DEVELOPER GUIDE
Foxit® WebPDF Viewer
# Table of Contents

1  Introduction to Foxit WebPDF Viewer ............................................................................................................. 1
   1.1 Why Foxit WebPDF Viewer is your choice ................................................................................................. 1
   1.2 Audience and Scope ......................................................................................................................................... 2
   1.3 Your Web Application .................................................................................................................................... 2
   1.4 Features .......................................................................................................................................................... 2
   1.5 Evaluation ...................................................................................................................................................... 3
   1.6 License .......................................................................................................................................................... 3

2  REST API ............................................................................................................................................................ 4
   2.1 Sample REST API (Java) Project .................................................................................................................... 4
      2.1.1 How to use the Sample Project ............................................................................................................. 5
         2.1.1.1 Start the Java project ...................................................................................................................... 5
         2.1.1.2 Configure of apiConfig.js ............................................................................................................. 6
         2.1.1.3 Verify the REST API docked to WebPDF Viewer ........................................................................... 6
      2.2 Introduction of apiConfig.js ..................................................................................................................... 7
          General response format ............................................................................................................................... 7
          Error code list ................................................................................................................................................. 8
   2.3 Sample REST API (Java) details .................................................................................................................... 8
      2.3.1 Document Interface ................................................................................................................................... 8
         2.3.1.1 Get document ....................................................................................................................................... 8
         2.3.1.2 Export document ............................................................................................................................ 9
         2.3.1.3 Get readable page count ................................................................................................................ 9
      2.3.2 Form interface ......................................................................................................................................... 10
         2.3.2.1 Import form data (XML format) into WebPDF ........................................................................... 10
         2.3.2.2 Export form data (XML format) to a third-party content management system ... 10
2.3.3 Signature interface .................................................................................................................. 11
  2.3.3.1 getSignature ..................................................................................................................... 11
  2.3.3.2 getContents ..................................................................................................................... 12
  2.3.3.3 getSignedDocument ........................................................................................................ 12
  2.3.3.4 exportSignedDocument .................................................................................................... 13
2.3.4 User interface .......................................................................................................................... 14
  2.3.4.1 getUserInfo .................................................................................................................... 14
  2.3.4.2 getUserPermission .......................................................................................................... 15
2.3.5 Print control interface .............................................................................................................. 16
  2.3.5.1 getPrintCount ................................................................................................................ 16
2.3.6 Text copy control interface ..................................................................................................... 16
  2.3.6.1 getCopyCount ................................................................................................................ 16
2.4 Call REST API in the front-end of WebPDF Viewer ..................................................................... 17
  2.4.1 Document ............................................................................................................................ 18
    2.4.1.1 Export document ......................................................................................................... 18
    2.4.1.2 Get document .............................................................................................................. 18
    2.4.1.3 Get Readable Page counts .......................................................................................... 19
  2.4.2 Form data import/export ....................................................................................................... 19
    2.4.2.1 Import form data (XML format) into WebPDF ............................................................ 19
    2.4.2.2 Export form data (XML format) into 3rd content system ........................................... 20
  2.4.3 Signature ............................................................................................................................. 20
  2.4.4 User Interface ....................................................................................................................... 21
    2.4.4.1 getUserInfo ................................................................................................................ 21
    2.4.4.2 getUserPermission ..................................................................................................... 22
  2.4.5 Print Control Interface ......................................................................................................... 23
1 Introduction to Foxit WebPDF Viewer

By using Foxit WebPDF Viewer, developers can deploy and customize a WebPDF Viewer that supports viewing PDF documents within a web browser. Integrating a WebPDF Viewer into a zero-footprint web app allows end users to view PDF documents on desktop and mobile devices without installing anything.

1.1 Why Foxit WebPDF Viewer is your choice

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

**Fully customizable**
Developers can easily design a unique style for their WebPDF Viewer interface, and make it consistent to their web application.

**Easy to integrate**
Developers can easily create a REST API to get and export document methods in the web application, which can be integrated with WebPDF Viewer.

**Integrated server components**
The installation package includes the web container and database. It also provides a way to quickly support cluster environments.

**Standard and consistent annotation data**
The annotations in WebPDF Viewer are consistent when viewing and editing in other applications.

**Full control of documents**
The original document can be stored and viewed server-side without being downloaded by user to the document owner retains full control. It also supports importing user permissions from a third-party system to set different functions for different users.

**Powered by Foxit’s high fidelity rendering PDF engine**
The core technology of WebPDF Viewer 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 fast and consistent in all environment.

In addition, Foxit's products are offered with the full support of our dedicated support engineers if support and maintenance options are purchased. Updates are released on a regular basis. Foxit WebPDF Viewer 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 developers who need to integrate Foxit WebPDF Viewer into their web applications. It covers the REST API definition and usage for integration as well as the front-end API for customization. Also Refer to the Deployment Guide before integration development begins.

1.3 Your Web Application

Foxit WebPDF Viewer 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 Viewer.

1.4 Features

Foxit WebPDF Viewer provides the most common PDF viewing features and allows developers to incorporate powerful PDF technology to their applications like viewing, text searching, printing, form filling 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.

<table>
<thead>
<tr>
<th>Features</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PDF View</td>
<td>Go to page, Zoom in/out, Fit width, Bookmark accessing, Thumbnail, Print, Rotate</td>
</tr>
<tr>
<td>PDF Text</td>
<td>Text selection and copy, Text search</td>
</tr>
<tr>
<td>Annotation</td>
<td>Display all existing annotations</td>
</tr>
<tr>
<td></td>
<td>15 annotation tools for editing (Highlight, Underline, Squiggly, Strikeout, Note, Pencil, Typewriter, Callout, Rectangle, Oval, Line, Arrow, Polyline, Polygon, Stamp)</td>
</tr>
<tr>
<td>Digital Signature</td>
<td>Provide REST API definition to developers to integrate third-party digital signatures, including images and ink-like signatures (Not supported in JR engine)</td>
</tr>
<tr>
<td>Acroform</td>
<td>Form filling, export/import PDF form data</td>
</tr>
<tr>
<td>Security</td>
<td>Supports password protection, watermarks</td>
</tr>
<tr>
<td>Rendering engine</td>
<td>Two engines for rendering.</td>
</tr>
</tbody>
</table>
SR, server rendering, convert PDF file into browser-identifiable files (HTML files) on the server.
JR, JavaScript rendering, rendering PDF with JavaScript in local browser.

1.5 Evaluation

Foxit WebPDF Viewer allows users to download the trial version to evaluate the SDK. The trial version is the same as the 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 Viewer.

1.6 License

Developers should purchase licenses to use Foxit WebPDF Viewer in their solutions. Licenses grant users permissions to release their applications based on WebPDF Viewer. However, users are prohibited to distribute any documents, sample codes, or source codes in the released packages of Foxit WebPDF Viewer to any third party without the permission from Foxit Software Incorporated.
2 REST API

WebPDF Viewer define a series of interfaces (name, parameter and return value) for third-party content management systems to follow to create the REST APIs for integration. The developer of the content management system can implement the REST APIs via several languages including PHP, .NET, Java etc. A sample implementation project in Java is provided.

WebPDF Viewer will send requests (with parameters) to the REST APIs, and then continue the function in WebPDF with the return objects (e.g. return a PDF object). When developers start to use the REST API, they need to remove the comments of the corresponding API in apiConfig.js.

2.1 Sample REST API (Java) Project

Foxit provides a sample REST API implementation project in Java for reference. You can find the sample project in ..foxitsoftware/webpdf/sample. There are 10 REST APIs in the sample project, please follow the below directions to run the project.

a. getDocument()
b. exportDocument()
c. getReadablePages()
d. getSignature()
e. getContents()
f. exportSignedDocument()
g. getSignedDocument()
h. importForm()
i. exportForm()
j. getUserInfo()
k. getUserPermission()
2.1.1.1 How to use the Sample Project

Start the webpdf-extension-sample project. Please note that a web server must be prepared to run up the project. (e.g. Tomcat)

Try to open localhost:8083/webpdf-extension-sample/systemuser, it will be shown as below if the project is running successfully.(change the web service port according to you environment)

Note: To support Chinese files, it requires to set the encoded to utf-8 before starting the project. E.g., add URIEncoding="UTF-8" in server.xml of Tomcat.
2.1.1.2 Configure of apiConfig.js

`apiConfig.js` is a configuration file to determine if the WebPDF uses REST API flow or not. All the REST APIs configuration are commented in the front-end of WebPDF Viewer by default. Remove the comments of the corresponding API in apiConfig.js to start using the REST API. Take “getUserInfo” API as an example:

```javascript
"getUserInfo": function(){
  return{
    "type": "restful",
    "url": "http://localhost:8083/webpdf-extension-sample/systemuser",
    "method": "get",
    "heads": {
      "defaultuser": getCookie('default_user')
    },
    "params":{}
  },
}
```

2.1.1.3 Verify the REST API docked to WebPDF Viewer

To verify the REST API in the Java sample project has been docked to WebPDF successfully, follow these steps.

Take the “getUserInfo” API as an example, a developer can use the corresponding JavaScript API of WebPDF “WebPDF.ViewerInstance.updateCurrentUserInfo()” to send a request to the REST API, and then verify if the return value is correct. The front-end WebPDF example code is in `viewer/webapp/scripts/control/pc/demo.js`:

```javascript
WebPDF.ready(docViewerId, optionsParams).then(function(data){
  ... 
  return WebPDF.ViewerInstance.updateCurrentUserInfo();
}).then(function(data) {
  WebPDF.ViewerInstance.on(WebPDF.EventList.DOCUMENT_LOADED, function(event, data){
    console.log("userId:"+WebPDF.AccountInstance.getUserId());
    console.log("userName:"+WebPDF.AccountInstance.getUserName());
    console.log("accesstoken:"+WebPDF.AccountInstance.getAccessToken());
  })
});
```
If the return value appears as below, then the REST API is running successfully.

The interface must be coordinated with REST API getUserInfo() for usage.

2.2 Introduction of apiConfig.js

The apiConfig.js file is located at ..\webpdf\viewer\webapp\scripts\config\apiConfig.js. It is a configuration file to determine if the WebPDF uses REST API flow or not. All the REST APIs are commented by default.

```javascript
{
  type: "restful" // restful api tag
  url: // The url of the REST API
  heads: // The head information when request
  params: // The parameters when request
}
```

General response format

Below is the general return format of each REST API:

```javascript
{
  ret:... // the error code of return value. 0 refers to a success.
  Message:... // the detailed message information of the return result.
  data:{// the data of the return result.
}
```
Error code list

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>SERVER ERROR</td>
<td>Interface exception</td>
</tr>
<tr>
<td>0</td>
<td>SUCCESS</td>
<td>Return success</td>
</tr>
<tr>
<td>10001</td>
<td>FILE_IS_NOT_EXIST</td>
<td>The file does not exist.</td>
</tr>
<tr>
<td>10002</td>
<td>SINATURE_CERT_IS_NOT_EXIST</td>
<td>The certification does not exist.</td>
</tr>
<tr>
<td>10003</td>
<td>PARSE_SINATURE_CERT_ERROR</td>
<td>Cannot parse the certification.</td>
</tr>
<tr>
<td>10004</td>
<td>SIGNATURE_FILTER_NOT_SUPPORT</td>
<td>Cannot support the signature filter.</td>
</tr>
<tr>
<td>10005</td>
<td>SIGNATURE_SUBFILTER_NOT_SUPPORT</td>
<td>Cannot support the signature subfilter.</td>
</tr>
</tbody>
</table>

2.3 Sample REST API (Java) details

This section takes the “webpdf-extension-sample” project as an example to learn about all of the REST API interfaces.

2.3.1 Document Interface

Usually a developer will use WebPDF Viewer as an online PDF viewing solution. So, WebPDF Viewer needs to have a way to get documents from third-party content management system and/or export PDFs into a system.

2.3.1.1 Get document

Get a PDF document from a third-party content management system

```javascript
/*
   "getDocument": function(){
       return {
           "type": "restful",
           "method": "get",
           "heads": { },
           "params": { }
       }
   }
*/
```

Response:

Response body: the binary stream of the PDF document

Response Headers:
2.3.1.2 Export document

Export PDF document and annotation data from WebPDF to a third-party content management system.

```javascript
/*
   "exportDocument": function(){
     return {
       "type": "restful",
       "method": "post",
       "heads": {},
       "params": {}
     }
   }
*/
```

Response

Response body:

In common status, return

```json
{"ret":"0","message":"SUCCESS"}
```

Return error code defined by API if error exists.

2.3.1.3 Get readable page count

Get the readable page count of current user for current document.

```javascript
"getReadablePages": function(){
  return {
    "type": "restful",
    "method": "get",
    "heads": {},
    "params": {}
  };*/
```

Response:

Response body

In common status, return
Return error code which defined by API if an error exists.

### 2.3.2 Form interface

When develop an online form filling solution, you probably want to collect all the filled data from responders to a database for centralized management. WebPDF Viewer supports importing and exporting form data from/to a remote server.

#### 2.3.2.1 Import form data (XML format) into WebPDF

Import form data (XML format) from a remote server (url).

```javascript
/*
   "importForm": function(){
       return {
           "type": "restful",
           "method": "get",
           "heads": {},
           "params": {}
       }
   }
*/
```

Response:

Response body

In common status, return

```json
{"ret":"0","message":"SUCCESS"}
```

Return error code which defined by API if error exists.

#### 2.3.2.2 Export form data (XML format) to a third-party content management system

Export form data (XML format) to a remote server (url).
2.3.3 Signature interface

WebPDF Viewer supports the integration of digital certificates, allowing end users to sign PDF document. The APIs define how WebPDF retrieves the needed information (signature appearance image, certification) from the system.

2.3.3.1 getSignature

Get certificate information of the signature

```javascript
/*
getSignature: function() {
    return {
        "type": "restful",
        "url": "http://localhost:8083/webpdf-extension-sample/signature",
        "method": "get",
        "heads": {},
        "params": {}
    }
}
*/
```

Response

Response body:

In common status, return

```json
{"ret":0,"message":"SUCCESS"}
```

Return error code which defined by API if an error exists.
Return error code which defined by API if an error exists.

2.3.3.2 getContents
Get the content of signature.

```javascript
/*
 * getContents: function() {
 *     return {
 *         "type": "restful",
 *         "url": "http://localhost:8083/webpdf-extension-sample/signature/content",
 *         "method": "get",
 *         "heads":{},
 *         "params":{}
 *     }
 * }
 */
```

Response:
Response body:
In common status, return

```json
{"ret":0,"message":"SUCCESS","data":{ "content":"..." // Hex string of signature content }}
```

Return error code which defined by API if an error exists.

2.3.3.3 getSignedDocument
Get the signed PDF document
Response

Response body:

In common status, return the binary stream of the PDF

2.3.3.4 exportSignedDocument

Export the signed PDF document

_RESPONSE_

Return _error code_ which defined by API if an error exists.
2.3.4 User interface

Integrating user accounts of a third-party content management system into WebPDF Viewer will allow users to save user specific data from the WebPDF Viewer, such as annotations, ink signature templates, form data, etc. It also supports importing user permissions from document management systems - WebPDF Viewer can then offer different functionality depending on each user’s permission settings.

2.3.4.1 getUserInfo

Get information about the user, such as user id, user name etc.

```javascript
/*
 * getUserInfo: function(){
 *   return{
 *     "type": "restful",
 *     "url": "http://localhost:8083/webpdf-extension-sample/systemuser",
 *     "method": "post",
 *     "heads": {
 *       "defaultuser": getCookie('default_user')
 *     },
 *     "params":{}
 *   }
 * }
 */
```

If developer does not want to set user information via web services, they also can do it front-end at ..\webpdf\viewer\webapp\scripts\control\pc\demo.js

```javascript
WebPDF.ready(docViewerId, optionsParams).then(function(data) {
  var viewToolBar = ViewToolBar.getViewToolBar(WebPDF.optionParameters);
  viewToolBar.init();
  var param = {
    accessToken:"testAccessToken",
    userName:"testUserName",
    userId:"testUserId"
  };
  return WebPDF.ViewerInstance.updateCurrentUserInfo(param);
})
```

Response:

Response body:

In common status, return
Return error code which defined by API if an error exists.

### 2.3.4.2 getUserPermission

Get the permission of current user for current document

```javascript
/*
Get the permission of current user for current document
*/
"getUserPermission": function(){
    return{
        "type": "restful",
        "url": "http://localhost:8083/webpdf-extension-sample/systemuser/permission",
        "method": "post",
        "heads": {},
        "params":{}
    }
}
```

**Response**

Response body:

In common status, return

```json
{"ret":"0 ",
"message":"SUCCESS",
"data":{
    "permission":-1,  // -1 for all permission
0 for no permissions including view
1 for view permission,
2 for login permission,
4 for extract content permission,
8 for print permission,
16 for download permission,
32 for comment permission,
64 for signature permission,
128 for form filling permission.
}}
```

Return error code which defined by API if error exists.
2.3.5 Print control interface

In some cases, document owners want to control the print page count that to limit a no-payment usage.

2.3.5.1 getPrintCount

Get the printable page range of current user for current document, and the remaining printable page count for current user of this document.

```
"getPrintCount": function ()
{
    /* return{
        "type": "restful",
        "url": "http://localhost:8083/webpdf-extension-sample/api/print/count",
        "method": "get",
        "heads": {},
        "params":{}
    }*/
}
```

Response

Response body:

In common status, return

```
{
    ret:0,
    message:"success",
    data:{
        printRange:1,2,3 //The printable page range at this time, can be set 1,2-5
        remainingPrintCount:3 //The reminding printable page count
    }
}
```

Return error code which defined by API if error exists.

2.3.6 Text copy control interface

In some case, document owners want to control the text copy count that to limit a no-payment usage.

2.3.6.1 getCopyCount

Get the text count that allow to copy and the remaining count.
"getPrintCount": function () {
    /* return{
        "type": "restful",
        "url": "http://localhost:8083/webpdf-extension-sample/api/print/count",
        "method": "get",
        "heads": {},
        "params":{}
    }*/
}

Response

Response body:

In common status, return

```json
{
    ret:0,
    message:"success",
    data:{
        copyCount:3 // the text count that allow to copy at this time
        remainingCount:0 // the remaining count
    }
}
```

Return error code which defined by API if error exists.

2.4 Call REST API in the front-end of WebPDF Viewer

This chapter details how to use these REST APIs from the front-end of the WebPDF Viewer. You can verify usage from the log information.

First, modify the logger level to “INFO” in {webapp}/webpdf-extension-sample/WEB-INF/classes/log4j.properties, and then restart the web project.
Check the log file at {webapp}/webpdfd-extension-sample/log/webpdf.log, and you should find the record as below:

```
INFO localhost-startStop-1 (DocumentController.java:49) - the EXPORT_HOME is: /D:/webpdf_home_webpdf/.metadata/.p
INFO localhost-startStop-1 (FormController.java:49) - the EXPORT_HOME is: /D:/webpdf_home_webpdf/.metadata/.plug
INFO http-bio-8083-exec-9 (SystemUserController.java:74) - get systemuser info, params: defautuser: 7a16d7dca800
INFO http-bio-8083-exec-5 (SystemUserController.java:88) - get systemuser permission, params: accesstoken: ACCESS
```

2.4.1 Document

2.4.1.1 Export document

There is a front-end API to call the REST API exportDocument():

```javascript
WebPDF.ViewerInstance.exportDocumentToUrl (exportFileName, params, successCallback, failCallback)
```

Front-end Example:

```javascript
$('#btn ExportPDFRemote').click(function(){
    WebPDF.ViewerInstance.exportDocumentToUrl("exportname.pdf",null,function(){alert("success")), function(){alert("fail")});
});
```

Verification:

Remove the comments of exportDocument(). Click the test button, if a PDF file be generated under the specific folder in the web service and the log appears as below, the REST API has been called successfully:

```
INFO localhost-startStop-1 (DocumentController.java:49) - the EXPORT_HOME is: /D:/webpdf_home_webpdf/.metadata/.p
INFO localhost-startStop-1 (FormController.java:49) - the EXPORT_HOME is: /D:/webpdf_home_webpdf/.metadata/.plug
INFO http-bio-8083-exec-9 (SystemUserController.java:74) - get systemuser info, params: defautuser: 7a16d7dca800
INFO http-bio-8083-exec-5 (SystemUserController.java:88) - get systemuser permission, params: accesstoken: ACCESS
```

2.4.1.2 Get document

There is no front-end API to call the REST API getDocument() directly, developer can use openFileByUri() at ..\webpdf\viewer\webapp\scripts\control\pc\demo.js

```javascript
WebPDF.ViewerInstance.openFileByUri()
```

Verification:
Remove the comments of getDocument() and reopen a PDF file. If the file is opened successfully and the log appears as below, the REST API has been called successfully.

2.4.1.3  Get Readable Page counts

There is no front-end API to call the REST API getReadablePages() directly. When front-end of viewer request image, text and annotation, WebPDF will call getReadablePages(), and trigger event WebPDF.EventList.PAGE_SHOW_LIMIT.

Verification:

Remove the comments of getReadablePages() and reopen a PDF file. If the file is opened successfully and the log (when request image) appears as below, the REST API has been called successfully.

At the same time, WebPDF.EventList.PAGE_SHOW_LIMIT will be triggered, and it can be checked from the result of data:

```javascript
viewer.on(WebPDF.EventList.PAGE_SHOW_LIMIT, function(event, data) {
    alert('pageIndex:'+ data.pageIndex);
});
```

2.4.2 Form data import/export

2.4.2.1 Import form data (XML format) into WebPDF

There is a front-end API to call the REST API importForm():

```javascript
WebPDF.ViewerInstance.getPluginByName(WebPDF.FormPluginName).importXMLFromUrl(params, successCallback, failCallback)
```

Front-end Example:

```javascript
$('#btnImportFormRemote').click(function(){
  WebPDF.ViewerInstance.getPluginByName(WebPDF.FormPluginName).importXMLFromUrl(null, function(){alert("success"), function(){alert("fail")});
});
```
Verification:

Remove the comments of importForm(). Click the test button, if the value of form field be filled out with the imported XML file and the log appears as below, the REST API has been called successfully:

```java
INFO http-bio-5083-exec-3 [FormController.java:60] - export form params (Method: POST), fileName:export.xml
INFO http-bio-5083-exec-2 [FormController.java:132] - import form
```

Note that importForm and exportForm interfaces in FormController.java of webpdf-extension-sample project uses (project)/WEB-INF/classes/export/ as default path. Developer can change the file path accordingly.

### 2.4.2.2 Export form data (XML format) into 3rd content system

There is a front-end API to call the REST API exportForm():

```javascript
WebPDF.ViewerInstance.getPluginByName(WebPDF.FormPluginName).exportXMLToUrl(filename, params, successCallback, failCallback)
```

Front-end Example:

```javascript
$("#btnImportFormRemote").click(function(){
    WebPDF.ViewerInstance.getPluginByName(WebPDF.FormPluginName).exportXMLToUrl("export.xml", null, function() {
        alert("success")
    }, function() {
        alert("fail")
    });
});
```

Verification:

Remove the comments of exportForm(). Click the test button, if a XML file be generated under the specific folder in the web service and the log appears as below, the REST API has been called successfully:

```java
INFO http-bio-5083-exec-3 [FormController.java:60] - export form params (Method: POST), fileName:export.xml
```

### 2.4.3 Signature

There is no front-end API to call the REST API of signature directly.

Verification:
a. Remove the comments of `getSignature()`, `getContents()`, `exportDocument()` and `getSignedDocument()`. If normal signature with digital certification function works, and the log appears as below, the REST API has been called successfully:

```
Controller.java:60) - export form params (Method: POST), fileName=export.xml
Controller.java:156) - export form
trueController.java:130) - get signature content, , params : hash (UOaE7v64C2Gli37nizable/VE8YQp, certSample Software Incorporated.pdf, filterAdobe.AVPL, e
trueController.java:79) - export signature document (Methods: POST)
trueController.java:89) - have signature document, params: doesilj : (forceField, signed.pdf)
trueController.java:84) - get signature document, params: accessDomain (accessDomain, userId : ne7e7e7ed6e56co26c0662c57cl, ui ?functionId_signed.pdf,
```

To support Chinese signature, it requires to set the encoded to utf-8. E.g., add `URIEncoding="UTF-8"` in `server.xml`.

```
<Connector port="8083" protocol="HTTP/1.1"
   connectionTimeout="20000"
   redirectPort="8443"  URIEncoding="UTF-8"/>
```

### 2.4.4 User Interface

#### 2.4.4.1 getUserInfo

There is a front-end API to call the REST API `getUserInfo()`:

```
WebPDF.ViewerInstance.updateCurrentUserInfo(param)
```

In index.html page integrated with WebPDF Viewer, refer to the sample code in `viewer/webapp/scripts/control/pc/demo.js`.

```
WebPDF.ready(docViewerId, optionsParams).then(function(data) {
    ...
    return WebPDF.ViewerInstance.updateCurrentUserInfo();
    })
```

Verification:

Remove the comments of `getUserInfo()`. Add the following code into `webapp/scripts/control/pc/demo.js`, and then verify the output information in the browser console.

```
WebPDF.ViewerInstance.on(WebPDF.EventList.DOCUMENT_LOADED, function(event, data) {
    console.log("userId:"+WebPDF.AccountInstance.getUserId());
    console.log("userName:"+WebPDF.AccountInstance.getUserAccount());
    console.log("accesstoken:"+WebPDF.AccountInstance.getAccessToken());
});
```
Check the information in the browser console. The REST API has been called successfully if the output information and the log appears as follows.

![Console log example](image)

2.4.4.2 `getUserPermission`

Verification:

Remove the comments of `getUserPermission`. Add following code into `webapp/scripts/control/pc/demo.js`,

```javascript
WebPDF.ViewerInstance.on(WebPDF.EventList.DOCUMENT_LOADED, function(event, data) {
    console.log("userPermission:" + WebPDF.ViewerInstance.getUserPermission());
});
```

Check the information in the browser console. The REST API has been called successfully if the output information and the log appears as follows.
2.4.5 Print Control Interface

2.4.5.1 getPrintCount

There is no front-end API to call the REST API getPrintCount() directly. When user set a print range in the print dialog, and then start to print, WebPDF will call getPrintCount(), and trigger event WebPDF.EventList.PAGEs_Prepare_PRINT.

Verification:

Remove the comments of getPrintCount() and try to print. If print successfully and the log appears as below, the REST API has been called successfully.

At the same time, WebPDF.EventList.PAGES_PREPARE_PRINT will be triggered, and it can be checked from the result data:
2.4.6 Text copy control interface

2.4.6.1 getCopyCount
There is no front-end API to call the REST API getCopyCount() directly. When user copy the text, WebPDF will call getCopyCount().

Verification:
Remove the comments of getCopyCount() and try to copy some text. If copy successfully and the log appears as below, the REST API has been called successfully.

At the same time, it will trigger the callback function as following:

```javascript
WebPDF.ViewerInstance.on(WebPDF.EventList.DOCUMENT_LOADED, function(event, data) {
  var textSelectionHandler = WebPDF.ViewerInstance.getToolHandlerByName(WebPDF.Tools.TOOL_NAME_SELECTTTT);
  var callback = function(copyInfo, remainingCount){
    console.log("current selected text: " + copyInfo+"remainingCount:"+remainingCount);
  }
  textSelectionHandler.addListener('Copy',callback);
});
```

3 Front-end API

Foxit WebPDF Viewer provides a series of APIs for developers to create their own WebPDF Viewer.
Create a HTML page for the WebPDF Viewer interface and then create a DIV tag as the container for the main viewer object in the HTML.

Create a HTML page named “myviewer” (choose a name you like), and place it in the installation directory at ..\webpdf\viewer\webapp\pc

Create a <div> tag in the HTML <body> section and reserve “docViewer” as the ID. This will be the container for the WebPDF Viewer.

```html
<div id="docViewer" style="background:#ABABAB;"></div>
```

Add a style for the viewer element in HTML <head> section so that takes up a reasonably sized area.
Add the css link in the `<head>` section

```html
<link rel="stylesheet" type="text/css" href="../styles/reader/pc/webpdf.frontend.mini.css" />
<link rel="stylesheet" type="text/css" href="../styles/reader/webpdf.mini.css"/>
```

Add the following script after the HTML body section closes (`</body>`) to create a new Viewer instance

```javascript
var docViewerId = 'docViewer';
$(document).ready(function() {
  var optionsParams = {
    language: getLanguage(),
    customizedUrl: getBaseUrl(),
    baseUrl: getBaseUrl()
  };
  WebPDF.ready(docViewerId,optionsParams).then(function(data) {
    return WebPDF.ViewerInstance.updateCurrentUserInfo();
  }).then(function(data) {
    var openFileParams = {
      url: getBaseUrl() + 'docs/sample/butterfiles.pdf'
    };
    WebPDF.ViewerInstance.openFileByUri(openFileParams);
  });
  $(window).resize(function() {
    if(WebPDF == null || WebPDF.ViewerInstance == null){
      return;
    }
    var $docViewer = $('div#docViewer');
    var viewWidth = $docViewer[0].offsetWidth + $docViewer[0].offsetLeft + $(window).width();
    var viewHeight = $docViewer[0].offsetHeight + $(window).height();
    WebPDF.ViewerInstance.updateLayout(viewWidth, viewHeight);
  });
});
```

3.1 WebPDF Viewer on PC

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

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

| File 1: ..\webpdf\viewer\webapp\pc\index.html |
| File 2: ..\webpdf\viewer\webapp\scripts\control\common |
| File 3: ..\webpdf\viewer\webapp\scripts \control\pc (All the JavaScript files of PC view) |
| File 4: ..\webpdf\viewer\webapp\styles\reader\pc (css files of pc view) |
| File 5: FoxitWebPDF_APIREFERENCE (API file) |

3.1.1 Open a PDF file on your own system

Once the `getDocument` REST API been integrated, the PDF files in the network can be opened by transferring the PDF URL to WebPDF Viewer via `openFile()`.

Please refer to the detailed code in ..\webpdf\viewer\webapp\scripts\control\pc\index.js.

Common APIs of Viewer:

<table>
<thead>
<tr>
<th>API Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>closeFile()</td>
<td>Close current PDF file</td>
</tr>
<tr>
<td>exportDocument()</td>
<td>Export documents to local browsers</td>
</tr>
<tr>
<td>exportDocumentToUrl()</td>
<td>Export current document to a remote server</td>
</tr>
<tr>
<td>exportAnnotsToFDF()</td>
<td>Export annotations of the document into a FDF file</td>
</tr>
<tr>
<td>exportAnnotsToXFDF()</td>
<td>Export annotations of current document into a XFDF file</td>
</tr>
<tr>
<td>exportAnnotsToXFDFStream()</td>
<td>Export annotations of current document into a XFDF stream</td>
</tr>
<tr>
<td>exportDocumentStream()</td>
<td>Extract document to stream</td>
</tr>
<tr>
<td>Fire()</td>
<td>Fires an event with the given name and data</td>
</tr>
<tr>
<td>focus()</td>
<td>Focus the viewport so it can be natively scrolled with the keyboard</td>
</tr>
<tr>
<td>generateDocId()</td>
<td>Generate assert ID of current PDF document. Before open a PDF file, the viewer needs to load the PDF content by the 3rd document loader.</td>
</tr>
<tr>
<td>getCurToolHandlerName()</td>
<td>Get the current tool handler name.</td>
</tr>
<tr>
<td>getCurZoomLevel()</td>
<td>Get the current zoom level.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>getFileID()</td>
<td>Get the ID of current open document</td>
</tr>
<tr>
<td>getFileName()</td>
<td>Get the file name of the current PDF document.</td>
</tr>
<tr>
<td>getOptions()</td>
<td>Get the configuration options.</td>
</tr>
<tr>
<td>getPageAnnots()</td>
<td>Get the annotation data of specific page in json format</td>
</tr>
<tr>
<td>getPageCount()</td>
<td>Get the page counts of current document.</td>
</tr>
<tr>
<td>getPluginByName()</td>
<td>Get an instance of plug-in object according to its name</td>
</tr>
<tr>
<td>getThumbnail()</td>
<td>Get the URL of page thumbnail by the specified page index.</td>
</tr>
<tr>
<td>getToolHandlerByName()</td>
<td>Get the instance of tool handler according to its name</td>
</tr>
<tr>
<td>getUserPermission()</td>
<td>Get user permission.</td>
</tr>
<tr>
<td>getViewMode()</td>
<td>Get current view mode.</td>
</tr>
<tr>
<td>getZoomLevels()</td>
<td>Get the list of available zoom levels</td>
</tr>
<tr>
<td>gotoNextPage()</td>
<td>Go to the next page.</td>
</tr>
<tr>
<td>gotoPage()</td>
<td>Jump to a page</td>
</tr>
<tr>
<td>gotoPageByDestination()</td>
<td>Jump to a page by the specified page index and the offset to be scrolled to</td>
</tr>
<tr>
<td>gotoPrevPage()</td>
<td>Go to the previous page</td>
</tr>
<tr>
<td>hasForm()</td>
<td>Detect whether the document has form fields</td>
</tr>
<tr>
<td>HideAllAnnots()</td>
<td>Hide all annotations</td>
</tr>
<tr>
<td>hideAnnots()</td>
<td>Hide annotations</td>
</tr>
<tr>
<td>highlightText()</td>
<td>Highlight an area by rectangle</td>
</tr>
<tr>
<td>importAnnotsFromFDF()</td>
<td>Import annotation to current document from FDF</td>
</tr>
<tr>
<td>importAnnotsFromXFDF()</td>
<td>Import annotations to current document from XFDF</td>
</tr>
<tr>
<td>isDocModified()</td>
<td>Check whether current document has been modified</td>
</tr>
<tr>
<td>isFitWidth()</td>
<td>Check whether the current zoom level is WebPDF.ZOOM_FIT_WIDTH</td>
</tr>
<tr>
<td>isJREngineReady()</td>
<td>Check whether JavaScript engine is ready</td>
</tr>
<tr>
<td>ifOffline()</td>
<td>Check whether the WebPDF Viewer is in Offline Mode.</td>
</tr>
<tr>
<td>load()</td>
<td>Load current document assets</td>
</tr>
<tr>
<td>off()</td>
<td>Removes an event handler from a given event If the handler is not provided; remove all handlers of the given type</td>
</tr>
<tr>
<td>on()</td>
<td>Adds a new event handler for a particular type of event</td>
</tr>
<tr>
<td>openFileByStream()</td>
<td>Open a PDF file from file stream</td>
</tr>
<tr>
<td>openFileByUri()</td>
<td>Open PDF file by url.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>openLastOpenFile()</td>
<td>Open the PDF which Viewer opened last time. This interface is valid under the Offline Mode only at present.</td>
</tr>
<tr>
<td>print()</td>
<td>Open the Print dialog</td>
</tr>
<tr>
<td>rotate()</td>
<td>Rotate the page view.</td>
</tr>
<tr>
<td>save()</td>
<td>Save user data of current PDF to server</td>
</tr>
<tr>
<td>searchAllText()</td>
<td>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</td>
</tr>
<tr>
<td>searchText()</td>
<td>Search text</td>
</tr>
<tr>
<td>setCurrentToolByName()</td>
<td>Set the current tool by its name</td>
</tr>
<tr>
<td>setFullScreenFlag</td>
<td>Set the flag to indicate current viewer whether show in full screen mode or not</td>
</tr>
<tr>
<td>setLayoutShowMode()</td>
<td>Change the page layout mode of the viewer</td>
</tr>
<tr>
<td>showAllAnnots()</td>
<td>Show all annotations</td>
</tr>
<tr>
<td>showAnnots()</td>
<td>Show annotations</td>
</tr>
<tr>
<td>showFullPage()</td>
<td>Set to display the specified page in full screen mode.</td>
</tr>
<tr>
<td>updateUserInfo()</td>
<td>Update current login user information.</td>
</tr>
<tr>
<td>updateLayout()</td>
<td>Update the layout of current viewer instance</td>
</tr>
<tr>
<td>updateUUID()</td>
<td>Update current UUID after user login.</td>
</tr>
<tr>
<td>zoomTo()</td>
<td>Zoom to the given value</td>
</tr>
</tbody>
</table>

### 3.1.2 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 Viewer supports annotation types including note, underline, highlight, typewriter, pencil, line, arrow, oval and rectangle. 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_HIGHTLIGHT)
```
After editing annotations in the document, users should save the changes. Call the “ViewerInstance.save()” interface which is defined in the API to save the annotations as well as all user data. Please refer to the detailed code in “toolbar.js”.

**Example: Click the Save button**
```javascript
$("#btnSave").off('click').click(function(){
    WebPDF.ViewerInstance.save();
});
```

### 3.1.3 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**
```javascript
$("#btnLeftRotate").off("click").click(function(){
    WebPDF.ViewerInstance.rotate(WebPDF.ROTATE_LEFT);
});
```

### 3.1.4 Download
Users may want to download PDF documents to a local disk for further use in WebPDF Viewer. 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**
```javascript
$("#btnExportPDF").off("click").click(function(){
    WebPDF.ViewerInstance.exportDocument(null);
    return false;
});
```

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

**Example: Click the Print button**
```javascript
```
3.1.6 Account

The developer may want to get the account information of the current user for further usage. Front-end API provide series interfaces associated to the account. In the demo, user account is generated randomly. Developer can use REST API to seamlessly integrate to the account of their own system. Please refer to the detailed code in “common.js”.

Common APIs of Account:

<table>
<thead>
<tr>
<th>API name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getToken()</td>
<td>Get access token of current user</td>
</tr>
<tr>
<td>getUserAccount()</td>
<td>Get account name of current user</td>
</tr>
<tr>
<td>getUserConfig()</td>
<td>Get configuration data of current user</td>
</tr>
<tr>
<td>getUserId()</td>
<td>Get user ID of current user</td>
</tr>
<tr>
<td>getWatermarkInfo()</td>
<td>Get the information of user watermark</td>
</tr>
<tr>
<td>initUserConfig()</td>
<td>Initialize to request current user configuration data after user login</td>
</tr>
<tr>
<td>isLogin()</td>
<td>Check whether user has login</td>
</tr>
<tr>
<td>setAccessToken()</td>
<td>Set access token of current user</td>
</tr>
<tr>
<td>setLoginState()</td>
<td>Set login status of current user</td>
</tr>
<tr>
<td>setUserAccount()</td>
<td>Set account of current user</td>
</tr>
<tr>
<td>getUserId()</td>
<td>Set ID of current user</td>
</tr>
<tr>
<td>setWatermarkInfo()</td>
<td>Set the user watermark information</td>
</tr>
</tbody>
</table>

3.1.7 Form

From version 1.2, WebPDF Viewer supports Acroform filling, as well as importing and exporting form data. Call export/import form data REST API 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

```javascript
$("#btnPrint").click(function(){
    WebPDF.ViewerInstance.print();
});
```
### Common APIs of Signature:

<table>
<thead>
<tr>
<th>API Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exportToXML()</td>
<td>Export form data to XML</td>
</tr>
<tr>
<td>exportToXMLStream()</td>
<td>Export form data to XML stream</td>
</tr>
<tr>
<td>exportXMLToUrl()</td>
<td>Export form data (XML format) to a remote server (url).</td>
</tr>
<tr>
<td>highlight()</td>
<td>Highlight all form fields.</td>
</tr>
<tr>
<td>importFromXML()</td>
<td>Import form data (XML format) from local machine</td>
</tr>
<tr>
<td>importFromXMLStream()</td>
<td>Import from data from XML stream</td>
</tr>
<tr>
<td>importXMLFromUrl()</td>
<td>Import form data (XML format) from a remote server (url).</td>
</tr>
<tr>
<td>isHighlightBarVisible()</td>
<td>Check the highlight bar visible or not.</td>
</tr>
<tr>
<td>showHighlightBar()</td>
<td>Show or hide current highlight bar.</td>
</tr>
</tbody>
</table>

### 3.1.8 Watermark

WebPDF Viewer 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 (check on the Deployment Guide). 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 Viewer will process the watermark data from the cookies, and then apply it in the document with “setWatermarkInfo()” when opening a PDF document with watermarks.
3.2 WebPDF Viewer on Mobile

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

File 1: ..\webpdf\viewer\webapp\mobile\index.html
File 2: ..\webpdf\viewer\webapp\scripts\control\common (All the common JavaScript files)
File 3: ..\webpdf\viewer\webapp\scripts\control\common\mobile (All the JavaScript files of Mobile view)
File 4: ..\webpdf\viewer\webapp\styles\reader\mobile (css files of Mobile view)
File 5: FoxitWebPDF_APIREFERENCE (API file)

3.2.1 Open a PDF file on your own system

After developer implement the getDocument REST API, the PDF files in the network can be opened by transferring the PDF URL to WebPDF Viewer.

Please refer to the detailed code in ..\webpdf\modules\reader\webapp\scripts\control\mobile\index.js.

See Common APIs of Viewer.

3.2.2 Annotation

See Annotation on PC.

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

3.2.3 Save Annotation

See Save Annotation on PC.

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

Example: Click the Save button

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

3.2.4 Rotation

See Rotation on PC.
Please refer to the detailed code in “index.html” and “more.js”.

Example: Click the Rotate button

```javascript
$("#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;
});
```

3.2.5   Account

See Account on PC

Please refer to the detailed code in “common.js”.

3.2.6   Web PDF Form

See Web PDF Form on PC

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

3.2.7   Watermark

See Watermark on PC.

4   Demo

From WebPDF Viewer 2.2, Viewer provides demos for developer to refer to.

There are 5 demos, please find it from:

..\webpdf\sample\webpdf-extension-sample\src\main\webapp\sample

From: You will find the Viewer can handle the PDF form data.
**Hide-show-specific-user-annotation:** You will find the Viewer supports to hide and show specific user’s annotation.

**Navigation-panel:** You will find Viewer supports to create new panel, and then load an html page.

**Sharing-annotation:** You will find Viewer supports to load multiple user’s annotation. And collaborate among multiple users.

**User-permission:** You will find Viewer supports to handle different kinds of user’s permission.

## 5 FAQ

1. **Is there any file size limitation for WebPDF Viewer?**
   Foxit WebPDF Viewer 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 Viewer, 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 Viewer display corrupted in my browser?**
   If you are using IE for viewing purposes, please check whether the browser is compatible. WebPDF Viewer 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 Viewer?**
   Please find the log information of WebPDF Viewer in the log folder in the root of the installation directory. You can clear the log files when the folder is full.

5. **Why is the number of pages of the output file more than that of the original file in WebPDF Viewer 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.

6. **Why I cannot open a PDF document with password even though the input is a correct password?**
   Please make sure to turn on the session function of web container when deployment WebPDF Viewer since WebPDF Viewer need to cache user session.

7. **Why there is no signature function when I run the demo?**
   It might because the rendering engine is set to JavaScript rendering in the console. The JavaScript Rendering engine does not support any signature feature at present in the demo.

8. **How can I make WebPDF Viewer run successfully if the front-end of WebPDF Viewer is hosted on a different domain from the back-end?**
   If your project which needs to be integrated with WebPDF Viewer is in a different domain from the WebPDF Viewer, you need to copy all the static files (front-end) of WebPDF Viewer into the
webapp folder of your project. Also, you should modify the parameter of customizedUrl when you initialize the Viewer in ../viewer/webapp/scripts/control/pc/demo.js.

For example:

```javascript
$(document).ready(function() {
    var optionsParams = {
        language: getLanguage(),
        serverBaseUrl: "http://127.0.0.1:8080/", // change the ip of Url to the one your webpdf viewer install
        baseUrl: getBaseUrl()
    };
    WebPDF.ready(docViewerId, optionsParams).then(function(data) {
        ...
    });
});
```

9. Why Eclipse reports “The method getTextContent() is undefined” always in org.w3c.dom.Node class when running webpdf-extension-sample project?
The project uses dom4j->xml-apis, it calls node class from dom4j but not from JDK. So it reports node class error. To solve the problem, right click properties of the project, select builder path, and then move the JDK up to maven dependencies, it will call the node class from JDK. The error will disappear.
10. **Can I disable rectangle text selection?**

Some users prefer the text selection always be line text selection, and want to disable rectangle selection. From 2.2, it is supported. Find config.js, and set below parameter to false, the rectangle selection will be disable.

```javascript
RectangularTextSelection: false,
```
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