Office Native WOPI Integration Documentation

Release 2016.01.27

unknown

Sep 20, 2019

OVERVIEW

Note

This documentation is a work in progress. Topics marked with a are placeholders that have not been written yet. You can track the status of these topics through our public documentation issue tracker.

You can use the Web Application Open Platform Interface (WOPI) protocol to integrate Office for iOS and Android with your application. The WOPI protocol enables Office for iOS and Android to access and change files that are stored in your service.

To integrate your application with Office for iOS and Android, you need to do the following:

1. Be a member of the Office 365 - Cloud Storage Partner Program. Currently integration with Office for iOS and Android using WOPI is available to cloud storage partners. You can learn more about the program, as well as how to apply, at http://dev.office.com/programs/officecloudstorage.
2. Implement the WOPI protocol - a set of REST endpoints that expose information about the documents that you want to view or edit in Office for iOS and Android. The set of WOPI operations that must be supported is described in the section titled *WOPI implementation requirements for Office for iOS and Android integration*.
3. Provide the required on-boarding information as described in the section titled *Onboarding information*.

CHAPTER

ONE

INTEGRATING WITH OFFICE FOR IOS AND ANDROID

You can integrate with Office for iOS and Android to enable your users to view and edit Excel, PowerPoint, and Word files directly on their mobile devices.

If you offer both iOS and Android experience, then we recommend you integrate both the experiences simultaneously. However, You can choose to onboard either of the experience first depending on your priority. Please note that once you have onboarded to WOPI APIs for any of the one experience i.e. Android or iOS experience, integrating with the other experience is very minimal work.

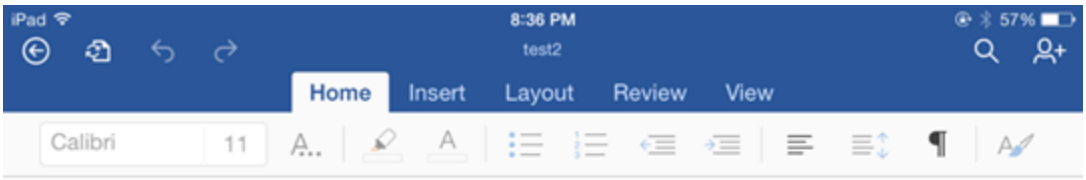


Figure 1.1: Editing a file in Office for iOS

If you deliver an iOS and/or Android experience that allows your users to store Office files or includes Office files as a key part of your solution, you now have the opportunity to integrate Office for iOS and Android into your experience. This integration works directly against files stored by you. Your users won't need a separate storage solution to view and edit Office files.

1.1 Editing and viewing Office files

When you integrate with Office for iOS and Android, your users can view and edit Excel, PowerPoint, and Word files directly on the mobile device. The level of editing support is determined by the type of user and whether they have an Office 365 subscription.

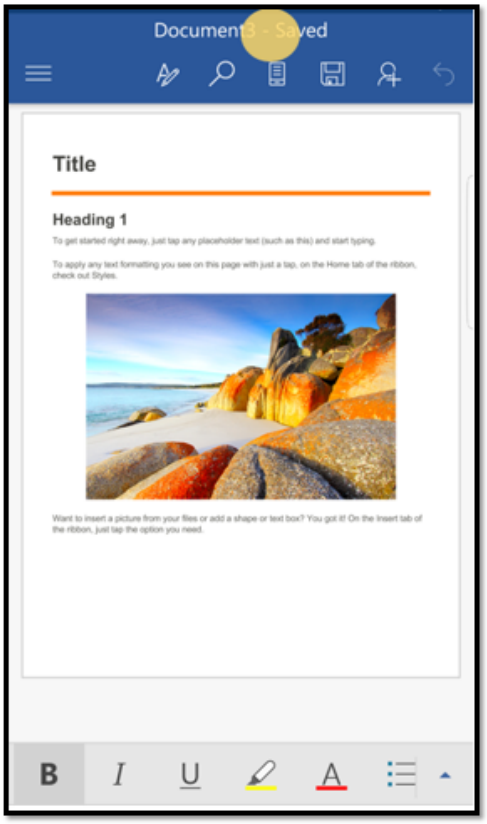


Figure 1.2: Editing a file in Office for Android

1.2 Integration process

You can use the Web Application Open Platform Interface (WOPI) protocol to integrate Office for iOS and Android with your application. The WOPI protocol enables Office for iOS and Android to access and change files that are stored in your service.

To integrate your application with Office for iOS and Android, you need to do the following:

1. Become a member of the Office 365 - Cloud Storage Partner Program. You can learn more about the program, as well as how to apply, at lcsppl.
2. Provide the required on-boarding information as described in the section titled *Onboarding information*.
3. Obtain your ProviderId and app store URLs from Microsoft. The app store URLs are the URLs you should use to launch Office on a platform's app store.
4. Add your domain to the WOPI domain allow list. This is required even if you do not integrate with Office for the web so you can verify your WOPI integration with the *validator app*.
5. Implement the WOPI protocol. The set of WOPI operations that must be supported is described in the section titled *WOPI implementation requirements for Office for iOS and Android integration*.
6. Implement required changes to your app. Contact your Office for iOS and Android integration contacts to receive directions to run Office for iOS and Android in test mode, to enable testing.
7. Run the *validator app* and fix any issues until the validator reports a 100% pass rate.
8. Complete the manual testing outlined in the *Manual Validation section* to verify the integration works as expected.
9. Once all manual test cases have a 100% pass rate, send your `Manual Validation report` and directions to run your validator.
10. Once the Office for iOS and Android team completes their manual testing, they will work with you to schedule your launch date.

1.2.1 Authentication

Authentication is handled by passing Office for iOS and Android an access token that you generate from a Bootstrapper URL. Assign this token a reasonable expiration date. Also, we recommend that tokens be valid for a single user against a single file to help mitigate the risk of token leaks.

1.2.2 Requirements

- You need to ensure that files are represented by a unique ID. See the full list of File ID requirements.
- You should have a mechanism for identifying file versions. See the Version requirements.
- In order to integrate with Office for iOS and Android, there are also a few promotional requirements which include:
  - Promoting Office for iOS and Android integration somewhere within your app
  - Promoting Office for iOS and Android integration in the context of editing and viewing Office documents
  - Using Office as the default app for opening Office documents within your app

1.2.3 Security Considerations

Office for iOS and Android is designed to work for enterprises that have strict security requirements. To make sure your integration is as secure as possible, ensure that:

- All traffic is SSL encrypted
- Server needs to support TLS 1.0+
- OAuth 2.0 is supported

1.3 Interested?

If you're interested in integrating your solution with Office for iOS and Android, take a moment to register at Office 365 Cloud Storage Partner Program.

CHAPTER

TWO

DIFFERENCES BETWEEN OFFICE FOR IOS, OFFICE FOR ANDROID AND OFFICE FOR THE WEB

Unlike Office for the web, partners need to implement all operations on the Container, Ecosystem and Bootstrapper endpoints. You can find more information about these operations in the *WOPI implementation requirements for Office for iOS and Android integration* section.

CHAPTER

THREE

WOPI IMPLEMENTATION REQUIREMENTS FOR OFFICE FOR IOS AND ANDROID INTEGRATION

Note: Following section is applicable Office for iOS and Android. Both of these endpoints require the same level of WOPI implementation.

This section documents specific requirements for your WOPI implementation for integration with Office for iOS and Android beyond what is documented in general for WOPI. You can learn more about how Office for iOS and Android uses these operations in the *Operational flows for Office for iOS and Office for Android* section.

3.1 Required WOPI Operations for Office for iOS and Android

The following WOPI operations are required for integration on Office for iOS and Android. Operations not listed here are not currently called by Office for iOS and Android.

If some operations are not supported for a specific user/file/container; just make sure to return the correct values in CheckFileInfo and CheckContainerInfo.

3.1.1 File Operations

- CheckFileInfo
- GetFile
- Lock
- GetLock
- RefreshLock
- Unlock
- UnlockAndRelock
- PutFile
- EnumerateAncestors (files)

3.1.2 Container Operations

- CheckContainerInfo
- CreateChildFile
- EnumerateAncestors (containers)

- EnumerateChildren (containers)

3.1.3 Ecosystem Operations

- CheckEcosystem
- GetRootContainer (ecosystem)

3.1.4 Bootstrapper

- Bootstrap
- GetNewAccessToken
- GetRootContainer (bootstrapper)

3.1.5 Future Support

While these WOPI operations are not currently used by Office for iOS and Android, they must be implemented. Office for iOS and Android will use these operations in the future.

- RenameFile
- DeleteFile
- CreateChildContainer
- DeleteContainer
- RenameContainer
- GetEcosystem (files)
- GetEcosystem (containers)

3.1.6 Other Requirements

- The **X-WOPI-ItemVersion** header must be included on PutFile, Lock, and Unlock responses
- For the Bootstrap operation, the Content-Type response header must be set to `application/json`
- IsEduUser and LicenseCheckForEditIsEnabled are required on CheckFileInfo and CheckContainerInfo. The values from CheckFileInfo must match that of the file's parent container.

CHAPTER

FOUR

BROWSING, OPENING, AND SAVING FILES FROM OFFICE FOR IOS AND OFFICE FOR ANDROID

Once you have finished integration with Office for iOS and Office for Android, users will be able to browse, open and save files from your storage location directly in the Office apps as well as open their Office files from your storage location. This section describes the user experience for browsing, opening and saving files from Office for iOS and Office for Android.

4.1 Open from Office for iOS

1. User selects *Add a Place* in their Office app.
2. A list of places is displayed and the user selects *Contoso*.
3. The user goes through Contoso's OAuth 2.0 flow.
4. The user can now browse through files stored on Contoso and open them.
5. The user can edit the file.

4.2 Open from Office for Android

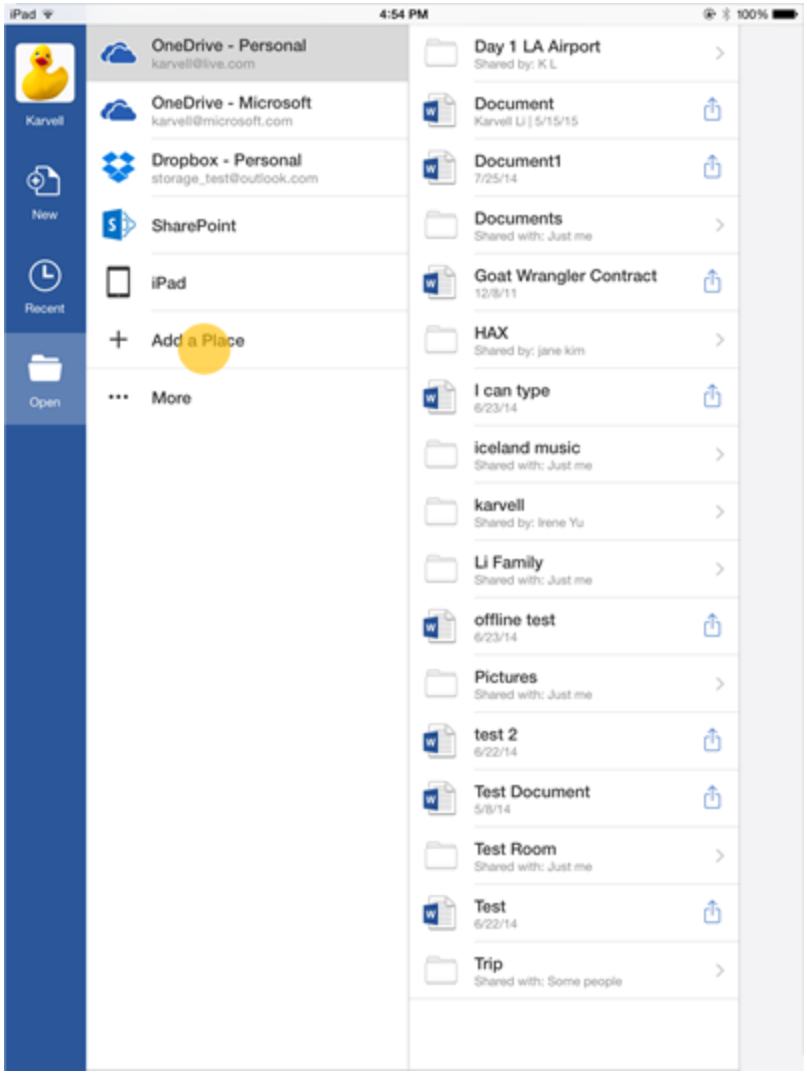
1. User selects *Add a Place* in their Office app.
2. A list of places is displayed and the user selects *Contoso*.
3. The user goes through Contoso's OAuth 2.0 flow.
4. The user can now browse through files stored on Contoso and open them.
5. The user can edit the file.

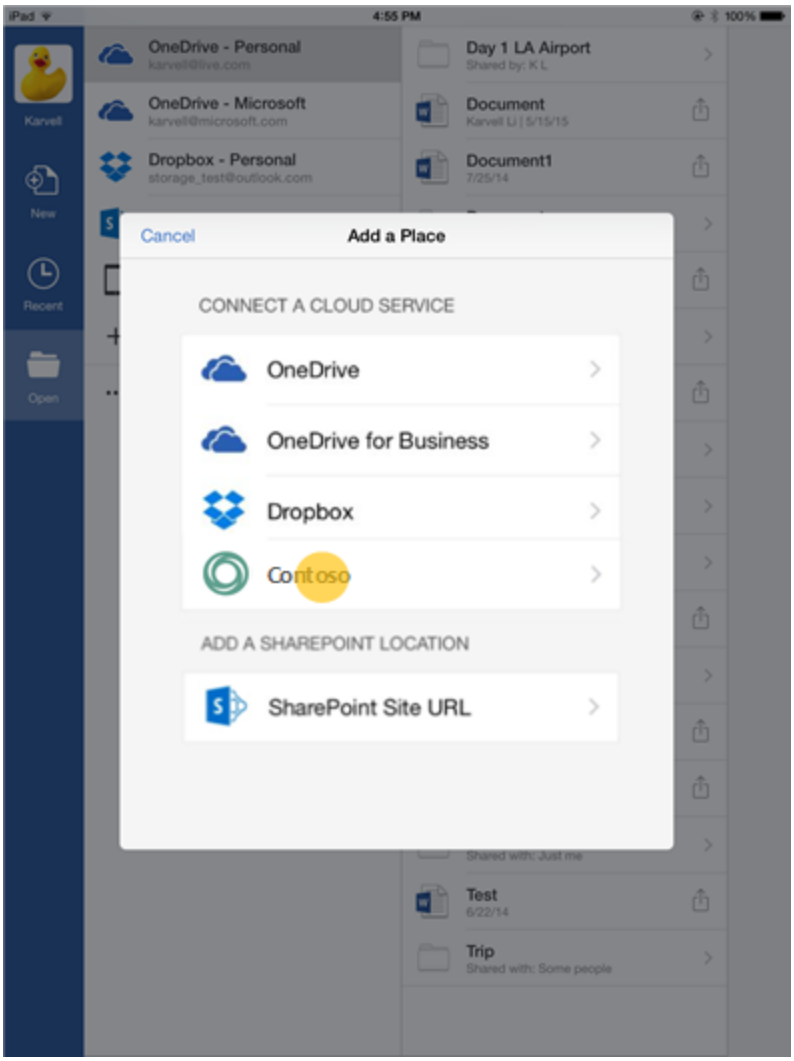
4.3 Open from your Cloud Storage App

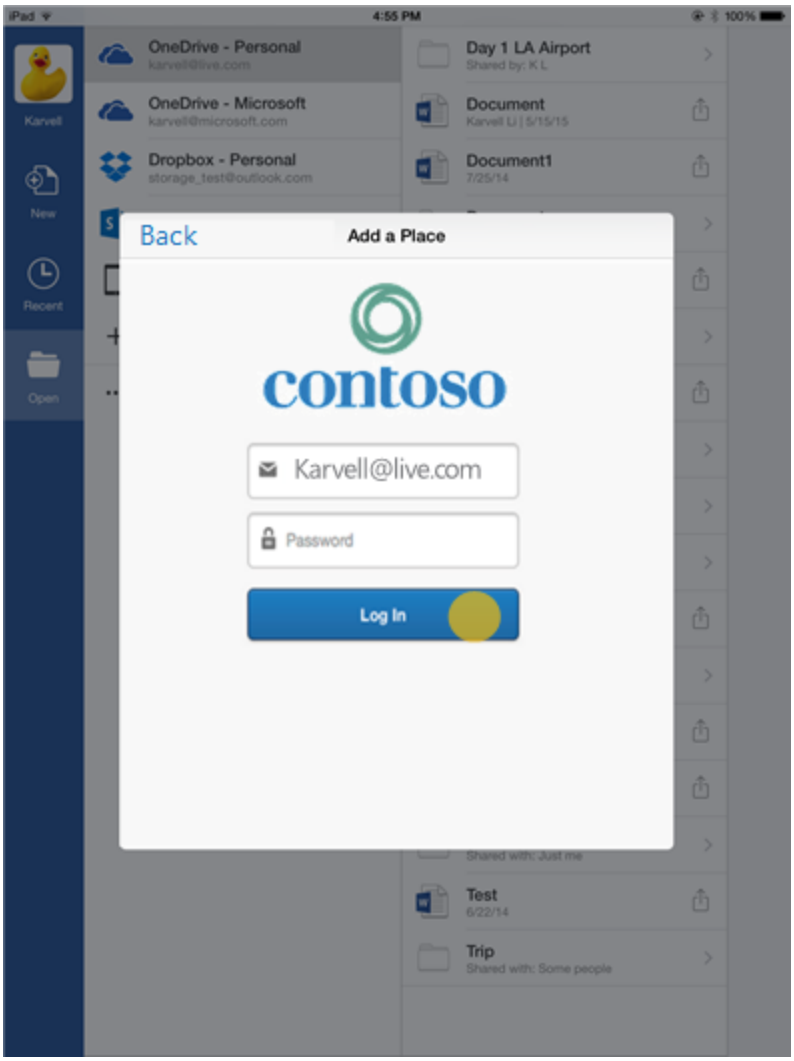
1. User browses to a file on your iOS or Android app.
2. User selects Edit.

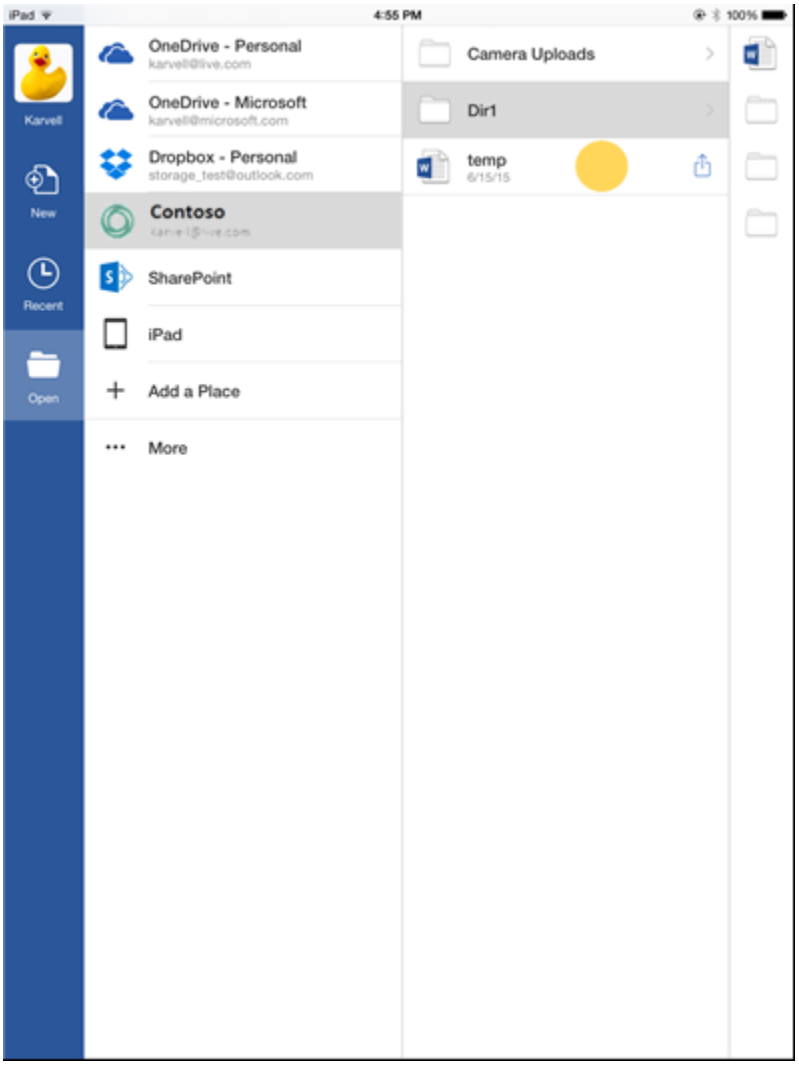
   Note: If you do not have an in-app preview, you may open Office in other screens.

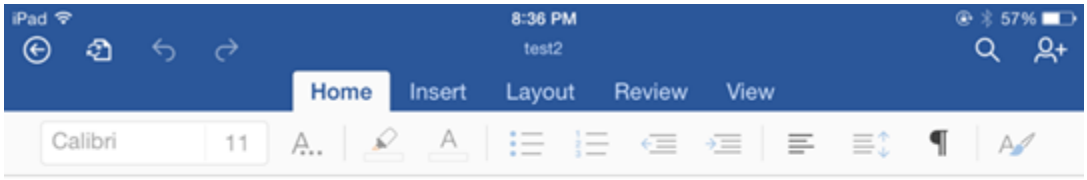
3. The file is opened in Office and any saves go back to your service directly.

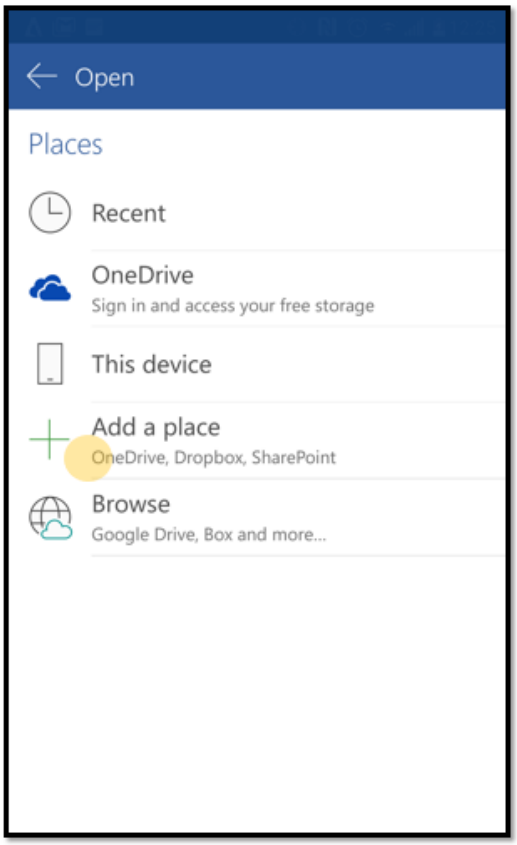

iPad
4:54 PM
100%
Karvell
New
Recent
Open
OneDrive - Personal
karvell@live.com
OneDrive - Microsoft
karvell@microsoft.com
Dropbox - Personal
storage_test@outlook.com
SharePoint
iPad
Add a Place
More
Day 1 LA Airport
Shared by: K L
Document
Karvell Li | 5/15/15
Document1
7/25/14
Documents
Shared with: Just me
Goat Wrangler Contract
12/8/11
HAX
Shared by: jane kim
I can type
6/23/14
iceland music
Shared with: Just me
karvell
Shared by: Irene Yu
Li Family
Shared with: Just me
offline test
6/23/14
Pictures
Shared with: Just me
test 2
6/22/14
Test Document
5/8/14
Test Room
Shared with: Just me
Test
6/22/14
Trip
Shared with: Some people

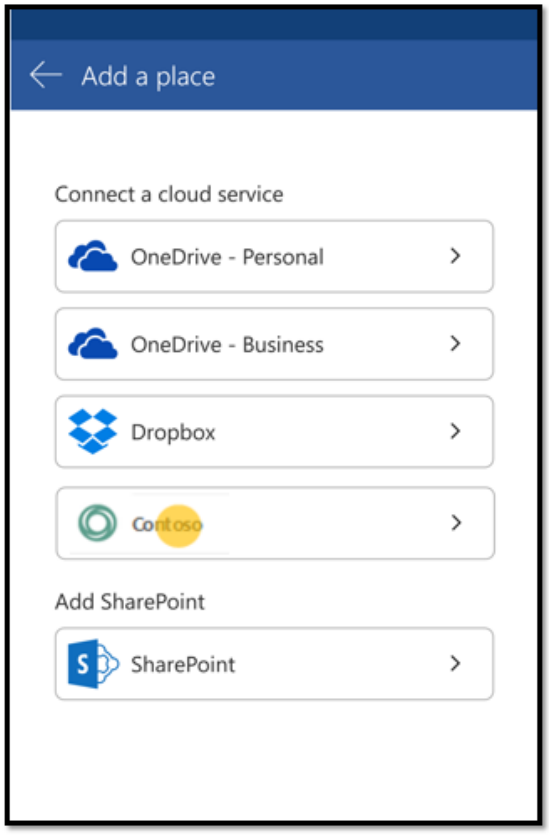

iPad
4:55 PM
100%
OneDrive - Personal
karvell@live.com
OneDrive - Microsoft
karvell@microsoft.com
Dropbox - Personal
storage_test@outlook.com
Karvell
New
Recent
Open
Day 1 LA Airport
Shared by: K L
Document
Karvell Li | 5/15/15
Document1
7/25/14
Cancel
Add a Place
CONNECT A CLOUD SERVICE
OneDrive
OneDrive for Business
Dropbox
Contoso
ADD A SHAREPOINT LOCATION
SharePoint Site URL
Test
6/22/14
Trip
Shared with: Some people


iPad
4:55 PM
100%
Karvell
New
Recent
Open
OneDrive - Personal
karvell@live.com
OneDrive - Microsoft
karvell@microsoft.com
Dropbox - Personal
storage_test@outlook.com
Day 1 LA Airport
Shared by: K L
Document
Karvell Li | 5/15/15
Document1
7/25/14
Back
Add a Place
contoso
Karvell@live.com
Password
Log In
Test
6/22/14
Trip
Shared with: Some people

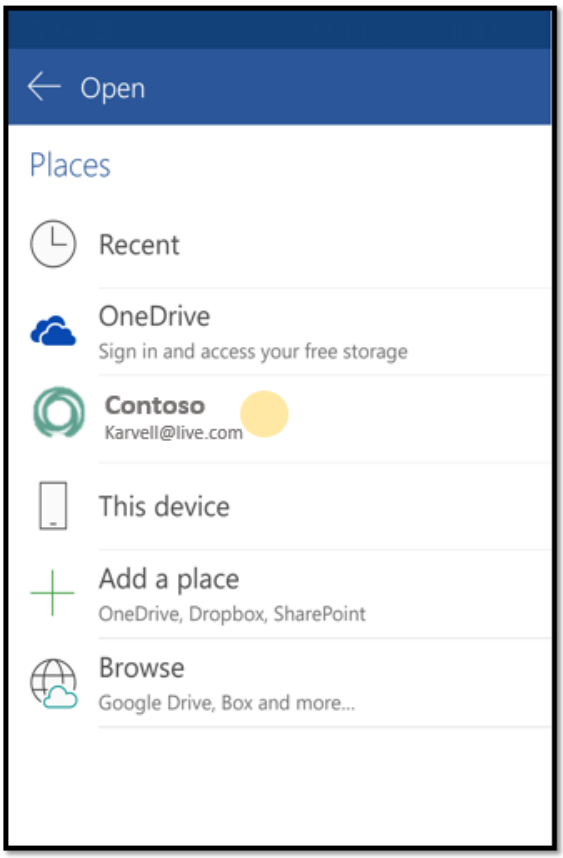

iPad
4:55 PM
100%
Karvell
New
Recent
Open
OneDrive - Personal
karvell@live.com
OneDrive - Microsoft
karvell@microsoft.com
Dropbox - Personal
storage_test@outlook.com
Contoso
SharePoint
iPad
Add a Place
More
Camera Uploads
Dir1
temp
6/15/15

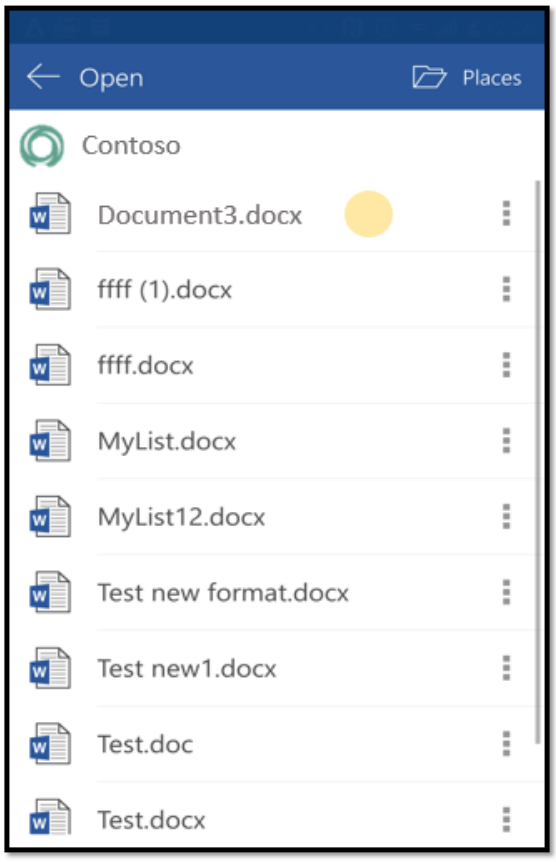

iPad
8:36 PM
57%
test2
Home
Insert
Layout
Review
View
Calibri
11


Open
Places
Recent
OneDrive
Sign in and access your free storage
This device
Add a place
OneDrive, Dropbox, SharePoint
Browse
Google Drive, Box and more...


Add a place
Connect a cloud service
OneDrive - Personal
OneDrive - Business
Dropbox
Contoso
Add SharePoint
SharePoint

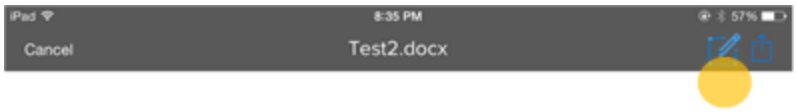

contoso
Karvell@live.com
Password
Log In


Open
Places
Recent
OneDrive
Sign in and access your free storage
Contoso
Karvell@live.com
This device
Add a place
OneDrive, Dropbox, SharePoint
Browse
Google Drive, Box and more...

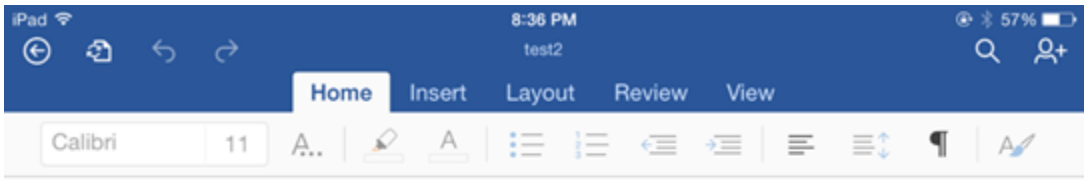

Open
Places
Contoso
Document3.docx
ffff (1).docx
ffff.docx
MyList.docx
MyList12.docx
Test new format.docx
Test new1.docx
Test.doc
Test.docx


Document3 - Saved
Title
Heading 1
B
I
U


iPad
8:35 PM
57%
Test2



wffsdafasdfa



CHAPTER

FIVE

OPENING FILES FROM YOUR APP IN OFFICE FOR IOS AND OFFICE FOR ANDROID

In order to invoke Office for iOS or Office for Android when opening an Office file from your app, you must use the URL schemes that Word, Excel, and PowerPoint for iOS or Android register when they are installed.

5.1 URL schemes for invoking Office for iOS and Office for Android

The following is the list of scheme names used to invoke :

- `ms-word:`
- `ms-powerpoint:`
- `ms-excel:`

The following information must be included along with the URL scheme:

- The mode to open the file in. Valid values are:
  - `ofv` for opening as read-only
  - `ofe` for opening for editing.
- The WOPISrc to the file, denoted by the `|u|` parameter
- The service identifier, denoted by the `|d|` parameter
- The user identifier, denoted by the `|e|` parameter
- The file name, with extension, denoted by the `|n|` parameter
- The source of the open action with `|a|`. The value of this parameter can be either:
  - `Web` – to be used in the case where a website is invoking the protocol handler
  - `App` – to be used in the case where a native app is invoking the protocol handler

Example:

```
ms-word:ofe|u|https://contoso/wopi/file/12312|d|Contoso|e|a3243d|n|document1.
↪docx|a|App
```

The URL used should be URL encoded.

Note that the file is opened directly against your service. Your app essentially passes the URL to the file to Office for iOS or Office for Android without passing the actual file. The Office for iOS or Office for Android app then opens the file directly using the WOPI protocol.

5.2 Identifying supported Office for iOS versions

Before using the URL schemes specified above to invoke Office for iOS, you must do the following:

1. Verify the particular Office for iOS application (Word, Excel, or PowerPoint) are installed.

   Use the iOS `canOpenURL` method to determine whether your application can open the resource. This method takes the URL for the resource as a parameter, and returns No if the application that accepts the URL is not available. If `canOpenURL` returns No, you'll need to prompt the user to install Office for iOS. Contact Microsoft to obtain links to install Office for iOS specific to you.

2. Check the version of Office for iOS installed is a supported version. This is done by verifying whether the following URL schemes are registered:

   - `ms-word-wopi-support-1605`
   - `ms-powerpoint-wopi-support-1605`
   - `ms-excel-wopi-support-1605`

Important: These URL schemes are only used to check if the currently installed Office for iOS supports opening files from a WOPI host and not for invoking the Office for iOS apps.

5.3 (Optional) Passback protocol for Office for iOS

If you want Office for iOS to return users to your iOS application when they choose the in app back arrow (distinct from the iOS back control), you will need to include the passback parameter when invoking Office for iOS. This is denoted by `|p|` followed by your app's registered URL scheme (without a colon).

You must ensure that your application can properly handle the response from Office for iOS.

Tip: You'll provide your app's registered URL scheme during the initial integration phase.

For security reasons, Office for iOS only returns users to the referring application if the file opened successfully. When the user chooses the back arrow, Office for iOS responds to the invoking application with the passback protocol, open mode, URL, upload pending status, and `document context`. The upload pending status uses the descriptor `|z|`, and is either yes or no.

`document context` is a string you provide via the `|c|` parameter when invoking Office for iOS. Office for iOS doesn't use this parameter; it is purely for your use, as needed by your app. Office for iOS does not limit the length of the string beyond any limits imposed by the operating system.

Schema format invoking your app when user chooses back:

```
<app protocol>:ofe|u|<URL>|z|<yes or no>|c|<doc context>
```

Example:

```
contosodrive:ofe|u|https://contoso/Q4/budget.docx|z|no|c|folderviewQ4
```

5.4 Office for Android Promotion guidance

Step 1 - Verify that Office has been installed

Your referring application will first need to verify that a particular Office application is installed. The following Office applications can be installed on Android devices for document viewing and editing:

- Excel
- PowerPoint
- Word

Use Android PackageManager to determine whether a particular Office application is installed on the device. The following table lists the package names for the Office applications that you can use in this process.

Application	Package Name
Excel	`com.microsoft.office.excel`
PowerPoint	`com.microsoft.office.powerpoint`
Word	`com.microsoft.office.word`

If yes Office is installed, go to Step 2a. Else go to Step 2b/2c.

Step 2 - Integrate promotional logic in your app

Step 2a - When Office apps are installed - Check version of Office apps

- Make sure Office apps are greater than 16.0.XXXX.XX version.

  Note: Exact Version number will be provided by Office on Android team.

- Guidance to determine Office application version number

  Use Android PackageInfo to determine whether a particular version of Office application is installed on the device

Example:

```
PackageInfo pInfo = getPackageManager().getPackageInfo(getPackageName(), 0);
String version = pInfo.versionName;
```

Step 2b - When Office apps not installed - promote via Google Play store

Use adjust URLs to throw market intent to install Office apps. These links will be created by Office Android team for you. These links will redirect to following Google Play store page for corresponding Office apps.

Application	Google Play Store
Excel	`https://play.google.com/store/apps/details?id=com.microsoft.office.excel`
PowerPoint	`https://play.google.com/store/apps/details?id=com.microsoft.office.powerpoint`
Word	`https://play.google.com/store/apps/details?id=com.microsoft.office.word`

Step 2c - When Office apps not installed - promote via China stores

In China, Office apps are uploaded on following app stores. Since Google Play is not supported in China, app installs would happen from one of the following app stores.

Stores	Word
Baidu	http://shouji.baidu.com/software/9450548.html
360	http://zhushou.360.cn/detail/index/soft_id/2483089
Tencent	http://android.myapp.com/myapp/detail.htm?apkName=com.microsoft.office.word
Wandoujia	http://www.wandoujia.com/apps/com.microsoft.office.word
Xiaomi	http://app.mi.com/detail/91625
Huawei	http://appstore.huawei.com/app/C10586094
Lenovo	http://www.lenovomm.com/app/20682833.html
Oppo	http://store.oppomobile.com/product/0010/458/460_1.html?from=1152_2

Note: Vivo Store - Coming Soon

Following guidance demos Tencent integration. These guidelines can be modified as needed for any other China specific WOPI integration.

Tencent will need to launch the market intent by showing only those app stores where Office apps are present. In order to also track the number of launches in promotional flow, we will make a call to tracking URL (i.e. adjust URL). Following guidance goes over special handling to make a call to tracking URL (i.e. adjust URL) first, and then show the valid list of app stores for app installations.

Guidance:

1. Working prototype for this is present here:
   - MainActivity.java
   - AppCompatActivity.java
2. Please change following variable values as per guidance from Office for Android team.
   - ADJUST_CHINA_STORE_LINK
   - APP_PACKAGE_MAKETING_FOR
   - REFERRERSTRING

CHAPTER

SIX

OPERATIONAL FLOWS FOR OFFICE FOR IOS AND OFFICE FOR ANDROID

This section outlines the implementation requirements and operational flows for user scenarios outlined in the *Browsing, opening, and saving files from Office for iOS and Office for Android* section.

6.1 Terminology

User Self-explanatory

Office Client The Office for iOS and Android app, i.e. Excel/Word/PowerPoint

Office Identity Service The Office Identity service is a service Office for iOS and Android uses for handling user identity information. In OAuth flows, the Office Identity Service behaves as the External User Agent

Bootstrapper URL URL which the Storage Provider hosts to allow an Office Client app with a valid OAuth 2.0 access token to retrieve a WOPI access token

Token Endpoint URL URL which the Storage Provider hosts and uses in order to generate access or refresh tokens

Services Manager Office-owned service which stores a Service Catalog, listing all of the available services for an app or endpoint

Storage provider Refers to a CSPP partner who has integrated their service into Office for iOS and Android

---

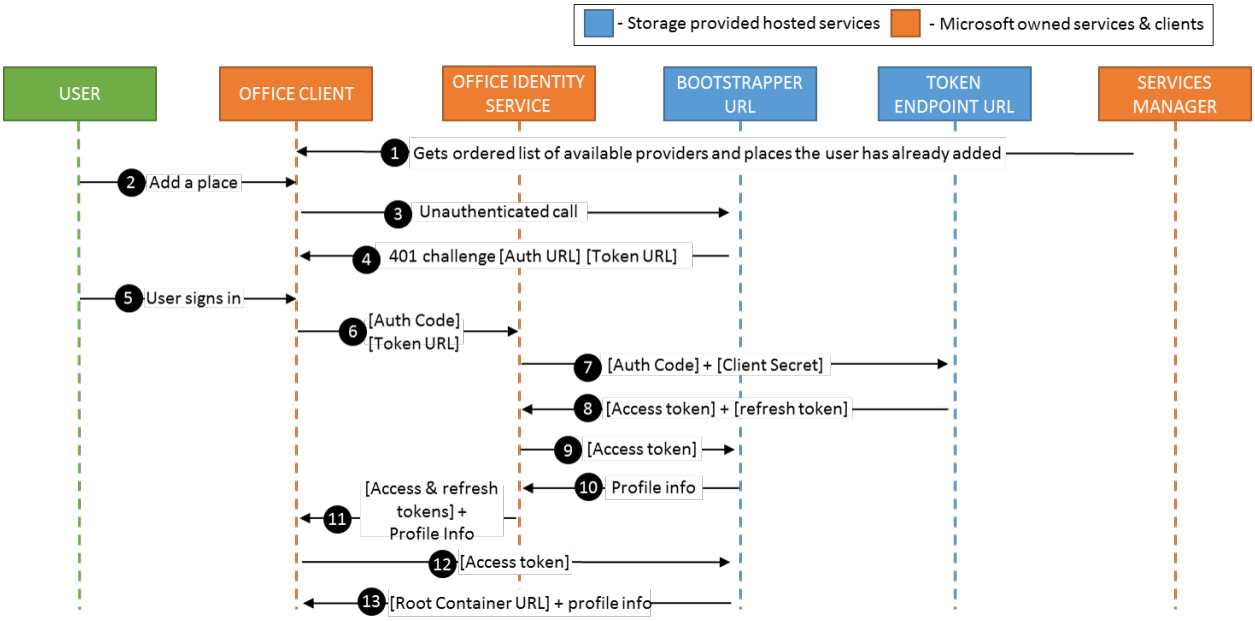
Note:

- Any time during these flows that Office for iOS and Android tries to hit the bootstrapper and have expired or invalid OAuth credentials, Office for iOS and Android will ask the user to log in again.
- Any time during these flows that Office for iOS and Android has a missing, invalid, or expired WOPI access token, Office for iOS and Android will attempt to get a valid WOPI access token by calling GetNewAccessToken.

6.2 Add a Place

This describes the first time a user adds your storage provider using the *Add a Place* in Office for iOS and Android. To correctly authenticate users, we use the following operational flow in order to connect a user's storage provider account with their Office for iOS and Android account.

1. When the Office client boots, it calls the Services Manager for a list of available services. The Services Manager returns a Service Catalog, which is an alphabetically-sorted list of available providers and places which the user can connect to.



2. When the user clicks *Add a Place* in the Office for iOS and Android backstage, they see your Storage Provider listed as an available place.
3. When the users adds the Storage Provider, the Office client makes an unauthenticated Bootstrap call.
4. The Bootstrap response includes the Authorization URI and token issue URI. Note that the Authorization URI must be provided as part of your *Onboarding information* so we can add it as a trusted domain.
5. Office for iOS and Android loads the Authorization URI in a web view which prompts the user to sign in with the service provider. Once the user finishes the sign in process, the web view reaches a redirect URI which causes it to close. The redirect URI also provides an auth code to Office for iOS and Android.
6. Office for iOS and Android sends the auth code and token issue URI to the Office Identity Service.
7. The Office Identity Service sends the auth code and the client secret to the Token endpoint.
8. The Token Endpoint issues an access token and a refresh token (if available) back to the Office Identity Service.
9. The Office Identity Service makes an authenticated Bootstrap call using the access token to retrieve the user profile information.
10. The Office Identity Service sends the access and refresh tokens and the user profile information to Office for iOS and Android.
11. The user has now added the Storage Provider as a place. For the operational flow on browsing, opening, and saving files, see the next sections.

6.3 Browsing and opening files

Here is the operational flow for browsing and opening files.

1. *Get the Root Container URL:* Office for iOS and Android calls GetRootContainer (bootstrapper) to obtain a Root Container URL.
2. *Get the contents of the container:* Office for iOS and Android calls EnumerateChildren (containers) on the Root Container. The results are a set of containers and files in the root container. If the user wants to browse to another container within the current container, Office for iOS and Android calls CheckContainerInfo on the

other container to check permissions, then calls EnumerateChildren (containers) on that second container. This step is repeated as the user browses the container hierarchy, until the user selects the file they want to open.

3. *Check file permissions:* Once the user selects a file, Office for iOS and Android calls CheckFileInfo on that file to verify that the user has permissions to the file.
4. *Check file lock:*
   - If the earlier CheckFileInfo call returned `true` for SupportsGetLock, Office for iOS and Android calls GetLock. If the GetLock response is a 409 Conflict or includes an **X-WOPI-Lock** header, the file is locked and Office for iOS and Android does not continue opening it.
   - If the earlier CheckFileInfo call returned `true` for SupportsGetLock, Office for iOS and Android sends a RefreshLock request with a known invalid lock ID. If the RefreshLock response is a 409 Conflict with a lock ID in the **X-WOPI-Lock** response header, the file is locked and Office for iOS and Android does not continue opening it.
5. *Take a lock on the file:* Office for iOS and Android calls Lock on the file, passing a lock ID it wishes to use in the **X-WOPI-Lock** request header. If the Lock call returns a 200 OK, the file is locked. Office for iOS and Android will use the same lock ID when making future PutFile requests.
6. *Download the file:* Office for iOS and Android makes a GetFile request on the file.

6.4 Saving and closing a file

1. *Save the file:* If the user has made changes to the file, Office for iOS and Android will update the file's contents by calling PutFile. The PutFile request will include the current WOPI lock ID previously used by Office for iOS and Android to lock the file.
2. *Unlock the file:* Office for iOS and Android will make an Unlock request against to unlock the file. This Unlock request will include the current WOPI lock ID previously used by Office for iOS and Android to lock the file.

CHAPTER

SEVEN

ONBOARDING INFORMATION

The following information must be supplied to Microsoft to enable end-to-end testing and release of Office for iOS integration.

Property	Data Type	Sample Value	Description
Localized Provider Name	String	Contoso	The name to display for this storage provider
Icon Locations	String[]	https://contoso.com/images/logo16.png https://contoso.com/images/logo32.png https://contoso.com/images/logo48.png https://contoso.com/images/logo64.png https://contoso.com/images/logo80.png https://contoso.com/images/logo96.png	The path to the provider logo icons, one for each of the following dimensions: • 16x16 • 32x32 • 48x48 • 64x64 • 80x80 • 96x96 Requirements: • must be PNG • icons must have at least a 1px white border to avoid bleed
BootstrapperUri	String	https://contoso.com/wopibootstrapper	The full path to the Bootstrapper endpoint. Must end in wopibootstrapper.
English Description	String	Free online storage. Store, access and share thousands of documents.	Used to show more information about the service. Please specify which part of the string should not be localized. E.g. your product or brand name if you include it in your description
Client ID	String	s6BhdRkqt3	OAuth2 Client ID (for Office)
Client Secret	String	7FjfpZBr1K3tDRbnfVdmIw9	OAuth2 Client Secret (for Office). Please do not provide this in email. This needs to be transferred securely.
Client App Name	String	Office	Please set this as the app name.
RedirectUri	String	https://localhost	The redirect URI used to return an authorization code. This URI is used as a known stop-URL and is not loaded by Office for iOS.
ProviderId	String	TP_CONTOSO	Supplied by Microsoft
TrustedDomain	String	contoso.com, qa-contoso.com	Known domains for bootstrapper, authorization and token issuance endpoints.
Scope (optional)	String	userprofile, editdocs	Set of comma-delimited scopes that are to be requested during authentication with the storage provider.
SupportRefresh	Bool	True	Denotes whether you will issue a refresh token that can be used to obtain a

7.1 Additional information required

- 3 test accounts on each environment you want us to onboard.
  - If your service has a commercial vs. consumer offering, please provide 3 of each.
- Website interface of each environment, if one exists.
- Whether each environment can be reached outside your network (if it's Internet accessible so we can use it)
- How many apps do you have on each platform? (E.g. on iOS, do you have a "for enterprise app" vs. a "for consumer app", or do you just have one?)
- If you have already integrated with Office for the web, please provide directions to access the validator.

7.2 Security Questions

- What is the expiry for access and refresh token for each environment?
- Do you actually check the redirect URI sent during OAuth2 flow against the one registered above, such that the authorization code would only ever be sent to the redirect URI?

CHAPTER

EIGHT

BOOTSTRAPPING OAUTH2

An **unauthenticated** request to the bootstrapper endpoint is the first step for the OAuth2 flow from Office for iOS to your application.

Unauthenticated means no access token is attached in the Authorization HTTP header, or an expired or otherwise invalid token is attached.

Important: The bootstrapper URL must be an HTTPS endpoint, and connections to the bootstrapper must be made using TLS (Transport Layer Security).

Important: The bootstrapper URL must be supplied as the BootstrapUrl property of the on-boarding information described in the section titled *Onboarding information*.

For full details, see the page Unauthenticated response.

CHAPTER

NINE

TOKEN ISSUANCE URL REQUIREMENTS

This section describes requirements for your token issuance endpoint (**RFC 6749#section-3.2**):

- Content-Type header must be set to `application/json`
- The OAuth2 access token expiration should be specified via the `expires_in` property, in seconds.
- If your access tokens do not expire, set it to a value of 0 (zero).

Example Response Body:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
    "access_token":"123dfevewafe",
    "token_type": "bearer",
    "expires_in": "86400"
}
```

CHAPTER

TEN

POST-AUTHORIZATION TOKEN ENDPOINT

The token endpoint URL (**RFC 6749#section-3.2**) is normally obtained from the initial unauthenticated call described at *Bootstrapping OAuth2*.

If the token endpoint URL cannot be determined before the end-user has completed the sign-in process, an alternative token endpoint URL may be supplied.

This is done via a `tk=` URL parameter appended to the value of the Location header from the 302 Found response at the end of the sign-in flow.

> **Important:** The contents of the `tk=` parameter must be URL encoded.

For example, to return the following information:

- Redirection URI is https://localhost
- Authorization code (**RFC 6749#section-4.1.2**) is "abcdefg"
- Token endpoint URL is https://contoso.com/api/token/?extra=stuff

The Location header in the 302 Found response would be:

```
Location: https://localhost?code=abcdefg&tk=https%3A%2F%2Fcontoso.com%2Fapi%2Ftoken%2F
↪%3Fextra%3Dstuff
```

As a result, all calls to the token endpoint for obtaining access token via authentication-code exchange, or refresh flows using the refresh token, will hit this URL instead of the one initially returned as described at *Bootstrapping OAuth2*.

CHAPTER

ELEVEN

OPTIONAL SESSION CONTEXT STRING

A Session Context String may be returned from the authentication process. Office for iOS will pass this string, in an HTTP header, in calls to the token endpoint URL (**RFC 6749#section-3.2**) and authenticated calls to the bootstrapper (GetNewAccessToken, Shortcut operations).

The Session Context String is optional, and for the use of the storage provider. A possible scenario would be to include a hint about a "tenant" so endpoints can know where they need to fetch and/or validate tokens.

Returning a Session Context string is done via an `sc=` URL parameter appended to the value of the Location header from the 302 Found response at the end of the sign-in flow.

> **Important:** The contents of the `sc=` parameter must be URL encoded.

For example, to return the following information:

- Redirection URI is https://localhost
- Authorization code (**RFC 6749#section-4.1.2**) is "abcdefg"
- Session Context String is "hello:World"

The Location header in the 302 Found response would be:

```
Location: https://localhost/?code=abcdefg&sc=hello%3AWorld
```

If present, the session context string will be included as an HTTP header when calls are made to the token exchange endpoint, and OAuth2 authenticated calls to the bootstrapper (GetNewAccessToken, Shortcut operations) as follows:

```
X-WOPI-SessionContext: hello:World
```

CHAPTER

TWELVE

OFFICE FOR IOS OAUTH2 SIGN-IN URL PARAMETERS

The HTTPS request to load the OAuth2 Authorization page (**RFC 6749#section-3.1**) includes standard OAuth2 parameters such as `client_id`, `redirect_uri`, etc. but may also include Office-specific parameters, documented here.

> **Important:** Early releases of Office for iOS may not include some or all of these URL parameters, so callers should fall back to reasonable defaults.

12.1 Culture

The request includes the Accept-Language header. The value of this header is determined by the iOS language settings. In addition, the following URL parameter is appended to the request URL:

```
rs=Culture
```

Culture contains the UI culture of the Office interface, e.g. `en-US`.

12.2 Build

The following URL parameter is appended to the request URL:

```
build=#
```

"#" will be a string showing the build of the Office application making the request.

12.3 Platform

The platform URL parameter communicates which platform the app is running on.

platform=iOS

12.4 Example

Here's a request to the sign-in endpoint showing all parameters in use:

```
https://contoso.com/api/oauth2/authorize?&rs=en-US&build=1.21.417&platform=iOS&client_
↪id=abcdefg&redirect_uri=https%3A%2F%2Flocalhost&response_type=code
```

CHAPTER

THIRTEEN

NOTES ON USE OF CLIENT SECRET

CHAPTER

FOURTEEN

APP TO APP SIGN IN FOR OFFICE FOR IOS AND OFFICE FOR ANDROID

The normal flow to sign in to your service from Office for iOS or Office for Android uses the OS UIWebView where your web sign in experience is rendered inside the Office for iOS or Office for Android app. Optionally, an additional optimization can be done where the user can sign in using your app.

The app to app flow involves Office for iOS or Office for Android invoking your app through your app's URL scheme to sign in to your service, and your app invoking Office for iOS or Office for Android when sign in is complete.

14.1 Sign in using your app

- When signing in to your service is required (i.e. the first time a file is opened from your app into Office, or when the user explicitly adds your service as a place in Office for iOS or Office for Android), Office for iOS or Office for Android calls your bootstrapper to obtain the `authorization_uri`, which it displays in a UIWebView. In addition to the `authorization_uri`, you should return a set of UrlSchemes that can be used to invoke your app.
- Office for iOS or Office for Android will first attempt to use the UrlSchemes to invoke your app. If none are returned, or if none of them are registered (i.e. your app is not installed), Office for iOS or Office for Android will fall back to the UIWebView.
- If the user does have your app installed, your app will be invoked via the UrlSchemes, and the user will be sent to your app to complete authentication. If the user declines to open your app, Office for iOS or Office for Android will fall back to the UIWebView.
- Once inside your app, you should do whatever is needed to obtain the user's auth code (for example, display "Grant permission to Office" dialog or ask for additional sign ins).
- Return the user to the Office for iOS or Office for Android app with the auth code via the Office URL scheme (see below).

14.2 Office for iOS Specific changes

14.2.1 URL scheme design

The data passed via the URL scheme is essentially the same as would be passed via `authorization_uri`.

Here is an example of a normal `authorization_uri` with parameters added. The parameters are described in RFC6749.

Invoking the sign in screen:

```
https://contoso.com/api/oauth2/authorize?client_id=abcdefg&redirect_uri=https%3A%2F
↪%2Flocalhost&response_type=code&scope=&rs=en-US&Build=16.1.1234&Platform=iOS
```

Host's redirect URL that ends the OAuth flow:

```
https://localhost?state=&code=abcdefg&tk=https%3A%2F%2Fcontoso.com%2Fapi%2Ftoken%2F
↪%3Fextra%3Dstuff
```

The same parameters are passed via the URL Schemes, with the addition of "action".

Office for iOS invoking your app ("To" URL):

```
Contoso:client_id=abcdefg&response_type=code&scope=wopi&rs=enUS&build=16.1.1234&
↪platform=iOS&app=word&action=76d173ad-a43f-4e3c-a5e7-0e7276b4c624
```

Your app invoking Office for iOS ("Back" URL):

```
ms-word-tp:code=abcdefg&tk=https%3A%2F%2Fcontoso.com%2Fapi%2Ftoken%2F%3Fextra%3Dstuff&
↪sc=xyz&action=76d173ad-a43f-4e3c-a5e7-0e7276b4c624
```

The values of the parameters should be URL encoded.

If authentication fails, the error parameters per RFC6749 can also be passed back to Office for iOS via the URL schemes, just as they can be passed back to Office for iOS via the redirect URI in the UIWebView model.

14.2.2 New URL schemes registered by Office for iOS

- `ms-word-tp`
- `ms-excel-tp`
- `ms-powerpoint-tp`

These are for Word, Excel, and PowerPoint respectively. Use these to invoke Office for iOS when the user is done with authentication on your side.

14.2.3 Action parameter

Action is a string passed to your app that should be passed back to Office unchanged.

14.3 Office for Android Specific changes

14.3.1 Service-side Changes

Adding information to the WWW-Authenticate response header on unauthenticated bootstrap requests

Office for Android will use Intent to invoke your App. The information Office needs are your App's Package name, Auth activity name and Version code. Office will consider provided Version code as the base version and will assume that your App with this Version and above will support App to App authentication.

The information Office needs is passed via the URLScheme parameter in the WWW-Authenticate response header to unauthenticated bootstrap request.

Parameter	Value	Required	Example
`Bearer`	n/a	Yes	Bearer
`authorization_uri`	The URL of the OAuth2 Authorization Endpoint to begin authentication against as described at: **RFC 6749#section-3.1**	Yes	https://contoso.com/api/oauth2/authorize
`tokenIssuance_uri`	The URL of the OAuth2 Token Endpoint where authentication code can be redeemed for an access and (optional) refresh token. See Token EndPoint at: **RFC 6749#section-3.2**	Yes	https://contoso.com/api/oauth2/token
`providerId`	A well-known string (as registered with with Microsoft Office) that uniquely identifies the host. Allowed characters: `[a-z,A-Z,0-9]`	No	TP_CONTOSO
`UrlSchemes`	Information used to invoke your app (despite the name of the parameter, this may not always be URL schemes; e.g. on Android, intent is used). This is an ordered list by platform. Omit any platforms you do not support. Office will attempt to invoke these in order before falling back to the WebView auth.	No	`{ "iOS": [ "contoso", "contoso-EMM" ], "Android": [ "Package1VersionCode", "Package1Name", "Package1AuthActivityName", "Package2VersionCode", "Package2Name", "Package2AuthActivityName" ], "UWP": [...] }`

14.3.2 Client-side Changes

Invoking your App on Android

1. Office will create an intent, which will take the package name and auth activity name of your App. We will set following two extras to intent:
    - `AuthorizeUrlQueryParams`: It is exactly same as used in iOS without the `action` parameter.

e.g.: `client_id=abcdefg&response_type=code&scope=wopi&rs=enUS&build=16.1.1234&platform=android&app=word`

- `UserId`: It will be an optional parameter and will be set whenever we will have it. Third party should use to verify that the sign in requested for the User signed in to their App.

Code sample 14.1: Sample intent creation code from App2AppSigninIntent.java

```
protected void LaunchIntent()
{
    Intent authIntent = new Intent();
    authIntent.setComponent(new ComponentName("com.android.thirdparty.packagename", "com.android.thirdparty.packagename.AuthActivityName"));
    authIntent.putExtra(AuthorizeUrlQueryParams, client_id=abcdefg&response_type=code&scope=wopi&rs=enUS&build=16.1.1234&platform =android&app=word);
    authIntent.putExtra(UserId, User123);
    startActivityForResult(authIntent, uniqueRequestCode)
}
```

2. After this Office will wait for result and will expect following from third party App

   - `ResponseUrlQueryParams`: Again this is exactly same as what we are getting in iOS minus the `action` parameter. The following are values of it in success and failure cases:

     – `code=abcdefg&tk=http://contoso.com&sc=xyz` [during RESULT_OK]

     – `error=invalid_request&error_description="optional human readable message"` [during RESULT_CANCELLED or anything else]

   - `UserId`: Third party should send which user is authenticated by it. Office will use it to show error in case `UserId` in request and response mismatch.

Code sample 14.2: Sample code from App2AppSigninIntent.java

```
protected void onActivityResult(int requestCode, int resultCode, Intent result)
{
    // Check which request we're responding to
    if (requestCode == uniqueRequestCode)
    {
         // Make sure the auth request was successful
         if (resultCode == RESULT_OK)
         {
           //successfully authed
           // Here we will assume that result will contain ResponponseUrlQueryParams
           // which will look like this code=abcdefg&tk=http://contoso.com&sc=xyz
          }
          else if (resultCode == RESULT_CANCELED)
          {
          //auth request cancel
          // Here we will assume that result will contain ResponponseUrlQueryParams
          // But this time we will not get tk and session context but may get error and error_description.
          }
     }
     else
     {
         //failed
         // Here we will assume that result will contain ResponponseUrlQueryParams
```

(continues on next page)

(continued from previous page)

```
        // But this time we will not get tk and session context but may get error and error_description.
    }
}
```

This is the work which your App needs to do

3. Add an intent filter to AndroidManifest.xml

Code sample 14.3: Add an intent filter to AndroidManifest.xml

```
<activity android:name="AuthActivityName">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
    </intent-filter>
</activity>
```

4. Handle the intent in AuthActivity

Code sample 14.4: Handle the intent using sample code from HandleIntent.java

```
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    // Get the intent that started this activity
    Intent intent = getIntent();
    // Get the bundle of Extras, it will have data set by caller App
    Bundle extras = intent.getExtras();
}
```

5. Returning result

Code sample 14.5: Return the result using sample code from HandleIntent.java

```
protected void returnResult()
{
    // create intent to deliver result data
    Intent result = new Intent();
    resut.putExtra("ResonponseUrlQueryParams", "code=abcdefg&tk=http://contoso.com&sc=xyz");
    setResult(Activity.RESULT_OK, result);
    finish();
}
```

More details here https://developer.android.com/training/basics/intents/filters.html#ReturnResult

CHAPTER

FIFTEEN

MANUAL VALIDATION

Your WOPI integration can be tested using the steps outlined in *Using the validator application to validate your WOPI implementation*. However, the validator app will not be able to test the complete end-to-end experience. You must run the following tests manually to verify the integration works correctly.

Please download the `Testing` document and fill it in as you perform your testing. After both the validator and manual tests are both 100% passing, please provide Microsoft with a link to your validator app, your test account information and manual test results document. Microsoft will complete a final test pass before launch.

Please contact Microsoft for directions to set the Office for iOS apps into a test mode that can be used for testing the below scenarios.

Note:

- "Company app/application" refers to your app.
- "Company service" refers to your service.
- Where it refers to "Office", please substitute Word, Excel and PowerPoint app. The tests should be re-run against each Office application.
- Please note the version of the Office app you tested against. This information can be found under *Settings → [App] → Version*

15.1 1 - Open from company app - first install

This test verifies the flow of using Office for the first time from the Company app. Repeat for each supported file type and each Office app.

1. Start with a fresh install of Company app. Ensure Office is not installed.
2. Boot up company app and login.

RESULT: General promotion for Office should be shown the first time after upgrading the company app.

1. Browse to a supported file type

RESULT: "Open with Microsoft [app]" promotion, drawing attention to control and enabling open with Office once per first open for each supported file type. "Open with Microsoft [app]" should be top choice if multiple choices are available. If a list is not shown, Office should be the default app for opening the file.

1. Activate control to open in Office

RESULT: User should be sent to app store page for the corresponding app.

1. Install the Office app.

2. After installation, go back to the Company app and activate the control to open in Office again.

RESULT: Office should start. User should be prompted for credentials to Company Service.

1. Enter credentials for company service.

RESULT: File should open in read only mode.

1. Click *Sign In* and sign in with a free Microsoft Account.

RESULT: File should open in edit mode if user is a consumer and read-only mode if the user is a commercial user.

1. Make changes (you will need to sign in with a subscription account for testing commercial user)
2. Click *Back (<-)*
3. Click *Open*

RESULT: Confirm company service is shown as a place.

15.2 2 - Open from Office for iOS - fresh install

This test verifies the flow of using Company Service for the first time from Office.

1. Launch a fresh install of Office.
2. Go through the First Run Experience.
3. Skip *Sign In*.
4. Go to *Open* → *Add a Place*

RESULT: Company service shows up. Verify the name and icon of your service.

1. Select your Company Service.
2. Enter credentials.

RESULT: Root folder should show.

1. Browse around the folder structure in your service.

RESULT: Browse works as expected.

1. Open a file from *Browse*.

RESULT: File should open in read-only mode.

1. Click *Sign In* and sign in with a free Microsoft Account.

RESULT: File should open in edit mode if user is a consumer and read-only mode if the user is a commercial user.

1. Make changes (you will need to sign in with a subscription account for testing commercial user).
2. Click *Back (<-)*.
3. Click *Open*.

RESULT: File should have the previously saved changes. Ensure changes are being saved on Company service.

15.3 3 - Open from company app - repeat usage

Repeat test 1 except with company service already added (i.e. from previous usage).

15.4 4 - Open from Office for iOS - repeat usage

Repeat test 2 except with company service already added (i.e. from previous usage).

15.5 5 - Save as/duplicate

Verify ability to duplicate a file to Company Service, both by adding a new place and using an existing place.

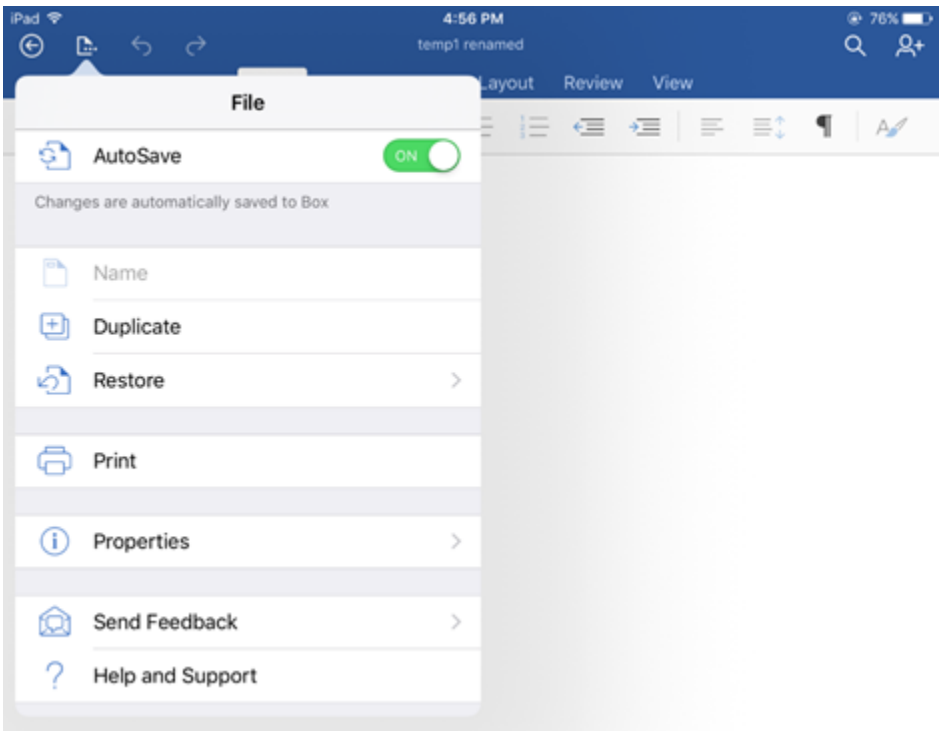


Figure 15.1: A screenshot that shows the document actions in Office for iOS.

15.6 6 - Create new [name]

Verify ability to create a new file saved to Company Service, both by adding a new place and using an existing place.

15.7 7 - Verify licensing

Verify editing a file for a commercial user requires O365 subscription or else it opens read only.

Important: Go to *Settings* → *[Microsoft App]* → *Reset Word* → *Delete Sign-In Credentials* and restart Office before doing this test.

15.8 8 - OAuth login page

Verify there is a link to the company's privacy statement on the company's login page when the user adds the company service as a place.

Verify login page fits in window for various iPad and iPhone sizes.

15.9 9 - Verify file properties

Verify file properties from *Recent* and from opened file. When opening the properties from the *Recent* tab or the *Open* tab, the fields Author, Created, Modified By and Company will be empty.

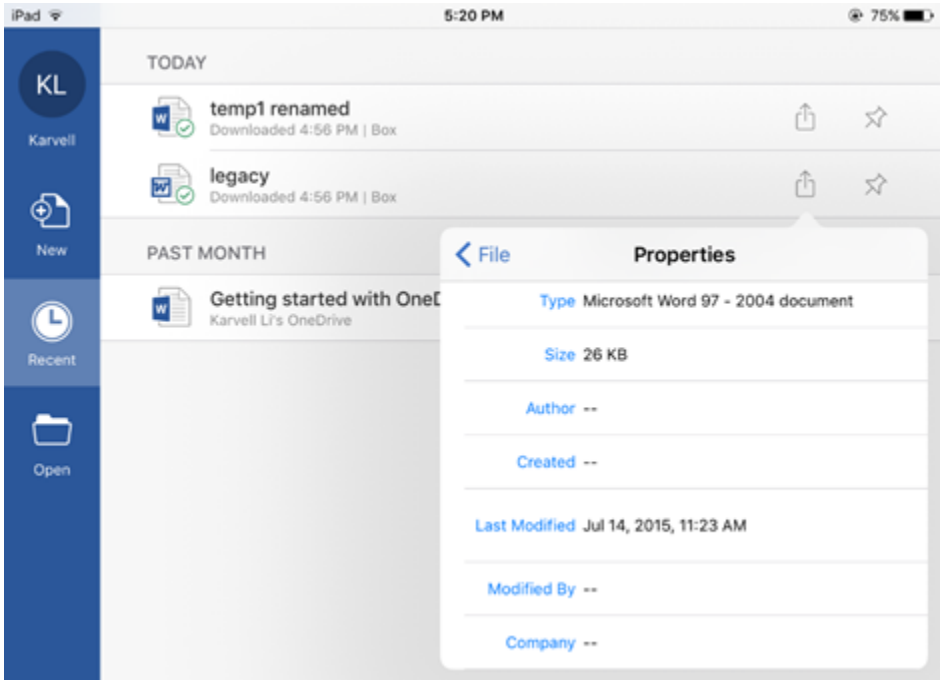


Figure 15.2: Properties view from Recent

15.10 10 - Change passwords

This test verifies the flow of using Company Service after the user changed passwords.

Note: This test changes based on how the Company Service handles authentication and refresh/access tokens. If you invalidate the access and refresh token after the user changes password, run this test. You can adapt this test to ensure the Office app is handling refresh and access tokens correctly.

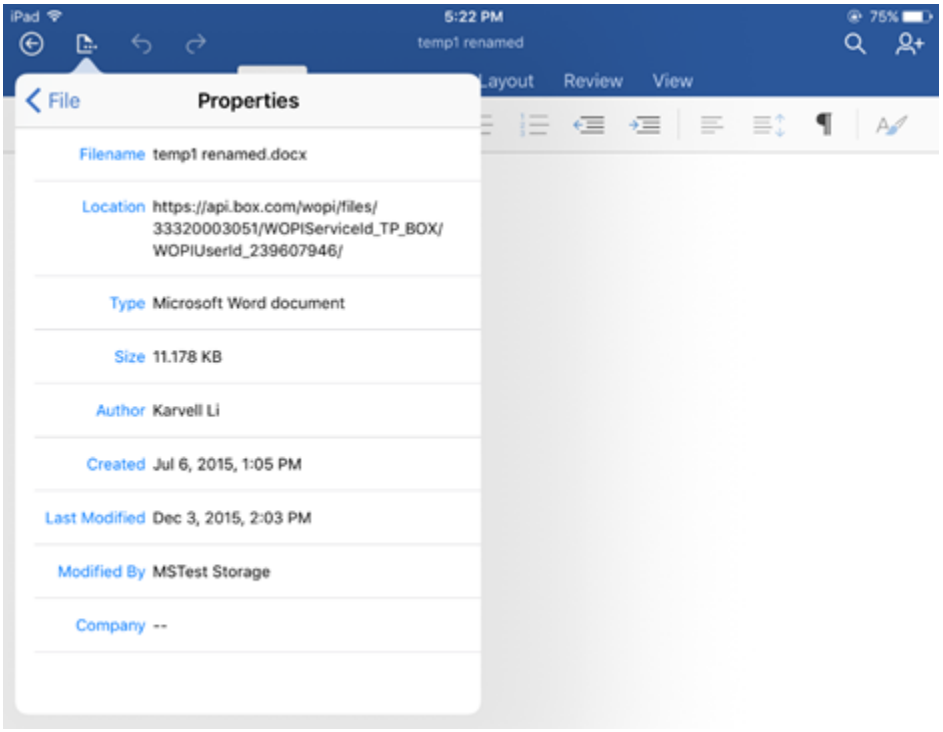


Figure 15.3: Properties view from within a document

1. Launch a fresh install of Office.
2. Go through the First Run Experience.
3. Skip Sign In.
4. Go to *Open* → *Add a Place*
5. Select your Company Service.
6. Enter credentials.
7. Browse around the folder structure in your service.
8. Open a file from *Browse*.
9. Click sign in and sign in with a free Microsoft Account.
10. Make changes (you will need to sign in with a subscription account for testing commercial user)
11. Click *Back*
12. On the Company Service app, change the password of the user.
13. Open the Office for iOS app and browse to the Company Service and open a file.

RESULT: You should be prompted to enter credentials again.

CHAPTER

SIXTEEN

USING THE VALIDATOR APPLICATION TO VALIDATE YOUR WOPI IMPLEMENTATION

Since WOPI is used in both the Office for the web and Office for iOS integrations, you can verify your WOPI implementation by following the instructions at WOPI Validation application.

If you do not integrate with Office for the web, you can still verify your WOPI implementation using the above instructions by building a minimal host page and setting the VALIDATOR_TEST_CATEGORY to `OfficeNativeClient`. This will run only the tests that are necessary for Office for iOS integration.

The validator application will not be able to verify the end user experience entirely, so manual validation must also be done.

Note: You will not be able to invoke any WOPI actions successfully unless your WOPI domain has been added to the WOPI domain allow list.

CHAPTER

SEVENTEEN

LAUNCH REQUIREMENTS

CHAPTER

EIGHTEEN

FREQUENTLY ASKED QUESTIONS

18.1 Why does calling the Token Issuance URL return "The request was aborted: Could not create SSL/TLS secure channel"?

If TLS 1.0, TLS 1.1, and TLS 1.2 in Advanced settings are enabled in the browser, but the app is failing to connect to the Token Issuance URL, this can be caused by an issue with the load balancer changing the SSL stream between the client and server. Please configure the load balancer to just load balance TCP/443 instead of using the built-in SSL protocol load balancing. By changing the load balancing to just load balance TCP/443, this should keep the load balancer from breaking the SSL communications.

See also:

For more information, please see this article.

18.2 I have already integrated my app with Office for the web. What additional work is there to integrate with Office for iOS?

See *Differences between Office for iOS, Office for Android and Office for the web*

18.3 What file types are supported in Office for iOS?

18.3.1 Excel

Modern	Legacy
`xlsx`	`xls`
`xltx`	`csv`
`xlsb`	`ods`
`xlsm`	

18.3.2 PowerPoint

Modern	Legacy
pptx	ppt
ppsx	pps
potm	pot
potx	odp
ppsm	
pptm	
thmx	

18.3.3 Word

Modern	Legacy
docx	doc
docm	dot
dotm	odt
dotx	

Note: Modern file type documents can be edited directly.

When users attempt to edit documents in the legacy formats, they will be prompted to convert the file to a modern format. Office for iOS will convert the legacy document to the modern format and save it as a new document alongside the existing document.

CHAPTER

NINETEEN

KNOWN ISSUES

19.1 Office for iOS will fail to load files with invalid characters in the file name, folder names or user ID

Office for iOS will fail to load files if there are invalid characters in certain CheckFileInfo or CheckContainerInfo properties. The following characters are not allowed:

```
\/:*?"<>|#{}^[]`%
```

These properties are affected:

CheckFileInfo

- BaseFileName
- UserId
- OwnerId

CheckContainerInfo

- Name

To work around this issue, hosts should replace these characters with −.

CHAPTER

TWENTY

GLOSSARY

INDEX

R