4.2	Run the Form Plug-in	25
5	Working with API.....	25
5.1	WebPDF Reader on PC	26
5.1.1	Open a PDF file on your own system.....	26
5.1.2	Annotation.....	29
5.1.3	Save Annotation	29
5.1.4	Rotation	29
5.1.5	Download	30
5.1.6	Print	30
5.1.7	Signature.....	30
5.1.7.1	Set the signature properties	30
5.1.7.2	Add a signature	31
5.1.7.3	Check the signature properties.....	31
5.1.8	Ink Signature.....	32
5.1.8.1	Sign with handwriting without a certificate	32
5.1.8.2	Sign with handwriting with a certificate	32
5.1.9	Stamp.....	34
5.1.10	Form	34
5.1.11	Bookmark.....	35
5.1.12	Thumbnail.....	36
5.1.13	Comment list	36
5.1.14	Search	38
5.1.15	Watermark.....	38
5.2	WebPDF Reader on Mobile.....	39

5.2.1	Open a PDF file on your own system.....	39
5.2.2	Annotation.....	40
5.2.3	Save Annotation	40
5.2.4	Rotation	40
5.2.5	Ink Signature.....	41
5.2.6	Web PDF Form.....	41
5.2.7	Bookmark.....	41
5.2.8	Thumbnail.....	42
5.2.9	Comment list	42
5.2.10	Search	42
5.2.11	Watermark.....	43
6	FAQ.....	43
	Support	44

1 Introduction to Foxit WebPDF SDK

By using Foxit WebPDF SDK, developers can deploy and customize a WebPDF Reader that supports viewing of PDF documents within a web browser. Integrating a WebPDF Reader into a zero foot print application allows end users to view PDF documents without any local applications.

1.1 Why Foxit WebPDF SDK is your choice

Foxit is an Amazon-invested leading software provider of solutions for reading, editing, creating, organizing, and securing PDF documents. WebPDF SDK is a cross-platform solution for PDF online viewing. Currently, WebPDF SDK has been chosen by many of the world's leading firms for integration in their solutions. Customers choose this product for the following reasons:

Full customizable

Developer can easily design a unique style for their WebPDF Reader interface, and make it consistent to their web application.

Easy to integrate

Developers can easily create a plug-in to get and export document methods in the web application, which can be integrated with WebPDF SDK.

Standard and consistent annotation data

The annotations in WebPDF Reader are consistent when viewing in local applications.

Full control of documents

The original document will not be downloaded by the end user, and will be in full control on the server-side. It also supports importing user permissions from a 3rd-party system to set different functions for different users.

Powered by Foxit's high fidelity rendering PDF engine

The core technology of WebPDF SDK is based on Foxit's PDF engine, which is trusted by a large number of well-known companies. Foxit's powerful engine makes document viewing no different on a variety of devices.

In addition, Foxit's products are offered with the full support of our dedicated support engineers if support and maintenance are purchased. Updates are released on a regular basis. Foxit WebPDF SDK will be the most cost-effective choice if you want to develop a cross-platform PDF document viewing solution that can control document distribution.

1.2 Audience and Scope

This document is intended for the developers who need to integrate Foxit WebPDF SDK into their web applications. It includes installation, license, integration, and customization sections.

1.3 Your Web Application

Foxit WebPDF SDK provides a solution that enables a web application to view PDFs seamlessly without any plugins or local applications. Developers should prepare a PDF hosting server before using WebPDF SDK.

1.4 Features

Foxit WebPDF SDK provides most common PDF viewing features and allows developers to incorporate powerful PDF technology to their applications like viewing, text searching, navigating with bookmarks, and annotating PDF documents.

Developers can use the sample Reader interface in the package or develop a custom interface and function that is enabled with web applications.

Features

PDF View	Go to page, Zoom in/out, Fit width, Bookmark accessing, Thumbnail, Print, Rotate
PDF Text	Text selection and copy, Text search
Annotation	Display all exiting annotations 6 annotation tools for editing (Highlight, Underline, Note, Pencil, Typewriter, stamp)
Digital Signature	Provide APIs to developers to integrate third-party digital signatures, including images and ink-like signatures
Acroform	Form filling, export/import PDF form data
Security	Supports password protection watermarks, and digital certificates

1.5 Evaluation

Foxit WebPDF SDK allows users to download the trial version to evaluate SDK. The trial version has no difference from a standard version except for the 30-day limitation for free trial and the trial watermarks in the generated pages. After the evaluation period expires, customers should contact the Foxit sales team and purchase licenses to continue using Foxit WebPDF SDK.

1.6 License

Developers should purchase licenses to use Foxit WebPDF SDK in their solutions. Licenses grant users permissions to release their applications based on WebPDF SDK. However, users are prohibited to

distribute any documents, sample codes, or source codes in the released packages of Foxit WebPDF SDK to any third party without the permission from Foxit Software Incorporated.

2 Getting Started

It is very easy to set up Foxit WebPDF SDK. It takes just a few minutes and we will show you how to use it in a server. The following sections introduce the package's components, installation instructions, license application, and how to use the demo.

2.1 System Requirements

Windows:

Item	Requirement
OS	2008 and 2012 (64 bit)
Processor	Minimum: 8 core @3.6 GHz Recommended: 32 core @2.0 GHz
Memory (RAM)	Minimum: 16GB Recommended: 64 GB
Hard Disk Space	500 MB for installation directory 10 GB for cache directory

Linux:

Item	Requirement
OS	Linux Centos (64 bit) (tested on Centos 7.2)
Processor	Minimum: 8 core @3.6 GHz Recommended: 32 core @2.0 GHz
Memory (RAM)	Minimum: 16 GB Recommended: 64 GB
Hard Disk Space	500 MB for installation directory 10 GB for cache directory

2.2 What's in the package

2.2.1 For Windows

Download the Foxit WebPDF SDK for Windows package and extract it to a new directory "foxitwebpdfsdk_1_3_windows_x64". The components of the release package are shown in **Figure 2-1**.

docs:	API references, developer guide
foxitwebpdfsdk_1_3_windows_x64:	The installation package

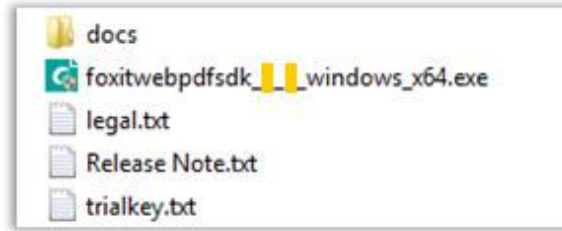


Figure 2-1

2.2.2 For Linux

Download the Foxit WebPDF SDK for Linux package and extract it to a new directory “foxitwebpdfsdk_1_3_linux_x64”. The components of the release package are shown in **Figure 2-2**.

docs:	API references, developer guide
foxitwebpdfsdk_1_3_linux_x64:	The installation package

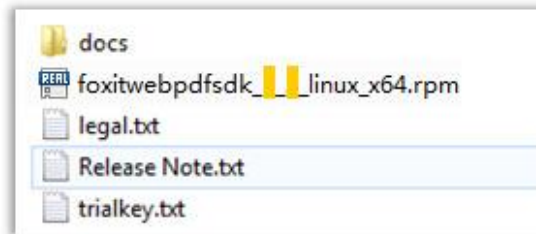


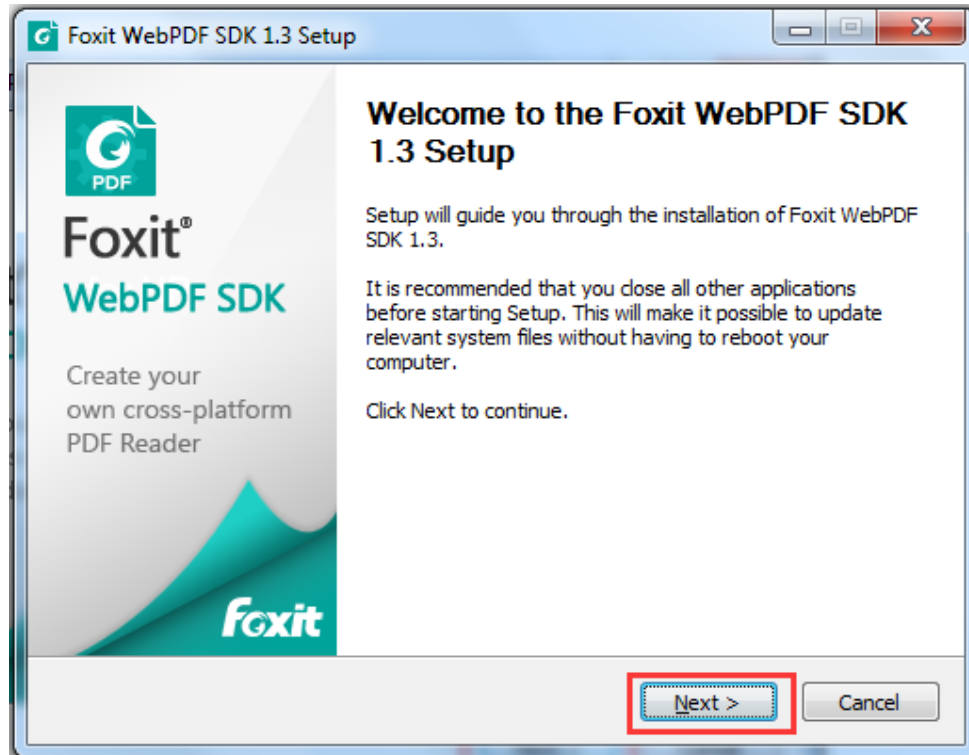
Figure 2-2

2.3 Installation

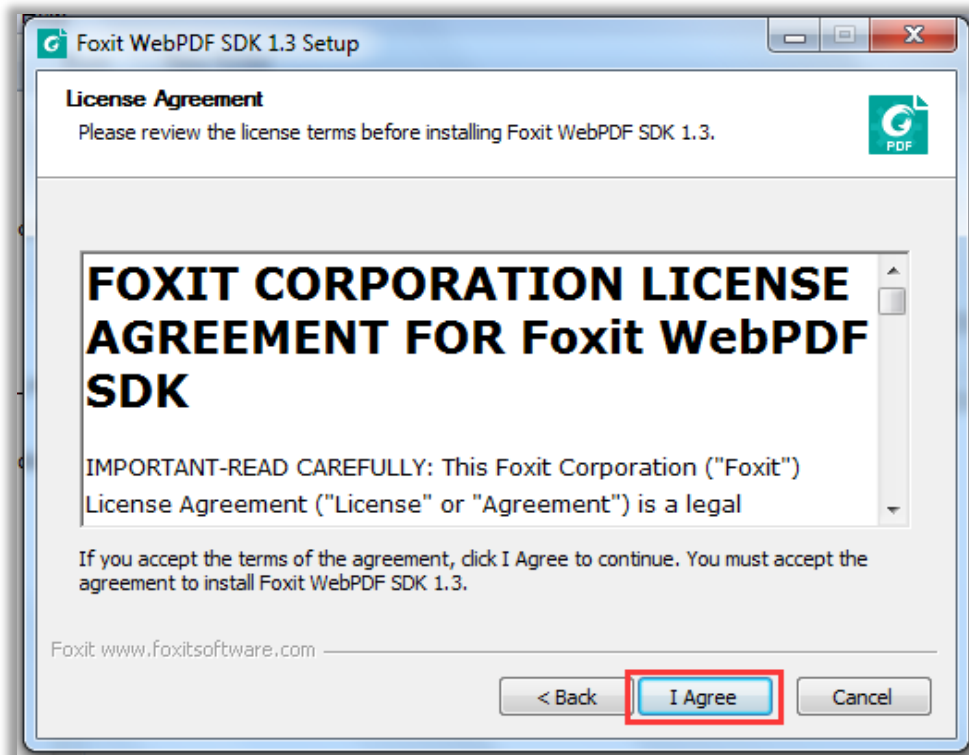
2.3.1 For Windows

2.3.1.1 Installation

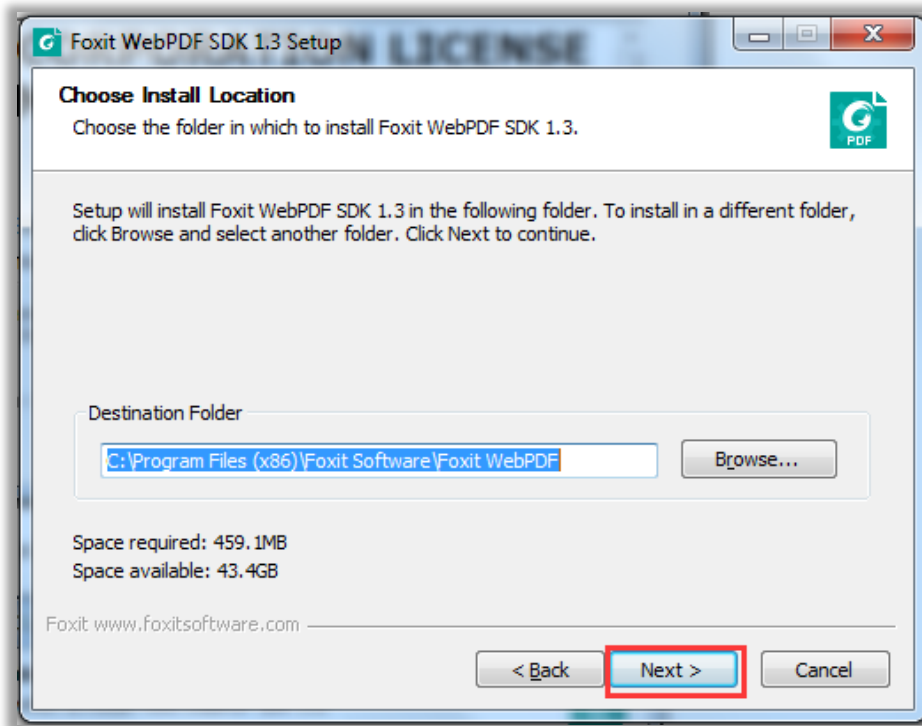
After downloading the package, extract, and double-click the executable file. Then follow the steps below to install WebPDF SDK. Click **Next**.



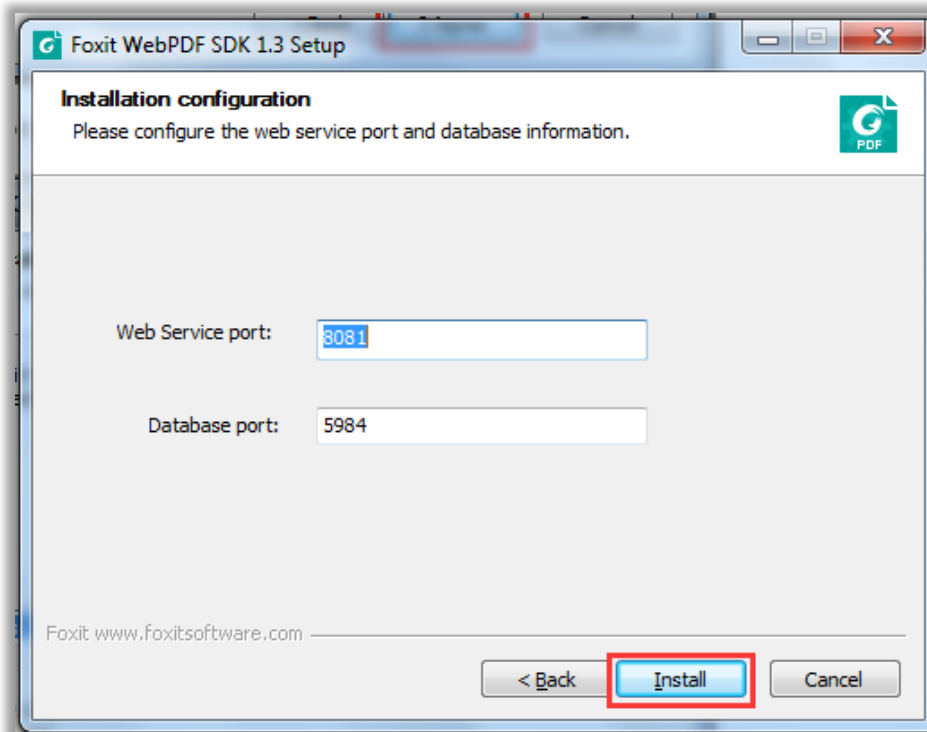
Please read the license terms and click **I Agree** to continue if you accept the terms and conditions of the License Agreement.

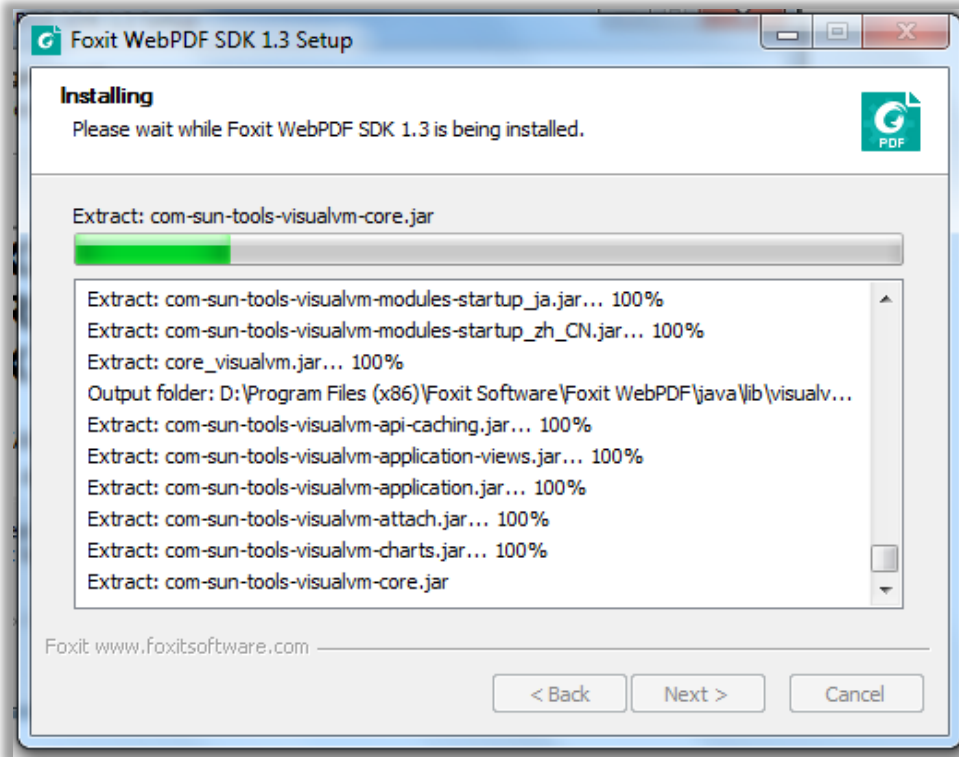


Choose the installation location and click **Next**.

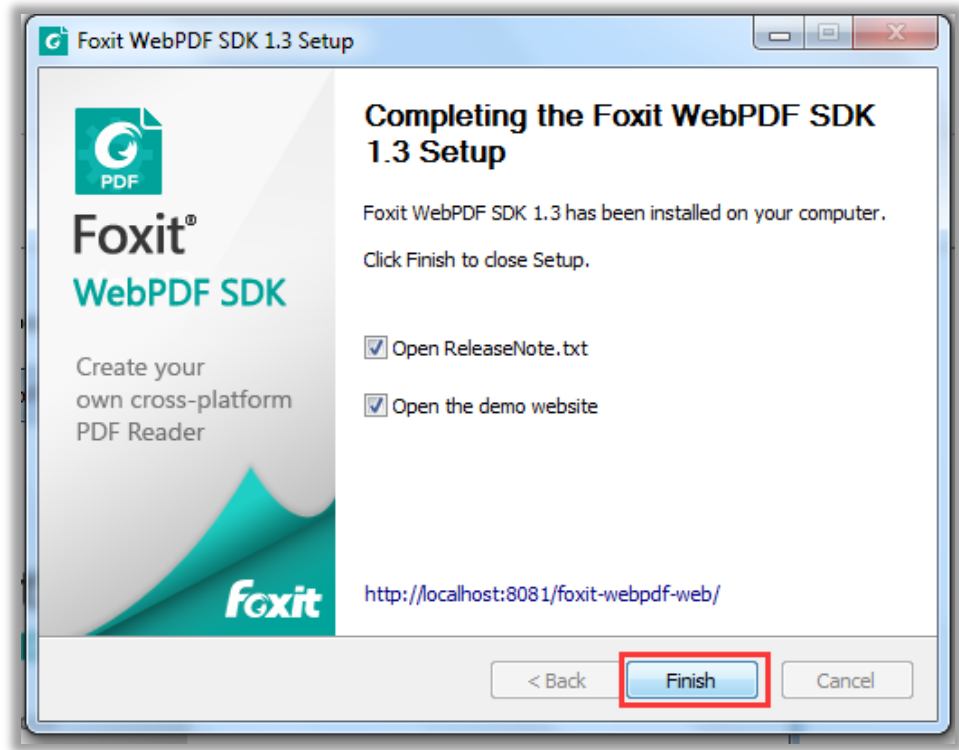


Set a port for the database and Web Service. And then click **Install** to start the installation.

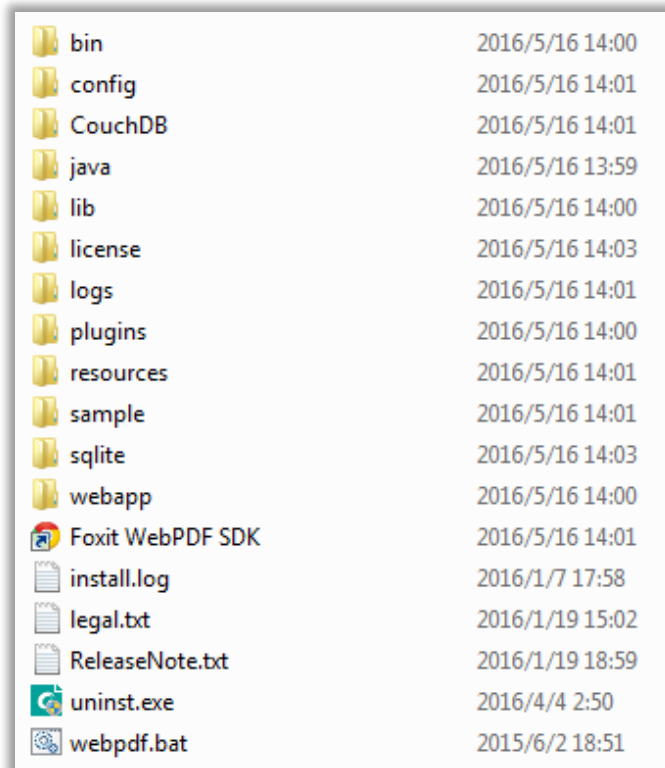




When the process is complete, click **Finish** and WebPDF SDK will be available to run.



After installing WebPDF SDK, the installation directory is shown in the following:

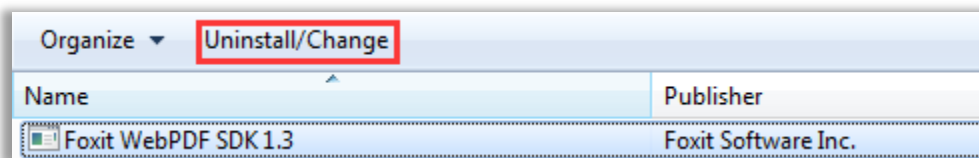


Please make sure the below 2 services are running.

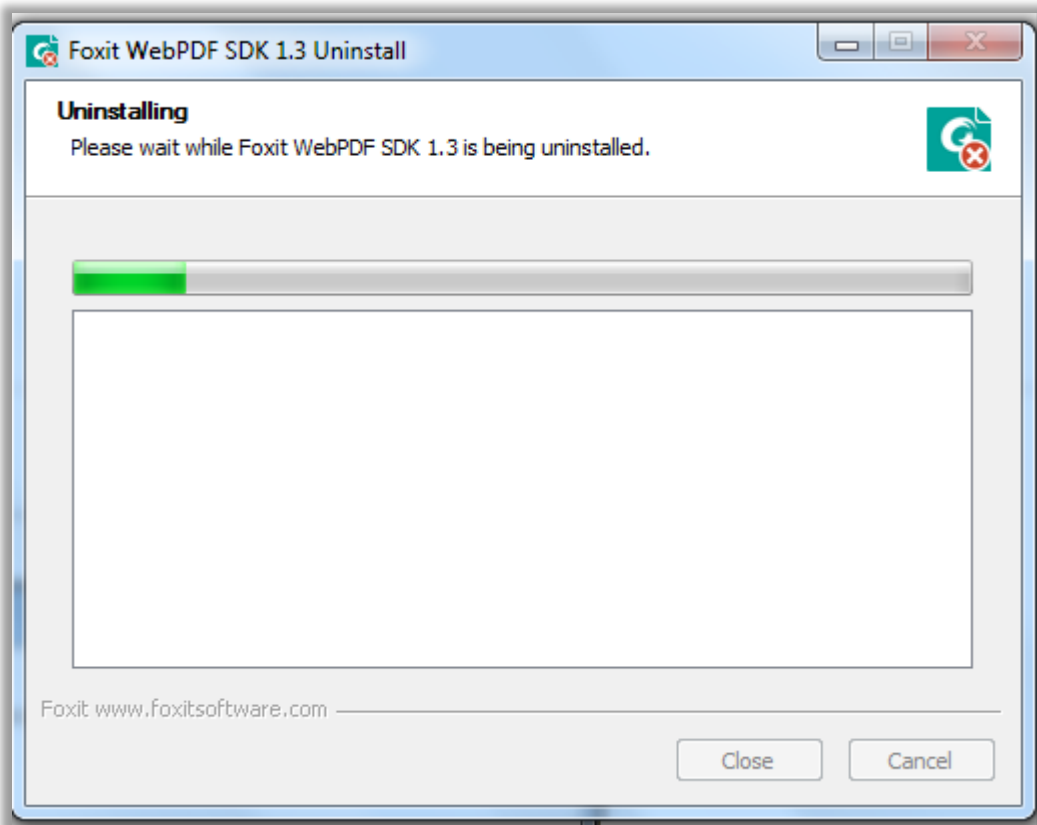
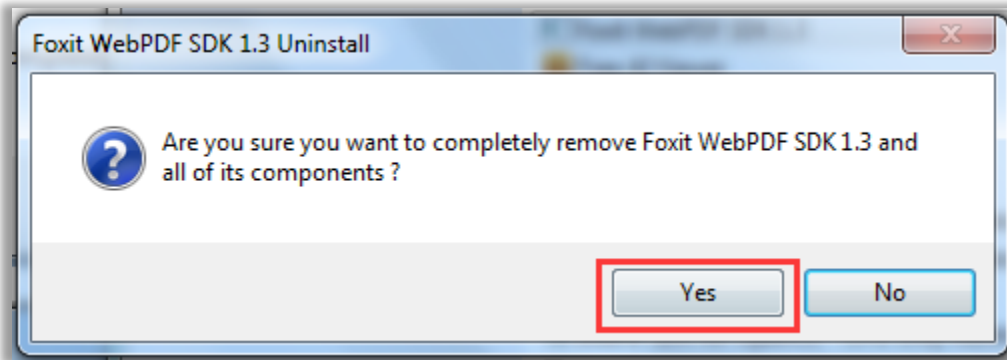
Name	Description	Status	Startup Type	Log On As
Foxit WebPDF Service	Provide PDF...	Running	Automatic	Local Syste...
CouchDB		Running	Automatic	Local Syste...

2.3.1.2 Uninstallation

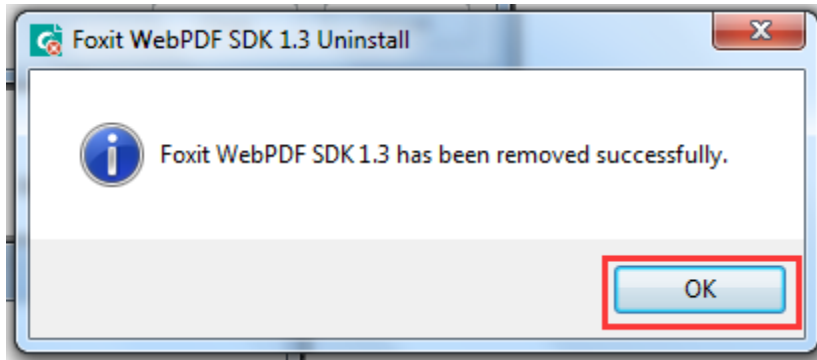
To uninstall the WebPDF solution, please open the Control Panel choose WebPDF SDK and click **Uninstall/Change**.



In the pop-up dialog box, click **Yes** to uninstall.



When the process is complete, click **OK** and WebPDF SDK will be uninstalled successfully.



2.3.1.3 Run WebPDF Reader

After installing WebPDF SDK, you can try WebPDF Demo by visiting this link: <http://localhost:8081/foxit-webpdf-web/>.

Note: “localhost” should be replaced with the corresponding IP or name of the server that has installed WebPDF SDK. The web service port set during installation should be replaced with “8081”. You should follow this note whenever you visit the link mentioned above.

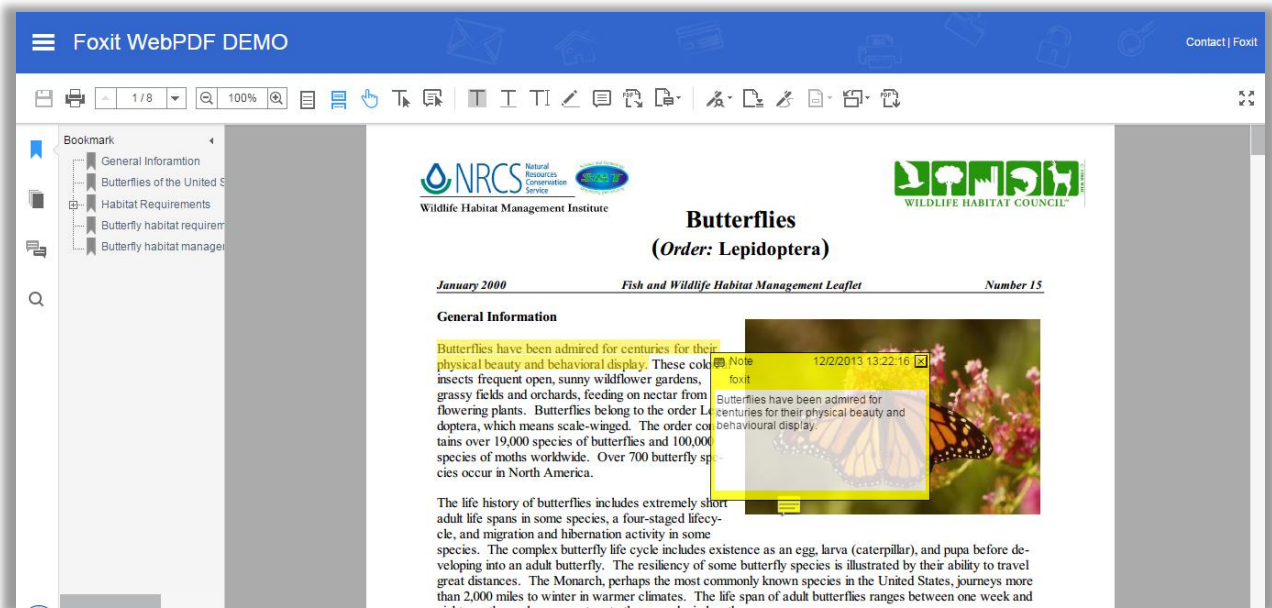


Figure 2-3

Then a web page (See **Figure 2-3**) will be opened. You can try the basic WebPDF SDK features in this demo, such as adding a watermark, annotating PDF files, and so on. Please refer to the User Guide for more information about how to use the DEMO.

Before you open a file in the DEMO, you have to activate the SDK. Refer to [how to activate trial mode](#).

2.3.2 For Linux

2.3.2.1 Installation

After downloading the package, extract, and double-click the rpm file or use a command line to install the package (Take 64-bit version for example):

```
rpm -ivh foxitwebpdfsdk_1_3_linux_x64.rpm
```

Please note that OpenSSL is required before the installation.

2.3.2.2 Uninstallation

Use command line to uninstall WebPDF SDK:

```
rpm -e webpdf-1.3.0
```

2.3.2.3 Run WebPDF Reader

After installing WebPDF SDK, you can try WebPDF Demo by visiting this link: <http://localhost:8080/foxit-webpdf-web/>.

Note: "localhost" should be replaced with the corresponding IP or name of the server that has installed WebPDF SDK. 8081 should be replaced with the web service port set during the installation. You should follow this note whenever you visit the link mentioned above.

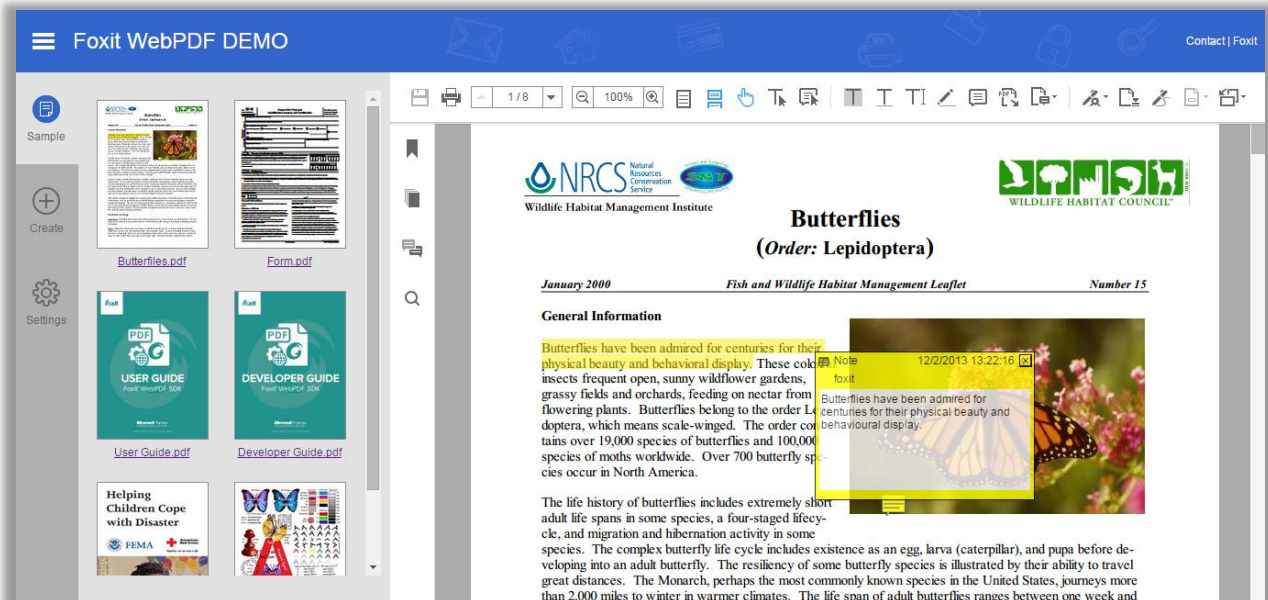


Figure 2-4

Then a webpage (See **Figure 2-4**) will be opened. You can try basic WebPDF Reader (based on the SDK) features in this demo, such as adding watermarks, annotating PDF files, and so on. Please refer to the User Guide to learn more information about how to use the DEMO.

Before you open a file in the DEMO, you have to activate the SDK. Refer to [how to activating a trial mode](#)

2.3.3 Docker container mode deployment

WebPDF SDK provides docker mode deployment. The WebPDF SDK docker image is a portable WebPDF SDK package that includes all the features in the regular WebPDF SDK with all components and dependencies pre-configured. It provides you with a simplified configuration, and helps to increase your productivity for faster deployment. You can try WebPDF SDK's docker image at <https://hub.docker.com/r/foxitsoftware/webpdf/> or follow by steps below.

2.3.3.1 Prepare a CentOS or Ubuntu server (physical or virtual machine)

The tested version: CentOS 7.2, Ubuntu: 16.04.

2.3.3.2 Install docker

Please refer to the document to install docker in different server.

CentOS:<https://docs.docker.com/engine/installation/linux/centos/>

Ubuntu:<https://docs.docker.com/engine/installation/linux/ubuntu/linux/>

2.3.3.3 Obtain the docker image

```
docker pull foxitsoftware/webpdf
```

2.3.3.4 Run the docker image

```
docker run -d -p 8080:8080 --restart=always --name webpdf  
foxitsoftware/webpdf/usr/local/foxitsoftware/webpdf/bin/start.sh
```

2.3.3.5 Run WebPDF Reader

After installing WebPDF SDK, you can try WebPDF Demo by visiting this link: <http://localhost:8080/foxit-webpdf-web/>.

Note: "localhost" should be replaced with the corresponding IP or name of the server that has installed WebPDF SDK. 8081 should be replaced with the web service port set during the installation. You should follow this note whenever you visit the link mentioned above.

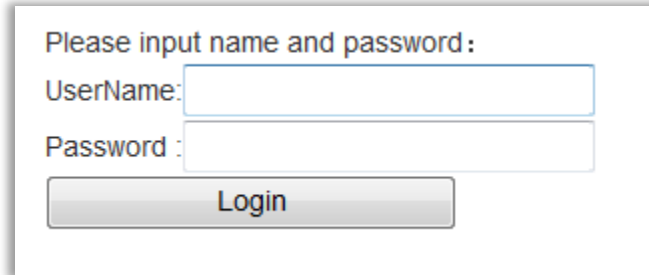
2.3.3.6 Supported Docker versions

This image is officially supported on Docker version 1.10.

Please see [the Docker installation documentation](#) for details on how to upgrade your Docker daemon.

2.3.4 Administrator management

WebPDF SDK provides a set of configuration for users to manage SDK easily. After installing the package, run <http://localhost:8081/foxit-webpdf-web/admin.jsp>



Please input name and password:

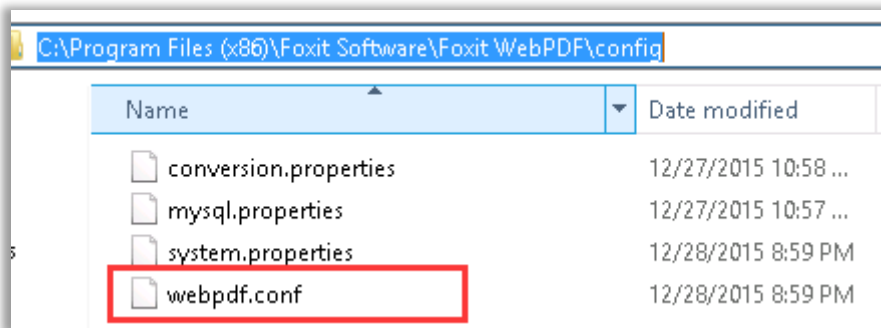
UserName:

Password:

Figure 2-5

Input default user name and password (See **Figure 2-5**): admin, admin.

Note: The default Admin user name and password can be reset. Open and reset in webpdf.conf file which is located in the installation directory .. \Foxit WebPDF\config\webpdf.conf



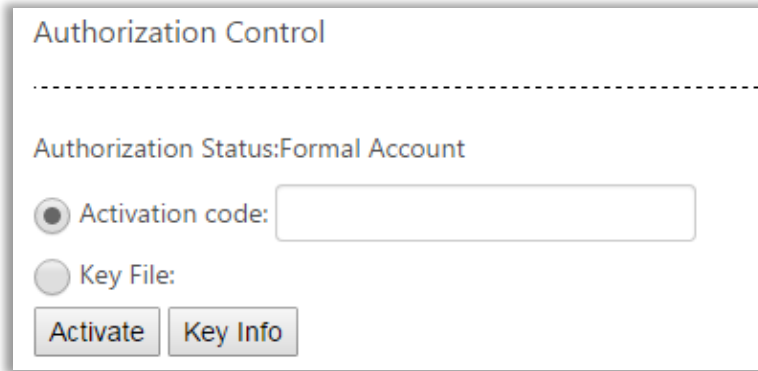
The following sections will introduce the procedures of five main configurations for WebPDF SDK.

2.3.4.1 Apply a license

Authorization Control shows the product license status. The package includes a default 30-days free trial key. After installing WebPDF SDK, activate the trial mode with a trial key file and make it enter the trial mode.

To continue using the product after the trial expires, please contact Foxit and purchase a license of WebPDF SDK.

After receiving the license code from Foxit, import the code in the administrator page to activate WebPDF SDK. After successful activation, the authorization status will be changed to Official, and the trial watermark will disappear when users open files in WebPDF Reader.



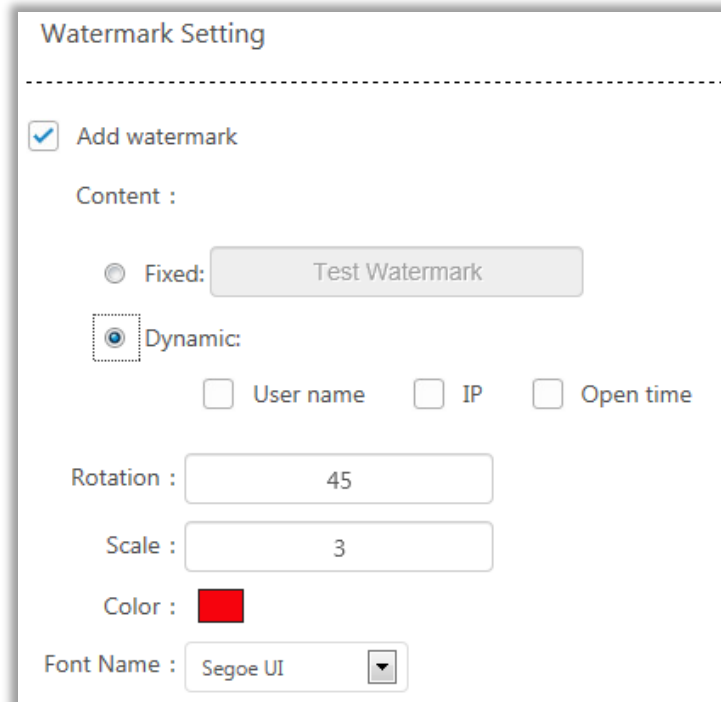
The screenshot shows a dialog box titled "Authorization Control". Below the title is a dashed horizontal line. Underneath, it says "Authorization Status: Formal Account". There are two radio buttons: "Activation code:" which is selected, and "Key File:". To the right of the "Activation code:" radio button is a text input field. At the bottom, there are two buttons: "Activate" and "Key Info".

Figure 2-6

2.3.4.2 Set a watermark

For security reasons or marking purposes, the administrator may want to add a watermark on the file while using WebPDF Reader. WebPDF supports fixed and dynamic watermarks. Set a watermark in the administrator page displayed in **Figure 2-7**.

You can set the options related to watermarks in the Watermark Settings interface, such as Rotation, Color, and Scale. All the PDF files will be added with the set watermark while opening WebPDF Reader.



The screenshot shows a dialog box titled "Watermark Setting". Below the title is a dashed horizontal line. The first option is a checked checkbox labeled "Add watermark". Below this is the label "Content :". There are two radio buttons: "Fixed:" and "Dynamic:". The "Dynamic:" radio button is selected. To the right of the "Fixed:" radio button is a text input field containing "Test Watermark". Below the "Dynamic:" radio button are three checkboxes: "User name", "IP", and "Open time", all of which are unchecked. Below these are three text input fields: "Rotation :" with the value "45", "Scale :" with the value "3", and "Color :" with a red color swatch. At the bottom, there is a "Font Name :" label followed by a text input field containing "Segoe UI" and a dropdown arrow icon.

Figure 2-7

2.3.4.3 Database Setting

The default database, SQLite, has been embedded into the package. WebPDF also supports MySQL. Administrators can reset the database with this setting from the default database to MySQL. If WebPDF Reader is frequently used in your system, or a distributed database is needed subsequently for storage, then we recommend you to choose MySQL.

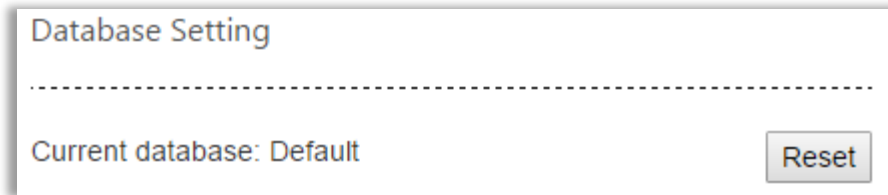


Figure 2-8

To reset the database, click **Reset** to reset the database to MySQL in **Figure 2-9**. Please note that the user account of the database should have remote access permission if MySQL and WebPDF are in different servers.

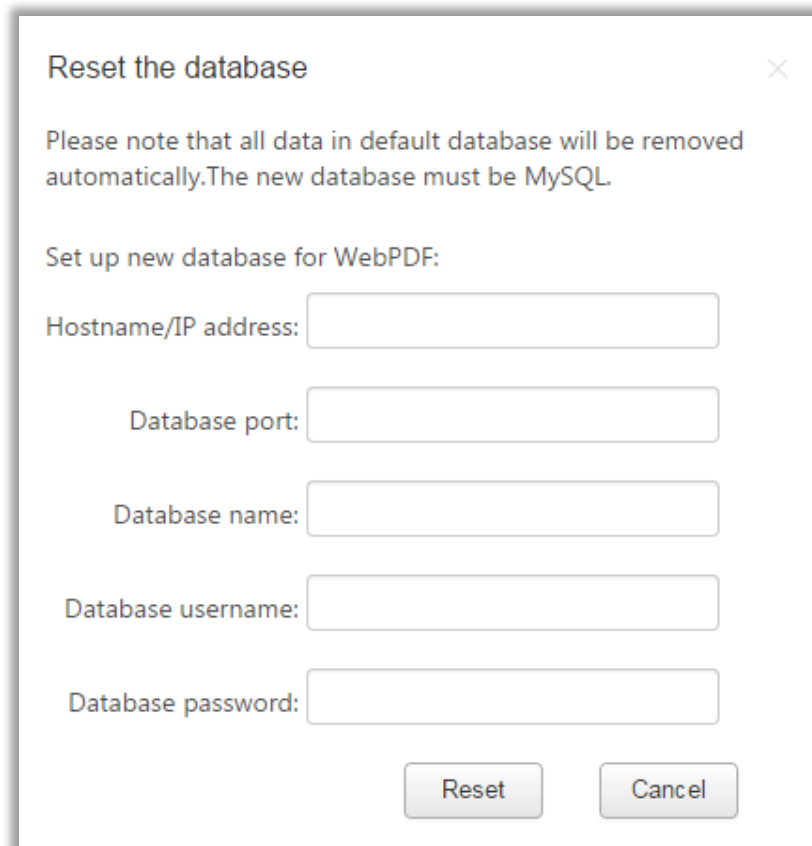


Figure 2-9

Administrators also can reset MySQL to the default database with this setting.

Note: WebPDF doesn't support data migration. We recommend users to fix the database when deploying WebPDF.

2.3.4.4 DPI settings

Administrator can change the output image's DPI value in this setting. If your program requires a high-definition document rendering, a higher value is recommended. But the higher the DPI, the larger an image will be. Moreover, the speed of the front-end will be slower if the network is not good.

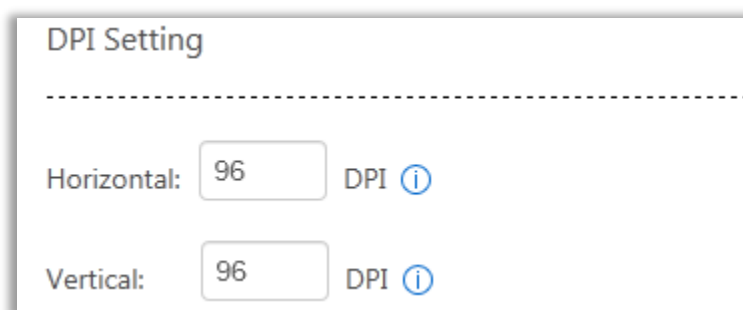
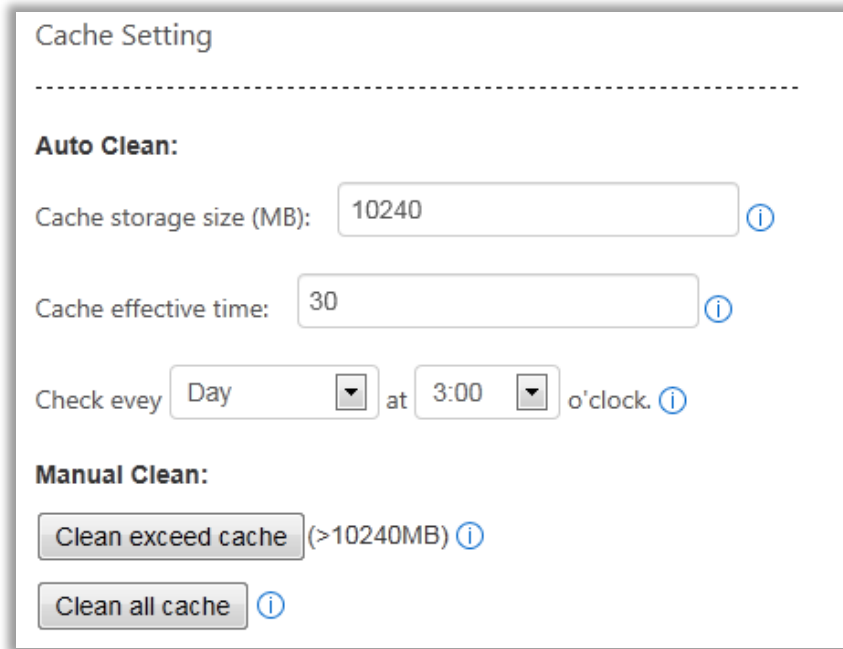


Figure 2-10

2.3.4.5 Manage the Cache

Some data files (such as images, PDF text stream, etc.) will be created when users view PDF files within WebPDF Reader on the server. It will consume some computer memory during data conversion. To reasonably reuse the data files as well as reduce memory consumption, WebPDF Reader will save the images and other converted data files to the internal storage. WebPDF Reader will extract the data files in the cache if the requested file information is stored internally.

Cache storage size	The cache storage size (MB): The default size is 10GB (10240 MB) in an individual server or server farm.
Cache effective time	The time it takes for the cache to be stored. Cache data will be deleted automatically after the effective time. The default value is 30 days. The min. value is 1.



The image shows a 'Cache Setting' dialog box. It has a title bar 'Cache Setting' and a dashed line separator. Under the 'Auto Clean:' section, there are three input fields: 'Cache storage size (MB):' with the value '10240', 'Cache effective time:' with the value '30', and 'Check every' with a dropdown menu showing 'Day', followed by 'at 3:00' with a dropdown menu showing '3:00', and 'o'clock.'. Each of these three fields has an information icon (i) to its right. Under the 'Manual Clean:' section, there are two buttons: 'Clean exceed cache (>10240MB)' and 'Clean all cache'. Both buttons have an information icon (i) to their right.

Figure 2-11

Note: Since all the uploaded documents on the demo site will be stored in the folder “..\Foxit WebPDF\resources\uploaded docs”, please delete old documents regularly to free up space for the new.

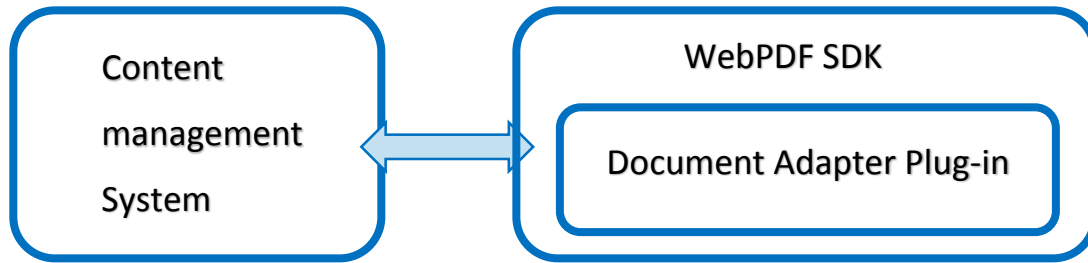
3 Database

By default, the database (SQLite) and Jetty service are embedded in a WebPDF SDK installation package. SQLite is used to store user data, such as access log and annotation data. Administrators can switch SQLite to MySQL. For more details, see also [Database Setting](#).

4 Working with SPI

4.1 Connect PDF files with WebPDF SDK (Document Adapter Plug-in)

When developers integrate WebPDF SDK into their content management system, it needs a method to make the SDK access documents in the system. This section provides a tutorial that guides you to create a Document Adapter Plug-in that acquires PDF files from a system of content management.



4.1.1 Create a Document Adapter Plug-in project

The Document Adapter Plug-in is an extension of the WebPDF SDK. It implements the related interfaces in accordance with the SPI (service provider interface) standard for SDK to access the content management system. To develop the plug-in, please follow the steps below.

1. Create a new java project.
2. Add the “foxit-webpdf-extension.jar” to the project build path. Find the jar file from ..\Foxit WebPDF\webapp\WEB-INF\lib.
3. Develop the Document Adapter Plug-in

To provide this plugin, you need to implement the ‘DocumentPluginSpi’ interface that defines how WebPDF SDK can **get or export PDF files** from a third-party document management system.

The following code shows an example of “DocumentPluginSpi” interface implementation. Please note that it provides a non-argument constructor and implements the **getDocument/exportDocument** methods defined by the SPI. WebPDF also provides a sample plug-in project in the folder “..\Foxit Software\Foxit WebPDF\sample\ foxit-webpdf-demo\src\main\java\com\foxit\webpdf\demo\HttpDocumentPlugin.java”.


```
package com.foxit.webpdf.demo;

import java.io.InputStream;
//.....
/**
 * Get the PDF file via URL.
 *
 */
public class HttpDocumentPlugin implements DocumentPluginSpi {

    @Override
    public FileInfo getDocument(String params) {
        String user = "Anonymous";
        Boolean readOnly = true;
        Boolean disablePrint = false;
        Boolean disableDownload = false;
        String[] sp = params.split("[?]");
        // get values of readOnly, disablePrint, disableDownload from params.
        // ...
        InputStream inputstream = null;
        String version = null;
        long fileSize = 0L;
        try {
            URL url = new URL(params);
            URLConnection conn = url.openConnection();
            inputstream = conn.getInputStream();
            version = conn.getHeaderField("Last-Modified");
            fileSize = Long.valueOf(conn.getHeaderField("Content-Length"));
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
        SimpleFileInfo info = new SimpleFileInfo();
        info.setFileInputStream(inputstream);
        info.setFileSize(fileSize);
        info.setFileName(getDocuemntDisplayName(params));
        info.setReadOnly(readOnly);
        info.setDisablePrint(disablePrint);
        info.setDisableDownload(disableDownload);
        info.setUserId(user);
        info.setUserFriendlyName(user);
        info.setFileUri(params);
        info.setVersion(version);
        return info;
    }

    @Override
    public String exportDocument(String params, InputStream in) {
        return "404";
    }

    // see the sample code form "sample folder" under the installation path.
    //...
}
```

After implementing those interfaces, you need to create a plug-in configuration file and locate it in “META-INF/services” (**directory of java project**). The file’s name is the full-qualified binary name of the service’s type. And the file must be encoded in UTF-8.

The configuration file contains the fully qualified class name of your plugin.

For example, to register the plugin `HttpDocumentPlugin` (the class name developer defined), create a txt file named `com.foxit.webpdf.extension.spi.DocumentPluginSpi`. The content of this text file is:

`com.sample.HttpDocumentPlugin`

4.1.2 Run the Document Plugin

Package the Java project to a JAR file, and copy this JAR file to the “plugins” folder in the WebPDF SDK installation directory.

Restart the “Foxit WebPDF Service”, and the plug-in will be automatically loaded.

The following 2 points are the sample code for how the front end calls on methods. Or you can refer to the sample code from the installation directory “..\Foxit WebPDF\webapp\pc\demo.html”.

4.1.2.1 Get a PDF file

The following example shows how to load a PDF file with the plug-in.

```
/**Get PDF file data
 * @param params URL of PDF file in 3rd file system
 * @ type Specific the plugin type of 3rd file system
 */
url: http://site url/api/file/add
method: POST
body:{
  'params': '/parameter in 3th document system',
  'type': 'com.sample.HttpDocumentPlugin'
}
```

Then the WEBPDF Server will load a PDF file with the plug-in (`com.sample.HttpDocumentPlugin`) and return the PDF Document ID.

4.1.2.2 Export a PDF file

The following example shows how to export a PDF file with the plugin to a third-party file system.

```
/**
 *Export PDF file to 3rd file system
 * @param params the directory in 3rd file system that PDF file export to
 * @param type Specific the plugin type of 3rd file system
 */
url: http://siteurl/export
method: POST
body:{
  'params': '/parameter in 3th document system',
  'type': 'com.sample.HttpDocumentPlugin'
}
```

4.2 Integrate Digital Signature (Signature Plug-in)

WebPDF SDK supports the integration of digital certificates, allowing end users to add digital signatures in PDFs. It needs to develop a Signature Plug-in to implement the “SignaturePluginSpi” interface, which defines how WebPDF SDK adds digital certificates in a PDF document.

4.2.1 Create a Signature Plug-in project

The Signature Plug-in is an extension of the WebPDF SDK. It implements the related interfaces in accordance with the SPI (service provider interface) standard for SDK in order to integrate third-party digital certificates for digital signature solutions. To develop the plug-in, please follow the steps below.

1. Create a new java project.
2. Add “foxit-webpdf-extension.jar” to the project build path. Find the JAR file from the folder “..\Foxit Software\Foxit WebPDF\webapp\WEB-INF\lib”.
3. Develop the Signature Plug-in.

Developers need to implement the “SignaturePluginSpi” interface. The “SignaturePluginSpi” interface provides a non-argument constructor and implements the **getSignature/getContents/exportDocument** methods defined by the SPI. WebPDF provides a sample plug-in project in the folder “..\Foxit Software\Foxit WebPDF\sample\ foxit-webpdf-demo\src\main\java\com\foxit\webpdf\demo\SignaturePlugin.java” for your information.

4.2.2 Run the Signature Plug-in

Package the Java project to a JAR file, and copy this JAR file to the "plugins" folder of the WebPDF SDK installation directory.

Restart the “Foxit WebPDF Service”, and the plug-in will be automatically loaded.

Refer to the specific code in the installation directory “.. \ Foxit WebPDF\webapp\webapp\scripts\reader\control\pc\signature.js” and reference the API to run the Signature Plug-in.

```
function signatureSign(signature, location, reason, imgUrl,
cerType, straddleType, percent, pageRanges, pos)
```

4.3 Integrate Document Management System User Accounts (System User Plug-in)

Integrating user accounts of a third-party document management system into WebPDF SDK will allow users to save their own data from the WebPDF Reader, such as annotations, ink signature templates, form data, etc. It also supports to importing user permissions from document management systems - WebPDF Reader will then offer different functionality depending on each user’s permission settings. To implement this, developers should develop a user plug-in for the document management system to implement the “SystemUserPluginSpi” interface, which defines how WebPDF SDK accesses the user information of the document management system.

4.3.1 Create a System User Plug-in

The User Plug-in of the document management system is an extension of the WebPDF SDK. It implements the related interfaces in accordance with the SPI (service provider interface) standard for the SDK to access the user information of the document management system. To develop the plug-in, please follow the steps below.

1. Create a new Java project
2. Add “foxit-webpdf-extension.jar” to the project build path. Find the jar file from the folder “..\Foxit Software\Foxit WebPDF\webapp\WEB-INF\lib”.
3. Implement the “SystemUserPluginSpi” interface

The “SystemUserPluginSpi” interface provides a non-argument constructor and implements the **getSystemUserInfo/getUserPermission** methods defined by the SPI. WebPDF provides a sample plug-in project in the folder “..\Foxit Software\Foxit WebPDF\sample\ foxit-webpdf-demo\src\main\java\com\foxit\webpdf\demo\SystemUserPlugin.java”.

4.3.2 Run the System User Plug-in

Package the Java project to a JAR file, and copy this JAR file to the “plugins” folder of the WebPDF SDK installation directory.

Restart the “Foxit WebPDF Service”, and the plug-in will be automatically loaded.

4.3.2.1 Ink signature template list

Here is an example for how to create an Ink signature template list associated with a user. User information is retrieved from the system user plug-in. Refer to the specific code in the installation directory “..\Foxit WebPDF\webapp\scripts\reader\control\pc\inkSign.js” and reference the API to run the system user Plug-in.

```
var params = {
    "ip":true
};
var paramData = {"plugin": "com.foxit.webpdf.demo.SystemUserPlugin","params":
JSON.stringify(params)};
// get the ink sign plugin object
_inkSignPlugin =
WebPDF.ViewerInstance.getPluginByName(WebPDF.InkSignPluginName);
// get the ink signature template list manager object
_inkSignListMgr = new InkSignListManager(baseUrl);
$.ajax({
    type: "GET",
    url: getUserUrl,
    dataType: 'json',
    data:paramData,
    success: function (response) {
        if(response.resultCode!=0){
            WebPDF.alert(null, i18n.t("InkSign.SigAlertTitle"),
i18n.t("InkSign.GetUserIdFalied"));
            return;
        }else{
            // obtain current system user id from spi interface,
            // please refer to demo code at 'SystemUserPlugin.java'
            var userId = response.resultBody.id;
            // acquire current ink signature list data
            _inkSignListMgr.initList(userId,showInkSignList);
        }
    },
    error: function (msg) {
        // need to alert the failure.
        WebPDF.alert(null, i18n.t("InkSign.SigAlertTitle"),
i18n.t("InkSign.GetUserIdFalied"));
    }
});
```

4.4 Import/export form data (XML) to/from a remote server (Form Plug-in)

If you develop an online form filling solution, you probably want to collect all the filling data from responders to a single server for centralized management. WebPDF SDK supports importing and exporting form data from/to a remote server with Form Plug-ins that define how the data is

transferred.. It needs to develop a Form Plug-in to implement the “FormPluginSpi” interface, which defines how WebPDF SDK import/export form data from/to a remote server.

4.4.1 Create a Form Plug-in project

The Form Plug-in is an extension of the WebPDF SDK. It implements the related interfaces in accordance with the SPI (service provider interface) standard for SDKs in order to export/import form data to/from a remote server. To develop the plug-in, please follow the steps below.

1. Create a new Java project.
2. Add “foxit-webpdf-extension.jar” to the project build path. Find the JAR file from the folder “..\Foxit Software\Foxit WebPDF\webapp\WEB-INF\lib”.
3. Develop the Form Plug-in.

Developers need to implement the “FormPluginSpi” interface. The “FormPluginSpi” interface provides a non-argument constructor and implements the **exportData/importData** methods defined by the SPI. WebPDF provides a sample plug-in project in the folder “..\Foxit Software\Foxit WebPDF\sample\ foxit-webpdf-demo\ src\main\java\com\foxit\webpdf\demo\FormPlugin.java” for your information.

4.4.2 Run the Form Plug-in

Package the Java project to a JAR file, and copy this JAR file to the "plugins" folder of the WebPDF SDK installation directory.

Restart the “Foxit WebPDF Service”, and the plug-in will be automatically loaded.

5 Working with API

Foxit WebPDF SDK provides a series of APIs for developers to create their own WebPDF Reader. There are two ways to implement this. One is that developers create an HTML page for the WebPDF Reader interface and then create a DIV tag as the container for the main viewer object in the HTML. Alternatively, developers can edit the FTL file to implement WebPDF Reader. The details about the two methods will be introduced in the following sections.

How to load a viewer object

A viewer object is the core component for rendering the document and its page displaying area in a WebPDF Reader. Create a DIV element in a HTML file that used for PDF document rendering with the reserved ID being “docViewer”. The HTML file should also include some other elements for the toolbar, status bar and the navigation panels.

Use the global function WebPDF.createViewer to create a viewer object in order to pass the reserved ID “docViewer” and the parameters for document rendering.

- After the document is loaded, complete the implementation of the navigation panels.

Check the following code for the createViewer function to initialize a viewer object. Please refer to the detailed code in ..\Foxit WebPDF\webapp\pc\demo.html.

```
function createViewer(url,newTab) {
    if (url == null) return;
    seajs.use(['../scripts/reader/release/webpdf.mini.js?v=20131209'],
function (init) {
    var options = {
        language: language,
        accessToken: "accessToken",
        url: url,
        scrollBarType: 0
    };
    var pos = url.indexOf("asserts");
    var baseUrl = url.substr(0, pos);

    WebPDF.createViewer("docViewer", options);
    initUIEvent(baseUrl,options,newTab);
    WebPDF.ViewerInstance.setWatermarkInfo(getWatermarkConfigs());
    WebPDF.ViewerInstance.load();

    });
}
```

5.1 WebPDF Reader on PC

This section will introduce how to use WebPDF SDK APIs on the PC sample.

The following files include the main reference codes used in this section.

```
File 1: ..\Foxit WebPDF\webapp\pc\demo.html
File 2: ..\Foxit WebPDF\webapp\scripts\reader\control\pc (All the JavaScript
files)
File 3: ..\Foxit WebPDF\webapp\styles\reader\pc
File 4: FoxitWebPDF_APIREFERENCE (API file)
```

5.1.1 Open a PDF file on your own system

Once the [adapter plug-in](#) named com.foxit.webpdf.demo.HttpDocumentPlugin has been created, the PDF files in the network can be opened by transferring the PDF URL to WebPDF Reader.

According to the demo HTML file, modify the parameter to “com.foxit.webpdf.demo.HttpDocumentPlugin”, and the PDF files in the corresponding system can then be opened with the “api/file/add” interface.

Check the following code for the openFile function. Please refer to the detailed code in ..\Foxit WebPDF\webapp\pc\demo.html.

Example: Open a PDF file

```
//.....
var loadParams = baseUrl + "?user=" + user
    + "&readOnly=false&disablePrint=false&disableDownload=false";
// Set the load type of SPI plugin. HTTP means demo plugin implement to
process the document from http protocol.
var loadType = " com.foxit.webpdf.demo.HttpDocumentPlugin ";
var data = {
    params: loadParams,
    type:loadType
};
// request to access demanded PDF file.
$.ajax({
    url: baseUrl + "api/file/add",
    type: "POST",
    data: data,
    async:false,
    success: function (data) {
        //.....
        if(typeof(WebPDF) != 'undefined' && WebPDF.ViewerInstance !=
null) {
            // open current file
            WebPDF.ViewerInstance.openFile(url);
            //.....
        }
    },
    //.....
});
```

Common APIs of Viewer:

API Name	Description
exportDocument()	Export documents to local browsers
exportToFDF()	Export annotations of the document into a FDF file
focus()	Focus the viewport so it can be natively scrolled with the keyboard
getBookmark()	Get Bookmark
getCurPageIndex()	Get the current page index
getCurZoomLevel()	Get the current zoom level

getFileID()	Get the ID of current viewer instance
getFileName()	Return the name of the current file.
getPageCount()	Get the page counts of the document
getPluginByName()	Get an instance according to the name of the plug-in
getThumbnail()	Get the URL of the Thumbnail by page index
getToolHandlerByName()	Get the tool handler by name.
getUserConfig()	Get the configuration data of the current user.
getUserName()	Get the name of the current user
getWatermarkInfo()	Get user watermark information
getZoomLevels()	Get the list of available zoom levels
gotoNextPage()	Go to the next page.
gotoPage()	Jump to a page
gotoPageByDestination()	Go to the page of the destination specified by the page position
gotoPrevPage()	Go to the previous page
hasForm()	Detect whether the document has form fields
highlightText()	Highlight an area by rectangle
isDocLoaded()	Check whether the document has been loaded
isFitWidth()	Check whether the current zoom level is WebPDF.ZOOM_FIT_WIDTH
load()	Initiate loading of document assets
off()	Removes an event handler from a given event If the handler is not provided; remove all handlers of the given type
on()	Adds a new event handler for a particular type of event
openFile(url)	Open a PDF file by URL.
print()	Open the Print dialog
rotate()	Rotate the page view.
saveAnnots()	Save annotation to the database
searchAllText()	Full-text search This function obtains the data in an asynchronous way. The data will be returned in the callback SEARCH_ALL_FINISHED, Please refer to the event WebPDF.EventList.SEARCH_ALL_FINISHED
searchText()	Search text
setCurrentToolByName()	Set the current tool by its name

setLayoutShowMode()	Change the page layout mode of the viewer
setUserConfig()	Set the current user configuration.
setWatermarkInfo()	Set watermark information
showAnnots()	Set all annotations as visible or hidden in the document.
updateLayout()	Update the layout
zoomTo()	Zoom to the given value

5.1.2 Annotation

An annotation associates an object such as underline, highlight, or note with a location on a page of a PDF document. It provides a way to interact with users by means of the mouse and the keyboard. Foxit WebPDF SDK supports annotation types including note, underline, highlight, typewriter, and pencil. It provides APIs to create, access, and delete annotations. Call the “WebPDF.ViewerInstance.setCurrentToolByName()” interface, which is defined in the API, to switch between different annotation tools. Please refer to the detailed code in “toolbar.js”.

Example: Set highlight as the current tool

```
WebPDF.ViewerInstance.setCurrentToolByName(WebPDF.Tools.TOOL_NAME_COMMENT_HIGHLIGHT)
```

5.1.3 Save Annotation

After editing annotations in the document, users should save the changes. Call the “ViewerInstance.saveAnnots()” interface which is defined in the API to save the annotations. Please refer to the detailed code in “toolbar.js”.

Example: Click the Save button

```
$("#btnSave").click(function(){
    if(!_isDocModified) {
        WebPDF.ViewerInstance.saveAnnots();
    }
});
```

5.1.4 Rotation

Users may need to rotate the page view if a document is not in a normal layout. To rotate a page view, call the “WebPDF.ViewerInstance.rotate()” interface in the API. Please refer to the detailed code in “toolbar.js”.

Example: Click the Rotate Left menu

```
$("#btnLeftRotate").click(function(){  
    WebPDF.ViewerInstance.rotate(WebPDF.ROTATE_LEFT);  
});
```

5.1.5 Download

Users may want to download PDF documents to a local disk for further use in WebPDF Reader. To download a PDF document, call “WebPDF.ViewerInstance.exportDocument()” in the API. Please refer to the detailed code in “toolbar.js”.

Example: Click the Download button

```
$("#btnExportPDF").off("click").click(function(){  
    WebPDF.ViewerInstance.exportDocument(null);  
    return false;  
});
```

5.1.6 Print

Users may want to print out the PDF document. To do this, call the “WebPDF.ViewerInstance.printt()” interface in the API. Please refer to the detailed code in “toolbar.js”.

Example: Click the Print button

```
$("#btnPrint").click(function(){  
    WebPDF.ViewerInstance.print();  
});
```

5.1.7 Signature

WebPDF SDK supports the integration of digital certificates to provide the digital signatures. Call “WebPDF.ViewerInstance.setCurrentToolByName()” to set signature as the current tool for signature adding. Please refer to the detailed code in “signature.js”.

5.1.7.1 Set the signature properties

According to the demo HTML, after defining a signature area, it will trigger the callback interface of “WebPDF.EventList.SIGNATURE_ADD”. Set the signature properties in this callback. Please refer to “signature.js” for the properties information set in the demo. It registers the callback interface “WebPDF.EventList.SIGNATURE_ADD” and pops up the properties dialog box of the signature.

Example: Trigger the callback interface of WebPDF.EventList.SIGNATURE_ADD

```
WebPDF.ViewerInstance.on(WebPDF.EventList.SIGNATURE_ADD,
function(event,data){
    if(WebPDF.SignatureType.Normal == data.sigType)
        //Show the normal signature properties dialog if adding a normal signature
        _showNormalSigDialog(data.sig);
    Else
        //Show the cross-page signature properties dialog if adding a cross-page
signature
        _showStraddleSigDialog(data.sig);
})
```

Call WebPDF.SingaturePlugin.getSignatureInfo for getting the properties of the signature. Please refer to “getSingatureInfo()” in the API file.

5.1.7.2 Add a signature

To add a signature, call the “WebPDF.SingaturePlugin.sign()” interface using the signature information returned from getSignatureInfo(). By default, the package includes the “com.foxit.webpdf.demo.SignturePlugin” plug-in to add a certificated signature. Save the signed document under the “resources\signature\singed docs” folder after signing. To open the signed document, transfer a successful callback to the “sign()” interface, and it will return the URL of the signed document. Use the SPI “api/file/add” to open the signed file. Please refer to the detailed code in “signature.js”.

Example: Add a signature

```
$("#" + _sigSettingDlgID).hide();
signatureSign(signature, location, reason, imageUrl, cerType);
```

Example: Open a signed document

```
$('#sigViewbutton').off("click").on("click", function() {
    openSignedFile(baseUrl, fileUrl);
    $("#" + _sigSigSuccessDlgID).hide();
    $signatureDlgLayer.hide();
});
```

5.1.7.3 Check the signature properties

Double-click a signature object to trigger the callback interface of “WebPDF.EventList.SIGNATURE_PROPERTIES” in WebPDF Reader. It will return the properties of the signature. Please refer to the detailed code in “signature.js”.

Example: Double-click a signature to show the properties

```
on(WebPDF.EventList.SIGNATURE_PROPERTIES, function(event,data){
    _showSignatureProperties(data.signer,data.reason,data.location,data.time);
});
```

Common APIs of Signature:

API Name	Description
cancelSign()	Cancel the current signature process
getSignatureInfo()	Get a json format string for signature information
sign()	Add a digital signature and save as a new file

5.1.8 Ink Signature

Additionally, WebPDF supports Ink Signatures, allowing users to add a signature with their own handwriting. Ink Signatures can be added with or without a digital certificate.

5.1.8.1 Sign with handwriting without a certificate

To add an ink signature without a certificate, call the “WebPDF.InkSignPlugin.sign()” interface using the signature information returned from getInkSignInfo(). The data of the ink signature will be saved to the database in a similar method as annotation data through the “WebPDF.InkSignPlugin.save()” interface. Please refer to the detailed code in “inksign.js”.

Example: Add an ink signature without certificate

```
var certType = data.inkSign.sigJSONData.inkData.certType;
// application can set reason, location hereby.
var reason = null;
var location = null;
var imageData = data.canvasImgData;

if(certType==""){
    // Non certificate sign
    var inkSignPlugin =
WebPDF.ViewerInstance.getPluginByName(WebPDF.InkSignPluginName);
    inkSignPlugin.sign(data.inkSign);
}else{
    // Certificate of sign
    inkSignSign(data.inkSign, location, reason, imageData,
certType);
}
```

5.1.8.2 Sign with handwriting with a certificate

To add an ink signature with a certificate, call the “WebPDF.InkSignPlugin.signWithCert ()” interface using the signature information returned from getInkSignInfo(). By default, the package includes the

“com.foxit.webpdf.demo.SignaturePlugin” plug-in to add a certificated signature. The signed document will be saved under the “resources\signature\singed docs” folder after signing. To open the signed document, transfer a successful callback to the “signWithCert()” interface, and it will return the URL of the signed document. Use the SPI “api/file/add” to open the signed file. Please refer to the detailed code in “inkSign.js”.

Example: Add an ink signature with a certificate

```
var paramsObj = {
    cert : cerType
};
var paramsStr = JSON.stringify(paramsObj);
var sinfoObj = {
    imgData: imgData,
    cert : cerType
};
var sinfoStr = JSON.stringify(sinfoObj);
var inkSignPlugin =
WebPDF.ViewerInstance.getPluginByName(WebPDF.InkSignPluginName);
var inkSignStr = inkSignPlugin.getInkSignInfo(inkSign,location,
reason);

var data = {
    plugin : 'com.foxit.webpdf.demo.SignaturePlugin',
    properties : inkSignStr,
    information : sinfoStr,
    contentParams : paramsStr,
    exportParams : WebPDF.ViewerInstance.getFileName(),
    success : successSignCallBack,
    fail : failSignCallBack
};
inkSignPlugin.signWithCert(data);
_showInkSignSigingDialog();
```

Common APIs of Signature:

API Name	Description
cancelSign()	Cancel the current signature process
getCurInkSignature()	Get the information of current ink signature
getInkSignInfo()	Get the <u>json</u> format string for the current ink signature information
getInkSignData()	
replace()	Replace the selected ink signature template to the current signature

Save()	Save ink signature without a certificate to the server
sign()	Apply the current ink signature without a certificate
signWithCert()	Apply the current ink signature with a certificate

5.1.9 Stamp

Users may want to give document reviewers a mark about the document's status or sensitivity. They can use the stamp tool to achieve this. WebPDF SDK supports standard stamps as well as dynamic stamps, which can obtain information from a server including name, date and time. To add a stamp, call the "setCurrentStamp()" interface. Please refer to the detailed code in "stamp.js".

Example: Set information of the current stamp object

```
var stamp = _stampListMar.getStamp(title,index);
WebPDF.ViewerInstance.setCurrentToolByName(WebPDF.Tools.TOOL_NAME_STAMP);
var stampToolHandler =
WebPDF.ViewerInstance.getToolHandlerByName(WebPDF.Tools.TOOL_NAME_STAMP);
stampToolHandler.setCurrentStamp(stamp);
```

Common APIs of stamps:

API Name	Description
setCurrentStamp()	Set the information of the current stamp object.
setAddStampCallback ()	Set a callback when user applies a stamp to the document.

5.1.10 Form

From version 1.2, WebPDF SDK supports Acroform filling, as well as importing and exporting form data. Call the "WebPDF.FormPlugin" interfaces to export/import form data or to save form data to the database. Please refer to the detailed code in "form.js" and "toolbar.js"

Export or import form data in XML format

```

$("#btnFormSwitch").parent().show();
//Export Form to XML
$("#btnExportForm").removeClass("disabled");
$("#btnExportForm").on("click", function () {
    var formPlugin =
WebPDF.ViewerInstance.getPluginByName(WebPDF.FormPluginName);
    if(formPlugin) {
        formPlugin.exportXML();
    }
});
//Import Form to XML
$("#btnExportForm").removeClass("disabled");
$("#importFormXMLForm").css("display", "");
/*$("#btnImportForm").on("click", function () {
    $("#inputImportFormXML").click();
});*/

```

Common APIs of Signature:

API Name	Description
exportXML()	Export form data (XML format) to the local machine
exportXMLToUrl()	Export form data (XML format) to a remote server (url).
importXML()	Import form data (XML format) from local machine
importXMLFromUrl()	Import form data (XML format) from a remote server (url).
highlight()	Highlight the form fields
save()	Save the current form data to server side

5.1.11 Bookmark

Users may want to use bookmarks for fast page navigation in a PDF document. Call the “WebPDF.ViewerInstance.getBookmark()” interface for getting a bookmark object. Developers can get the information and events relevant to the bookmark from this object. Please refer to the detailed code in “bookmark.js”.

Example: Go to Page by clicking a bookmark

```

var count = _bookmarkInstance.countActions();
for(var i = 0; i<count ;i++ ){
    var action = _bookmarkInstance.getAction(i);
    var des = action.getDestination();
    WebPDF.ViewerInstance.gotoPageByDestination(des);
}

```


Common APIs of Bookmark:

API Name	Description
countActions()	Count the actions of the current bookmark node
getAction()	Get the specified bookmark action
getNode()	Get bookmark node by ID
getNodeID()	Get the ID of the current bookmark node
getParentNodeID()	Get the parent node ID.
getTitle()	Get the title of the current node
isRoot()	Check whether the current bookmark item is the root or not
moveToFirstChild()	Move to the first child of the PDF bookmark if any
moveToNextSibling()	Move to the next sibling of PDF bookmark if it any
moveToParent()	Move to the parent of PDF bookmark if exists
moveToRoot()	Move to the root of PDF bookmark

5.1.12 Thumbnail

Users may want to use thumbnails for fast page navigation in a PDF document. To achieve this, call the “WebPDF.ViewerInstance.getThumbnail()” interface, which defines the page index of a thumbnail. Developers can get the information and events relevant to the page thumbnail from this object. Please refer to the detailed code in “thumbnail.js”.

Example: Go to Page by clicking a thumbnail

```
var imageUrl = $("#" + imageID).attr("src");
if (imageUrl == null || imageUrl == undefined || imageUrl == '') {
    loadingImgUrl = baseUrl + "images/reader/loading.gif";
    $("#" + imageID).attr("src", loadingImgUrl);
    WebPDF.ViewerInstance.getThumbnail(currentIndex, imageID);
}
```

5.1.13 Comment list

Users may want to view all comments in a document in a single list jump to a comment by clicking on it. WebPDF SDK provides APIs to create a comments panel list to load, edit, jump to, and delete comments in the list. Please refer to the detailed code in “commentlist.js”.

Example: Load more annotations

```
$('.dtree-content').off('click', '.more').on('click', '.more', function() {
    if (clickMore) {
        clickMore = false;
        commentListPlugin.loadAnnots(function(annotDatas, annotCount, isMore) {
            addCommentlist(annotDatas, annotCount, isMore);
            resizeScrollBar();
        }, $.noop);
    }
});
```

Example: Delete an annotation from comment list

```
WebPDF.ViewerInstance.on(WebPDF.EventList.ANNOT_DELETED, function(event,
data) {
    deleteAnnot(data.annotName, data.pageIndex);
    var count = commentListPlugin.getAnnotsCount();
    count--;
    commentListPlugin.setAnnotsCount(count);
    updateAnnotCount();
    resizeScrollBar();
});
```

Example: Go to the specific page by clicking an annotation

```
$('.dtree-content').off('click', '.annot-class').on('click', '.annot-
class', function(event) {
    $('.dtree-content>ul>li>ul>li').removeClass('selected');
    $(this).addClass('selected');
    var pageIndex = $(this).attr('page-index');
    var annotName = $(this).attr('annot-name');
    commentListPlugin.gotoAnnotPosition(pageIndex, annotName);
    resizeScrollBar();
});
```

Common APIs of Comment list:

API Name	Description
setAnnotContent()	Set the content of annotation pop-ups
deleteAnnot()	Delete an annotation from the comment list.
loadAnnots(doneHandler, failedHandler)	WebPDF does not load all the annotation data of the document at a time. Use this interface to load annotation data progressively.
getAnnotsCount()	Get the count of loaded annotations in the comment panel.
setAnnotsCount(count)	Set the count number of the annotations in comment list.

gotoAnnotPosition(pageIndex,
annotName)

Go to a specific page by clicking an annotation in the comments panel.

5.1.14 Search

Users may want to search text for faster browsing in a document. WebPDF SDK provides APIs to search for all as well as previous/next instances. Please refer to the detailed code in “search.js”.

Example: Click the Search All button

```
$searchAllTextBtn.on("click", function () {
    WebPDF.ViewerInstance.searchAllText(_getSearchText(), 0);
});
// Search-previous button click event
$searchPreviousBtn.on("click", function () {
    WebPDF.ViewerInstance.searchText(_getSearchText(),
WebPDF.SearchDirection.Up);
});
// Search-next button click event
$searchNextBtn.on("click", function () {
    WebPDF.ViewerInstance.searchText(_getSearchText(),
WebPDF.SearchDirection.Down);
});
```

Common APIs of Search:

API Name	Description
SearchResult	Search Result object
count()	Get the counts of the search results
getContent()	Get the content of the search result
getPageIndex()	Get the page index of a search result
getRects()	Get the rects of the text from the index search

5.1.15 Watermark

WebPDF SDK supports two kinds of watermark settings. One is the global watermark setting in the admin page, which will take effect to all documents in the system. See [Set a watermark](#). The other one is watermark API, which defines the content of each watermark in each document. The two kinds of watermarks are separate and will not affect each other.

Watermark API provides two interfaces, “setWatermarkInfo()” and “getWatermarkInfo()”. The watermark data is saved in cookies in the default demo (invalid if the cookies are erased). WebPDF

Reader will process the watermark data from the cookies, and then apply it in the document with “setWatermarkInfo()” when opening a PDF document with watermarks. Please refer to the detailed code in “menuCreate.js”.

Example: Open a PDF with watermark setting

```
$("#createViewBtn").click(function(){
    saveWatermarkConfigs(setupWmkConfigs(getWmkConfigsFromPage()));
    WebPDF.ViewerInstance.setWatermarkInfo(getWatermarkConfigs());
    var fileUrl = getFileUrl();
    if( fileUrl != false){
        var openUrl = null;
        if(isRadioCheck($localFileBtn))
            openUrl = openLocalFile(baseUrl,fileUrl,afterOpenUrl);
        else
            openUrl = openFile(baseUrl,fileUrl,afterOpenUrl);
    }
});
```

5.2 WebPDF Reader on Mobile

This section will introduce how to use WebPDF SDK APIs on mobile sample. The following files include the main reference code used in this section.

```
File 1: ..\Foxit WebPDF\webapp\mobile (All the html files)
File 2:.. Foxit WebPDF\webapp\scripts\reader\control\mobile (All the
JavaScript files)
File 3: ..\Foxit WebPDF\webapp\styles\reader\mobile
File 4: FoxitWebPDF_APIREFERENCE (API file)
```

5.2.1 Open a PDF file on your own system

Once the [adapter plug-in](#) named com.foxit.webpdf.demo.HttpDocumentPlugin is created, the PDF files in the network can be opened by transferring the PDF URL to WebPDF Reader.

According to the “common.js”, modify the interface parameter to “com.foxit.webpdf.demo.HttpDocumentPlugin”, and then the PDF files in the corresponding system can be opened with the api/file/add interface.

Example: Open a PDF file on Mobile

```
var loadParams = fileUrl + "?user=" + user
    + "&readOnly=false&disablePrint=false&disableDownload=false";
var loadType = " com.foxit.webpdf.demo.HttpDocumentPlugin";
var data = {
    params: loadParams,
    type:loadType
};

// to avoid new window being blocked by browser
var newTab = window.open('about:blank');
$.ajax({
    url: "../api/file/add",
    type: "POST",
    data: data,
    async:false,
    success: function (data) {
        .....
    },
    error: function() {
        .....
    }
});
```

See [Common APIs of Viewer](#).

5.2.2 Annotation

See [Annotation on PC](#).

Please refer to the detailed code in “demo.html” and “readercontrol.js”.

5.2.3 Save Annotation

See [Save Annotation on PC](#).

Please refer to the detailed code in the “demo.html” and “readercontrol.js”.

Example: Click the Save button

```
/*
 * handle tap event of save button.
 */
$("#fwrn-main-btnSave").on("tap", function(event) {
    if (_isDocModified) {
        WebPDF.ViewerInstance.saveAnnots();
    }
});
```

5.2.4 Rotation

See [Rotation on PC](#).

Please refer to the detailed code in “demo.html” and “more.js”.

Example: Click the Rotate button

```
$("#rotateLeft").off("tap").on("tap", function () {  
    viewerInstance.rotate(WebPDF.ROTATE_LEFT);  
    event.stopPropagation();  
    event.preventDefault();  
    return false;  
});  
/*  
 * bind tap event on rotate right button.  
 */  
$("#rotateRight").off("tap").on("tap", function () {  
    viewerInstance.rotate(WebPDF.ROTATE_RIGHT);  
    event.stopPropagation();  
    event.preventDefault();  
    return false;  
});
```

5.2.5 Ink Signature

See [Ink Signature on PC](#).

Please refer to the detailed code in “inksign.html” and “inksign.js”.

5.2.6 Web PDF Form

See [Web PDF Form on PC](#)

Please refer to the detailed code in “demo.html” and “readercontrol.js”.

5.2.7 Bookmark

See [Bookmark on PC](#).

Please refer to the detailed code in “demo.html”, “readercontrol.js” and “bookmark.js”.

Example: Click the Bookmark button

```
// bind tap event for bookmark button. To show up bookmark panel.
$("#fwrn-main-btnBookmark").on("tap", function(event) {
    if (document.getElementById("fwrn-panel").style.display == "none")
    {
        $("#fwrn-panel").css("display", "block");
        _showChildNodes(WebPDF.ViewerInstance);
    }
    // hide rotate buttons
    $("#fwrn-menu-rotate").hide();

    event.stopPropagation();
    event.preventDefault();
});
```

See [Common APIs of Bookmark](#).

5.2.8 Thumbnail

See [Thumbnail on PC](#).

Please refer to the detailed code in “demo.html”, “readercontrol.js” and “thumbnail.js”.

5.2.9 Comment list

See [Thumbnail on PC](#).

Please refer to the detailed code in “demo.html”, “readercontrol.js” and “commentlist.js”

Example: Load more annotations

```
$('.fwrn-annot-edit .ok').off('tap').on('tap', function() {
    var $annotEdit = $('.fwrn-annot-edit');
    var pageIndex = $annotEdit.attr('page-index');
    var annotName = $annotEdit.attr('annot-name');
    var content = $annotEdit.find('.edit-
content').get(0).innerText;
    var annot = commentListPlugin.setAnnotContent(pageIndex,
annotName, content);
    updateAnnot(annot);
    $annotEdit.hide();
    return false;
});
```

5.2.10 Search

See [Search on PC](#).

Please refer to the detailed code in “demo.html”, “readercontrol.js” and “search.js”.

See [Common APIs of Search](#).

5.2.11 Watermark

See [Watermark on PC](#).

Please refer to the detailed code in “upload.html” and “upload.js”.

Example: Add watermarks in the DEMO page

```
$("#btnViewFile").click(function () {  
    var fileUrl =getFileUrl();  
    if (!isFileNameValid(fileUrl)) {  
        //alert("No PDF file found!");  
        alert(i18n.t("Demo.NotFound"));  
        return;  
    }  
    saveWatermarkConfigs(setupWmkConfigs(getWmkConfigsFromPage(false)));  
    // WebPDF.ViewerInstance.setWatermarkInfo(getWatermarkConfigs());  
    openFile(fileUrl);  
});
```

Example: Save watermark data in cookies

```
function saveWatermarkConfigs(wmkConfigs) {  
    var exdate = new Date()  
    exdate.setTime(exdate.getTime() + 30*24*60*60*1000);  
    document.cookie = "userWatermark=" + escape(JSON.stringify(wmkConfigs))  
        + ";expires=" + exdate.toGMTString() + ";path="/;
```

6 FAQ

1. Is there any file size limitation for WebPDF Reader?

Foxit WebPDF Reader supports PDF files under 200 MB.

2. Why can't I open the file with the previous URL?

When a user opens a PDF file within WebPDF Reader, the URL actually points to a database. So if the cache in the database of the PDF file has been cleared, the URL will not work again. In this case, the user should reopen the file through the file system to reactivate the cache.

3. Why is my WebPDF Reader display corrupted in my browser?

If you are using IE for viewing purposes, please check whether the browser is compatible. WebPDF Reader does not support the older IE versions like IE 6 or IE7, as the file display may be corrupted if used by these versions.

4. How can I check the log information of WebPDF SDK?

Please find the log information of WebPDF SDK in the log folder in the root of the installation directory. You can clear the log files when the folder is full.

5. **When I reset Database from default to MySQL, the message “Can’t connect to MySQL server.” pops up.**

If the message pops up, please make sure that all the parameters have been set correctly. The MySQL account has remote access permission.

6. **Why is the number of pages of the output file more than that of the original file in WebPDF Reader when I print the output file? How do you solve it?**

The print configurations are not the same for different browsers. Therefore, we suggest setting the correct layout, paper size, and margins before printing.

7. **Why I cannot open a PDF document with password even input a correct password?**

Please make sure to turn on the session function of web container when deployment WebPDF SDK since WebPDF Reader need to cache user session.

Support

Foxit Support:

<http://www.foxitsoftware.com/support/>

Sales Contact:

Phone: 1-866-680-3668

Email: sales@foxitsoftware.com

Support & General:

Phone: 1-866-MYFOXIT or 1-866-693-6948

Email: support@foxitsoftware.com