WOPI REST Documentation

Release 2016.01.27

unknown

Aug 04, 2020

MICROSOFT OFFICE WOPI INTEGRATIONS

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.

The Web Application Open Platform Interface Protocol (WOPI) defines a set of operations that enables a client to access and change files stored by a server. WOPI is a REST-based protocol. It defines a set of *REST endpoints* as well as operations that can be executed by issuing HTTP requests to those endpoints.

The WOPI protocol is used by Office applications, such as Office for the web, to view and edit files that are stored in a cloud service. Thus, Office for the web is a *WOPI client*, while the cloud service that stores the files is a *WOPI server*, often referred to as a *host*.

This documentation is a reference for all of the defined WOPI operations. However, not all operations must be implemented. WOPI is modular and a WOPI server need only implement the operations required by the WOPI client(s) the server intends to integrate with. To learn more about the specific requirements for integrating with Office for the web or Office for iOS and Android, see the sections below:

- Using the WOPI protocol to integrate with Office for the web
- Integrating with Office for iOS and Android

CHAPTER

ONE

KEY CONCEPTS

The following concepts are referred to extensively within this documentation, and understanding them is important to understanding the requirements for integration with WOPI clients such as Office for the web and Office for iOS.

File ID A File ID is a string that represents a file or folder being operated on via WOPI operations. A host must issue a unique ID for any file used by a WOPI client. The client will, in turn, include the file ID when making requests to the WOPI host. Thus, a host must be able to use the file ID to locate a particular file.

A file ID must:

- Represent a single file.
- Be a URL-safe string because IDs are passed in URLs, principally via the *WOPISrc* parameter.
- Remain the same when the file is edited.
- Remain the same when the file is moved or renamed.
- Remain the same when any ancestor container, including the parent container, is renamed.
- In the case of shared files, the ID for a given file must be the same for every user that accesses the file.

Note that the file ID is not provided to a WOPI client directly. Rather, it is passed as part of the *WOPISrc* value.

Office for the web Tip

See Action URLs and WOPI_SOURCE for more information about how the file ID should be passed to Office for the web.

Access token An access token is a string used by the host to determine the identity and permissions of the issuer of a WOPI request.

The WOPI host that stores the file has the information about user permissions, not the WOPI client. For this reason, the WOPI host must provide an access token that the client will then pass back to it on subsequent WOPI requests. When the WOPI host receives the token, it either validates it, or responds with an appropriate HTTP status code if the token is invalid or unauthorized.

Office for the web Tip

In Office for the web, an access token is generated by the host and passed to the client using the `access_token` parameter before the first WOPI request. See Action URLs and Passing access tokens securely for more information about how access tokens should be passed to Office for the web.

A WOPI client requires no understanding of the format or content of the token; the WOPI client simply includes it in WOPI requests and expects the host to validate it. However, WOPI access tokens must adhere to the following guidelines:

- Access tokens must be scoped to a single user and resource combination. A WOPI client will never assume that an access token issued for a particular user/resource combination is valid for a different user/resource combination.
- Access tokens must be valid for the *user permissions* that are provided by the host in the *CheckFileInfo* response. For example, if the `view` action is invoked, and the *UserCanWrite* property is set to `true` in the *CheckFileInfo* response, then the client may re-use that token when transitioning to edit mode. Thus, a WOPI client will expect that any access token is valid for operations that the user has permissions to perform. If a host wishes to issue access tokens that are more narrowly scoped, then the *user permissions properties* in the *CheckFileInfo* response must reflect the permissions that the token provides.
- Access tokens should expire (become invalid) automatically after a period of time. Hosts can use the *access_token_ttl* property to specify when an access token expires.

Important: WOPI clients expect that an access token will remain valid until it expires (as indicated by the *access_token_ttl* value). Hosts should not revoke access tokens as a standard part of their operations; tokens should only be revoked if a user's permissions have changed or been revoked.

If hosts revoke access tokens regularly, users may experience strange behavior depending on the WOPI client. For example, in some cases Office for the web will time out a user's session and present them with a dialog asking if they'd like to refresh their session. If the access token is not yet expired based on the *access_token_ttl*, Office for the web will refresh the session using the existing access token, assuming that it is still valid.

There are a number of cases like this, and WOPI clients regularly make such assumptions. For this reason, access tokens should remain valid until their expiry.

Important: Note that WOPI clients are not required to pass the *access token* in the Authorization header, but they must send it as a URL parameter in all WOPI operations. Thus, for maximum compatibility, WOPI hosts should either use the URL parameter in all cases, or fall back to it if the Authorization header is not included in the request.

---

access_token_ttl The access_token_ttl property tells a WOPI client when an access token expires, represented as the number of milliseconds since January 1, 1970 UTC (the date epoch in JavaScript). Despite its misleading name, it does *not* represent a duration of time for which the access token is valid. The access_token_ttl is never used by itself; it is always attached to a specific access token.

Office for the web Tip

To prevent data loss for users, Office for the web will prompt users to save and refresh their sessions if the access token for their session is close to expiring. In order to do this, Office for the web needs to know when the access token will expire, which it determines based on the access_token_ttl value.

Hosts can set the access_token_ttl value to `0`. This will effectively tell the client that the token expiry is either infinite or unknown. In this case, clients may disable any UI prompting users to refresh their sessions. This can lead to unexpected data loss due to access token expiry, so specifying a value for access_token_ttl is strongly recommended.

Note: Future updates to the WOPI protocol may rename this parameter so its name is less confusing.

Lock A lock is a conceptual construct that is associated with a file. Locks serve two purposes in WOPI:

1. First, a lock prevents anyone that does not have a valid lock ID from making changes to a file. A WOPI client will lock files prior to editing them to prevent other entities from changing the file while the client is also editing them.
2. A lock is also used to store a small bit of temporary data associated with a file. This metadata is called the *lock ID* and is a string with a maximum length of 1024 ASCII characters (see *note on lock ID lengths*). WOPI clients can use this metadata for a variety of purposes, but hosts do not need any knowledge or understanding of the contents of the lock ID. Hosts must treat it as an opaque string.

Therefore, WOPI locks must:

- Be associated with a single file.
- Contain a lock ID of maximum length 1024 ASCII characters.
- Prevent all changes to that file unless a proper lock ID is provided.
- Expire after 30 minutes unless refreshed (see *RefreshLock*).
- *Not* be associated with a particular user.

All WOPI operations that modify files, such as *PutFile*, will include a lock ID as a parameter in their request. Usually the ID will be passed in the **X-WOPI-Lock** request header (but not always; *UnlockAndRelock* is an exception). WOPI requires that hosts compare the lock ID passed in a WOPI request with the lock ID currently on a file and respond appropriately when the lock IDs do not match. In particular, WOPI clients expect that when a lock ID does *not* match the current lock, the host will send back the current lock ID in the **X-WOPI-Lock** response header. This behavior is critical, because WOPI clients will use the current lock ID in order to determine what further WOPI calls to make to the host.

It is important to note that WOPI locks are *not* user-owned. In other words, a WOPI client may execute lock-related operations using multiple access tokens, and hosts are expected to execute those operations as long as they are valid as described in this documentation. For example, a WOPI host may receive a *Lock* call with an access token that belongs to User A. The host may later receive an *Unlock* call with an access token that belongs to User B. As long as User B has rights to edit the file, and the **X-WOPI-Lock** request header matches the lock ID, the *Unlock* request should be honored.

WOPI locks must automatically expire after 30 minutes if not renewed by the WOPI client. This ensures that files do not stay locked indefinitely in error cases. Since locks are client-controlled from a protocol perspective (that is, the WOPI client sets and manages the lock) and clients can be unreliable, the WOPI host must expire the locks in such cases.

Office for the web Tip

WOPI defines a *GetLock* operation. However, Office for the web does not use it in all cases, even if the host indicates support for the operation using the *SupportsGetLock* property in *CheckFileInfo*. Instead, Office for the web will sometimes execute lock-related operations on files with missing or known incorrect lock IDs and expects the host to provide the current lock ID in its WOPI response. Typically the *Unlock* and *RefreshLock* operations are used for this purpose, but other lock-related operations may be used.

The specific conditions for each response are covered in the documentation for each of the following lock-related WOPI operations:

- *Lock*
- *RefreshLock*
- *Unlock*
- *UnlockAndRelock*
- *PutFile*

Note: Lock ID lengths are currently less than 256 ASCII characters. However, we anticipate requiring longer lock IDs to support future WOPI integration scenarios, so we have increased the limit to 1024 ASCII characters. Hosts must indicate that they support lock IDs of this length using the *SupportsExtendedLockLength* property in *CheckFileInfo*.

Share URL A Share URL is a URL to a webpage that is suitable for viewing a shared WOPI file or container. The URL should be appropriate for being launched in a web browser, but the experience is defined by the host. For example, the host may choose to have the URL navigate to the host's browse experience or to a preview of the file using Office for the web or another file previewer.

A host may support different types of Share URLs that may be used for different purposes. For example, a particular Share URL type may not allow users to edit the file by using the Share URL. The list of possible types are defined under the *SupportedShareUrlTypes* property.

WOPISrc The WOPISrc (*WOPI Source*) is the URL used to execute WOPI operations on a file. It is a combination of the *Files endpoint* URL for the host along with a particular *file ID*. The WOPISrc does *not* include an *access token*.

For example, a WopiSrc might look like this:

```
https://wopi.contoso.com/wopi/files/abcdef0123456789
```

The WOPISrc is needed beyond just a file ID so that a WOPI client can know what URL to call back to when executing WOPI operations on a file. In practice, the WOPISrc and a *file ID* are synonymous, since WOPI client typically work with the WOPISrc itself, not the raw *file ID*.

Office for the web Tip

See WOPI_SOURCE for more details on how the WOPISrc is passed to Office for the web.

Container Not yet documented.

Root Container Not yet documented.

Ecosystem Not yet documented.

Bootstrapper Not yet documented.

CHAPTER

TWO

WOPI REST ENDPOINTS

A WOPI host needs to provide some information about the files it stores, as well as the binary contents of those files. Because WOPI is a REST-based callback interface, this information is provided via specific URLs. A WOPI host provides a small REST API around its files. WOPI clients such as Office for the web then use those REST API to work with the files.

Not all operations and endpoints are required. All actions require the *CheckFileInfo* and *GetFile* operations.

Each WOPI endpoint supports a number of operations specified by the caller in the **X-WOPI-Override** request header. Parameters for each operation are also passed in HTTP headers, which all begin with `X-WOPI-`. This way, executing a WOPI operation is as simple as issuing a request to the appropriate REST endpoint and passing appropriate HTTP header values with the request.

See also:

Standard WOPI request and response headers

2.1 Endpoint URLs

All WOPI host endpoints must be located at a URL that *starts* with `/wopi`. For example, all of the following URLs are **valid** URLs for the *File contents endpoint* for a file with the ID `abc123`:

- `https://api.contoso.com/modules/wopi/files/abc123/contents`
- `https://test.wopi.contoso.com/wopi_test/files/abc123/contents`

However, the following URLs are **not valid**:

- `https://api.contoso.com/api_wopi/files/abc123/contents` (invalid because the endpoint URL does not *start* with `/wopi`)
- `https://api.contoso.com/files/abc123/contents` (invalid because the endpoint URL does not *start* with `/wopi`)
- `https://test.wopi.contoso.com/officeonline/files/abc123/contents` (invalid because the endpoint URL does not *start* with `/wopi`)
- `https://wopi.contoso.com/wopi/files/ids/abc123/contents` (invalid because the endpoint URL contains `/ids`, which is not permitted)

2.2 Files endpoint

The Files endpoint provides access to file-level operations.

URL `/wopi/files/(file_id)`

The following operations are exposed through this endpoint:

- *CheckFileInfo*
- *PutRelativeFile*
- *Lock*
- *Unlock*
- *RefreshLock*
- *UnlockAndRelock*
- *DeleteFile*
- *RenameFile*

2.3 File contents endpoint

The File contents endpoint provides access to retrieve and update the contents of a file.

URL `/wopi/files/(file_id)/contents`

The following operations are exposed through this endpoint:

- *GetFile*
- *PutFile*

2.4 Containers endpoint

URL `/wopi/containers/(container_id)`

The following operations are exposed through this endpoint:

- *CheckContainerInfo*
- *CreateChildContainer*
- *CreateChildFile*
- *DeleteContainer*
- *EnumerateAncestors (files)*
- *EnumerateChildren (containers)*
- *RenameContainer*

2.5 Ecosystem endpoint

The Ecosystem endpoint serves as a bridge for WOPI clients that do not have a File or Container ID that they are operating on.

URL `/wopi/ecosystem`

The following operations are exposed through this endpoint:

- *CheckEcosystem*

- *GetFileWopiSrc (ecosystem)*
- *GetRootContainer (ecosystem)*

2.6 Bootstrapper endpoint

The following operations are exposed through this endpoint:

- *Bootstrap*
- *GetNewAccessToken*
- *Shortcut operations*

CHAPTER

THREE

STANDARD WOPI REQUEST AND RESPONSE HEADERS

GET /wopi/

All WOPI requests may contain the following request and response headers. Note that individual WOPI operations may send additional request headers or require additional response headers. These unique headers are described in the documentation for each WOPI operation.

Tip: HTTP header names are *case-insensitive*. See **RFC 7230#section-3.2** for more information.

Request Headers

- Authorization – The **string** value `Bearer <token>` where `<token>` is the *access token* for the request.

  Important: Note that WOPI clients are not required to pass the *access token* in the Authorization header, but they must send it as a URL parameter in all WOPI operations. Thus, for maximum compatibility, WOPI hosts should either use the URL parameter in all cases, or fall back to it if the Authorization header is not included in the request.

- *X-Request-ID* – A **string** that the host should log when logging server activity to correlate that request with a specific WOPI call to the host.
- *X-WOPI-AppEndpoint* – A **string** used to indicate the endpoint of the WOPI client sending the request. This is typically used to indicate geographic location, datacenter, etc. This string must not be used for anything other than logging.
- *X-WOPI-RequestingApplication* – A **string** used to indicate the WOPI client sending the request. This string must not be used for anything other than logging.
- *X-WOPI-ClientVersion* – A **string** that the host should log indicating the version of the WOPI client making the request. There is no standard for how this string is formatted, and it must not be used for anything other than logging.
- *X-WOPI-CorrelationId* – A **string** that the host should log when logging server activity to correlate that activity with WOPI client activity.

  Office for the web Tip

  See Troubleshooting interactions with Office for the web for more information on how this ID is used in Office for the web.

- *X-WOPI-DeviceId* – A **string** that the host should log indicating the ID of the device making the request. This string must not be used for anything other than logging.
- *X-WOPI-SessionId* – A **string** that the host should log to correlate WOPI client activity within a session. This string must not be used for anything other than logging.
- *X-WOPI-MachineName* – A **string** indicating the name of the WOPI client machine making the request. This string must not be used for anything other than logging.
- *X-WOPI-PerfTraceRequested* – This header is reserved for future use.
- *X-WOPI-Proof* – A **string** representing data signed using a SHA256 (A 256 bit SHA-2-encoded [FIPS 180-2]) encryption algorithm. See Verifying that requests originate from Office for the web by using proof keys for more information regarding the use of this header value.
- *X-WOPI-ProofOld* – A **string** representing data signed using a SHA256 (A 256 bit SHA-2-encoded [FIPS 180-2]) encryption algorithm. See Verifying that requests originate from Office for the web by using proof keys for more information regarding the use of this header value.
- *X-WOPI-TimeStamp* – A **64-bit integer** that represents the number of 100-nanosecond intervals that have elapsed between 12:00:00 midnight, January 1, 0001, UTC (Coordinated Universal Time) and the UTC time of the request. This value can be set in .NET using the following C# code: `DateTime.UtcNow.Ticks`.

  See also:

  DateTime.Ticks Property

Response Headers

- Content-Type – This header should be set to a value appropriate to the type of data being included in the response. For example, when responding to a WOPI request with JSON-encoded data in the response body, the Content-Type header should be set to `application/json`. WOPI clients may ignore a response with a Content-Type header that does not match the expected type.
- *X-WOPI-HostEndpoint* – A **string** used to indicate the endpoint of the WOPI host handling the request. This is analogous to the **X-WOPI-AppEndpoint** request header and is typically used to indicate geographic location, datacenter, etc. This string must not be used for anything other than logging.
- *X-WOPI-MachineName* – A **string** indicating the name of the WOPI host server handling the request. This string must not be used for anything other than logging.
- *X-WOPI-PerfTrace* – This header is reserved for future use.
- *X-WOPI-ServerError* – A **string** indicating that an error occurred while processing the WOPI request. This header should be included in a WOPI response if the status code is 500 Internal Server Error, but may be returned on any response with a non-200 status code. The value should contain details about the error. This string must not be used for anything other than logging.
- *X-WOPI-ServerVersion* – A **string** indicating the version of the WOPI host server handling the request. There is no standard for how this string is formatted, and it must not be used for anything other than logging.

CHAPTER

FOUR

SECURITY CONSIDERATIONS

4.1 Preventing 'token trading'

Some WOPI operations, such as *GetEcosystem (files)*, *EnumerateAncestors (files)*, and *EnumerateChildren (containers)*, return URLs that must include WOPI *access tokens*. WOPI access tokens must always be treated as per-resource, per-user by a WOPI client, and they must always be expected to expire after a period of time. WOPI deliberately does not define a means for a client to refresh a WOPI access token given another WOPI access token.

However, WOPI is also deliberately designed to support navigation from a file or container to the *Ecosystem endpoint*, then back to a container or file. This means that it is possible for a client to 'refresh' their WOPI tokens indefinitely unless the WOPI host is careful when issuing new access tokens to mitigate these threats. We refer to this threat as 'token trading.'

To illustrate the token trading threat, consider the following scenario:

1. A WOPI client, on behalf of User A, is issued a WOPI access token, `TOKEN1`, for the file `Document.docx`. The token has a lifetime of 12 hours.
2. While `TOKEN1` is still valid, the client calls the *EnumerateAncestors (files)* operation using `TOKEN1` as the access token.
3. The server responds with a URL to the parent container along with a new WOPI access token for that container, `TOKEN2`.
4. The client then calls *EnumerateChildren (containers)* using `TOKEN2` as the access token.
5. The server responds with the URL for the children files, including `Document.docx`, along with a new access token for `Document.docx`, `TOKEN3`.

At this point, the client has effectively 'traded' `TOKEN1` for `TOKEN3`. If each token issued has a lifetime of 12 hours, then this means that a client can effectively refresh their access token for a particular file by going through the container hierarchy, without actually authenticating with the server using a non-WOPI authentication mechanism.

If unmitigated, this scenario greatly increases the potential damage caused by token leakage. If a malicious attacker gains access to `TOKEN1`, then they can potentially access `Document.docx` indefinitely, as long as User A still has access to it. In other words, the attacker can impersonate User A indefinitely.

In addition, an attacker could use *EnumerateAncestors (files)* and *EnumerateChildren (containers)* to trade `TOKEN1` for a token that is valid for any other document in the container hierarchy that the user has access to, then impersonate User A indefinitely to access those other documents and containers as well.

4.1.1 Mitigation

The preferred way to mitigate this threat is to create new WOPI access tokens with the same token lifetime as the WOPI access token that was used when the operation was called. In other words, in the scenario described above, the

lifetime of `TOKEN2` and `TOKEN3` should be the same as `TOKEN1`. All three tokens should expire at the same time.

Note that this should only apply when a WOPI access token is used to create another WOPI access token. In other cases where WOPI access tokens are issued, such as from *Bootstrap* or other OAuth-authenticated *Shortcut operations*, or, in the case of web-based WOPI clients, by visiting a host page that issues the access token, the host-defined default access token lifetime can apply. This is safe in those cases because there is a separate primary authentication method that is also in use. In the case of *Bootstrap*, it is an OAuth token. In the case of web-based WOPI clients, it is the host's standard web authentication system.

CHAPTER

FIVE

CHECKFILEINFO

Required for

GET /wopi/files/ (*file_id*)

The CheckFileInfo operation is one of the most important WOPI operations. This operation must be implemented for all WOPI actions. CheckFileInfo returns information about a file, a user's permissions on that file, and general information about the capabilities that the WOPI host has on the file. In addition, some CheckFileInfo properties can influence the appearance and behavior of WOPI clients.

Office for the web Tip

Some properties are not supported within the Office 365 - Cloud Storage Partner Program. These operations are marked .

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-SessionContext* – The value of the session context, if provided on the initial WOPI action URL.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

5.1 Response

The response to a CheckFileInfo call is JSON (as specified in **RFC 4627**) containing a number of properties, most of which are optional.

All optional values default to the following values based on their type:

Type	Default value
Boolean	`false`
String	The empty string
Integer/Long	Varies; see individual properties for details
Array	Empty array

Important: No properties should be set to `null`. If you do not wish to set a property, simply omit it from the response and WOPI clients will use the default value.

5.2 Required response properties

The following properties must be present in all CheckFileInfo responses:

BaseFileName The **string** name of the file, including extension, without a path. Used for display in user interface (UI), and determining the extension of the file.

OwnerId A **string** that uniquely identifies the owner of the file. In most cases, the user who uploaded or created the file should be considered the owner.

> **Important:** This ID is subject to uniqueness and consistency requirements. See *Requirements for user identity properties* for more information.

Size The size of the file in bytes, expressed as a **long**, a 64-bit signed integer.

UserId A **string** value uniquely identifying the user currently accessing the file.

> **Important:** This ID is subject to uniqueness and consistency requirements. See *Requirements for user identity properties* for more information.

Version The current version of the file based on the server's file version schema, as a **string**. This value must change when the file changes, and version values must never repeat for a given file.

> **Important:** This value must be a **string**, even if numbers are used to represent versions.

5.2.1 Requirements for user identity properties

Hosts use the *OwnerId* and *UserId* properties to provide user ID data to WOPI clients. User identity properties are intended for telemetry purposes, and thus should not be shown in any WOPI client UI. These properties must meet the following requirements:

- Unique to a single user.
- Consistent over time. For example, if a particular user uses a WOPI client to view a document on Monday, then returns and views another document on Tuesday, the value of the user identity properties should match.

Important: Hosts should avoid the following characters in user identity properties in order to support the widest range of WOPI clients:

```
<>"#{}^[]`\/
```

5.3 WOPI host capabilities properties

The WOPI host capabilities properties indicate to the WOPI client what WOPI capabilities that the host supports for a file. All of these properties are optional and thus default to `false`; hosts should set them to `true` if their WOPI implementation meets the requirements for a particular property.

Important: If a WOPI server sets any capabilities properties to `true`, WOPI clients will assume that all of the operations represented by that capability property are supported. Thus, a WOPI host must implement all operations represented by that capability property if they set the property to `true`.

SupportedShareUrlTypes An **array** of strings containing the *Share URL* types supported by the host.

These types can be passed in the **X-WOPI-UrlType** request header to signify which Share URL type to return for the *GetShareUrl (files)* operation.

Possible Values:

ReadOnly This type of Share URL allows users to view the file using the URL, but does not give them permission to edit the file.

ReadWrite This type of Share URL allows users to both view and edit the file using the URL.

SupportsCobalt A **Boolean** value that indicates that the host supports the following WOPI operations:

- ExecuteCellStorageRequest
- ExecuteCellStorageRelativeRequest

SupportsContainers A **Boolean** value that indicates that the host supports the following WOPI operations:

- *CheckContainerInfo*
- *CreateChildContainer*
- *CreateChildFile*
- *DeleteContainer*
- *DeleteFile*
- *EnumerateAncestors (containers)*
- *EnumerateAncestors (files)*
- *EnumerateChildren (containers)*
- *GetEcosystem (containers)*
- *RenameContainer*

Tip: SupportsContainers is a superset of *SupportsDeleteFile*. However, WOPI hosts should explicitly return `true` for both properties if SupportsContainers is `true`.

SupportsDeleteFile A **Boolean** value that indicates that the host supports the *DeleteFile* operation.

SupportsEcosystem A **Boolean** value that indicates that the host supports the following WOPI operations:

- *CheckEcosystem*
- *GetEcosystem (containers)*
- *GetEcosystem (files)*
- *GetRootContainer (ecosystem)*

SupportsExtendedLockLength A **Boolean** value that indicates that the host supports lock IDs up to 1024 ASCII characters long. If not provided, WOPI clients will assume that lock IDs are limited to 256 ASCII characters.

Important: While the 256 ASCII character lock length is currently sufficient, longer lock IDs will likely be required to support future scenarios, so we recommend hosts support extended lock lengths as soon as possible. See *lock ID lengths* for more information.

SupportsFolders A **Boolean** value that indicates that the host supports the following WOPI operations:

- CheckFolderInfo,
- EnumerateChildren (folders)
- *DeleteFile*

SupportsGetFileWopiSrc A **Boolean** value that indicates that the host supports the *GetFileWopiSrc (ecosystem)* operation.

SupportsGetLock A **Boolean** value that indicates that the host supports the *GetLock* operation.

SupportsLocks A **Boolean** value that indicates that the host supports the following WOPI operations:

- *Lock*,
- *Unlock*
- *RefreshLock*
- *UnlockAndRelock* operations for this file.

SupportsRename A **Boolean** value that indicates that the host supports the *RenameFile* operation.

SupportsUpdate A **Boolean** value that indicates that the host supports the following WOPI operations:

- *PutFile*
- *PutRelativeFile*

SupportsUserInfo A **Boolean** value that indicates that the host supports the *PutUserInfo* operation.

New in version 2015.08.03.

5.4 User metadata properties

The following properties are used to provide additional information about users.

IsAnonymousUser A **Boolean** value indicating whether the user is authenticated with the host or not. Hosts should always set this to `true` for unauthenticated users, so that clients are aware that the user is anonymous.

When setting this to `true`, hosts can choose to omit the *UserId* property, but must still set the *OwnerId* property.

New in version 2017.02.15.

IsEduUser A **Boolean** value indicating whether the user is an education user or not.

New in version 2016.01.27.

LicenseCheckForEditIsEnabled A **Boolean** value indicating whether the user is a business user or not.

See also:

Supporting document editing for business users

UserFriendlyName A **string** that is the name of the user, suitable for displaying in UI.

UserInfo A **string** value containing information about the user. This string can be passed from a WOPI client to the host by means of a *PutUserInfo* operation. If the host has a UserInfo string for the user, they must include it in this property. See the *PutUserInfo* documentation for more details.

New in version 2015.08.03.

5.5 User permissions properties

A WOPI client must always assume that users have limited permissions to documents. If a host does not set the appropriate user permissions properties, users will not be able to perform operations such as editing documents using a WOPI client.

Ultimately, the host has final control over whether WOPI operations attempted by the client should succeed or fail based on the *access token* provided in the WOPI request. Thus, these properties do not act as an authorization mechanism. Rather, these properties help WOPI clients tailor their UI and behavior to the specific permissions a user has. For example, a WOPI client can hide file renaming UI if the *UserCanRename* property is `false`.

However, a WOPI client expects that even if that UI were somehow made available to a user without appropriate permissions, the WOPI *RenameFile* request would fail since the host would determine the action was not permissible based on the *access token* passed in the request.

Note that there is no property that indicates the user has permission to read/view a file. This is because WOPI requires the host to respond to any WOPI request, including *CheckFileInfo*, with a 401 Unauthorized or 404 Not Found if the access token is invalid or expired.

ReadOnly A **Boolean** value that indicates that, for this user, the file cannot be changed.

RestrictedWebViewOnly A **Boolean** value that indicates that the WOPI client should restrict what actions the user can perform on the file. The behavior of this property is dependent on the WOPI client.

UserCanAttend A **Boolean** value that indicates that the user has permission to view a broadcast of this file.

UserCanNotWriteRelative A **Boolean** value that indicates the user does not have sufficient permission to create new files on the WOPI server. Setting this to `true` tells the WOPI client that calls to *PutRelativeFile* will fail for this user on the current file.

UserCanPresent A **Boolean** value that indicates that the user has permission to broadcast this file to a set of users who have permission to broadcast or view a broadcast of the current file.

UserCanRename A **Boolean** value that indicates the user has permission to rename the current file.

UserCanWrite A **Boolean** value that indicates that the user has permission to alter the file. Setting this to `true` tells the WOPI client that it can call *PutFile* on behalf of the user.

5.6 File URL properties

Hosts can return a number of URLs for the file that WOPI clients may navigate to in various scenarios.

Office for the web Tip

These properties can be used to customize the user experience of the Office for the web applications. See Customizing Office for the web using CheckFileInfo properties for more information about how each property is used in Office for the web.

CloseUrl A URI to a web page that the WOPI client should navigate to when the application closes, or in the event of an unrecoverable error.

DownloadUrl A user-accessible URI to the file intended to allow the user to download a copy of the file.

> **Important:** This URI should directly download the file. In other words, WOPI clients expect that when directing users to this URL, the file will be immediately downloaded. This URL should not direct the user to some separate UI to download the file.

> **Important:** This URI should always provide the most recent version of the file.

FileEmbedCommandUrl A URI to a location that allows the user to create an embeddable URI to the file.

FileSharingUrl A URI to a location that allows the user to share the file.

FileUrl A URI to the file location that the WOPI client uses to get the file. If this is provided, the WOPI client may use this URI to get the file instead of a *GetFile* request. A host might set this property if it is easier or provides better performance to serve files from a different domain than the one handling standard WOPI requests. WOPI clients must not add or remove parameters from the URL; no other parameters, including the *access token*, should be appended to the FileUrl before it is used.

> **Important:** The FileUrl is meant as a performance enhancement. The *GetFile* operation must still be supported for the file even when the FileUrl property is provided.

> **Note:** Requests to the *FileUrl* can not be signed using proof keys. The FileUrl is used exactly as provided by the host, so it does not necessarily include the access token, which is required to construct the expected proof.

FileVersionUrl A URI to a location that allows the user to view the version history for the file.

New in version 2017.02.15.

HostEditUrl A URI to a host page that loads the `edit` WOPI action.

HostEmbeddedViewUrl A URI to a web page that provides access to a viewing experience for the file that can be embedded in another HTML page. This is typically a URI to a host page that loads the `embedview` WOPI action.

HostViewUrl A URI to a host page that loads the `view` WOPI action. This URL is used by Office Online to navigate between view and edit mode.

SignoutUrl A URI that will sign the current user out of the host's authentication system.

See also:

SignInUrl

5.7 PostMessage properties for web-based WOPI clients

CheckFileInfo supports a number of properties that can be used by web-based WOPI clients such as Office for the web to customize the user interface and experience when using those clients. See PostMessage properties for more information on these properties and how to use them.

5.8 Breadcrumb properties

Breadcrumb properties are used by some WOPI clients to display breadcrumb-style navigation elements within the WOPI client UI.

BreadcrumbBrandName A **string** that indicates the brand name of the host.

BreadcrumbBrandUrl A URI to a web page that the WOPI client should navigate to when the user clicks on UI that displays *BreadcrumbBrandName*.

BreadcrumbDocName A **string** that indicates the name of the file. If this is not provided, WOPI clients may use the *BaseFileName* value.

BreadcrumbFolderName A **string** that indicates the name of the container that contains the file.

BreadcrumbFolderUrl A URI to a web page that the WOPI client should navigate to when the user clicks on UI that displays *BreadcrumbFolderName*.

5.9 Other miscellaneous properties

AllowAdditionalMicrosoftServices A **Boolean** value that indicates a WOPI client may connect to Microsoft services to provide end-user functionality.

New in version 2016.06.27.

Office for the web Tip

In Office for the web, setting this property to `true` enables the following features:

Bing spelling and proofing Office for the web will use the Bing Spell Check API to provide better spelling and proofing suggestions for supported languages.

Smart Lookup Office for the web will use Bing to power the Smart Lookup feature, which provides quick access to definitions, Wiki articles, and top related searches from the web.

Additional features might be added in the future.

AllowErrorReportPrompt A **Boolean** value that indicates that in the event of an error, the WOPI client is permitted to prompt the user for permission to collect a detailed report about their specific error. The information gathered could include the user's file and other session-specific state.

New in version 2017.06.01.

Tip: This value should be omitted (or explicitly set to `false`) if no additional collection should be done on errors, or if the user has opted out of telemetry collection.

AllowExternalMarketplace A **Boolean** value that indicates a WOPI client may allow connections to external services referenced in the file (for example, a marketplace of embeddable JavaScript apps).

ClientThrottlingProtection

A **string** value offering guidance to the WOPI client as to how to differentiate client throttling behaviors between the user and documents combinations from the WOPI host. Under times of stress, the WOPI client may choose to make use of this field to vary the level of impact of client side throttling behaviors within the set of active host documents. If the WOPI client chooses to differentiate throttling of client behaviors that are not necessarily tied to WOPI calls to the host, it may apply the most reduced quality of service to the LeastProtected document/users and the least reduced quality of service to the MostProtected documents/users. As in the case of *RequestedCallThrottling*, it is advised that hosts sharing this value between responses for distinct users of the same document at any given time may yield more deterministic results from the clients.

Possible Values:

MostProtected The most protected documents/users.

Protected The documents/users that should be protected more than the average ones.

Normal The documents/users with the standard level of protection from throttling.

LessProtected The documents/users that can be throttled more heavily than a typical than the average ones.

LeastProtected The least proteced documents/users.

If no value is specified, or an invalid value is provided, the default behavior is to assume the WOPI host intended the value of None.

CloseButtonClosesWindow A **Boolean** value that indicates the WOPI client should close the window or tab when the user activates any *Close* UI in the WOPI client.

Office for the web Tip

This property may not behave as expected in Office for the web.

See also:

Why should I avoid using the CloseButtonClosesWindow property in Office for the web?

CopyPasteRestrictions

A **string** value indicating whether the WOPI client should disable Copy and Paste functionality within the application. The default is to permit all Copy and Paste functionality, i.e. the setting has no effect.

Possible Values:

BlockAll Copy and Paste are completely disabled within the application.

CurrentDocumentOnly Copy and Paste are enabled but content can only be copied and pasted within the file currently open in the application.

Any values other than those listed above must be ignored by the WOPI client.

Office for the web Tip

This property is only respected by Excel for the web. It has no effect in PowerPoint for the web or Word for the web.

DisablePrint A **Boolean** value that indicates the WOPI client should disable all print functionality.

DisableTranslation A **Boolean** value that indicates the WOPI client should disable all machine translation functionality.

FileExtension A **string** value representing the file extension for the file. This value must begin with a `.`. If provided, WOPI clients will use this value as the file extension. Otherwise the extension will be parsed from the *BaseFileName*.

Tip: While this property is not required, hosts should set it rather than relying on the *BaseFileName* parsing.

FileNameMaxLength An **integer** value that indicates the maximum length for file names that the WOPI host supports, excluding the file extension. The default value is 250. Note that WOPI clients will use this default value if the property is omitted or if it is explicitly set to `0`.

Office for the web Tip

This property is optional; however, hosts wishing to enable file renaming within Office for the web should verify that the default value is appropriate and set it accordingly if not. See the *RenameFile* operation for more details.

LastModifiedTime A **string** that represents the last time that the file was modified. This time must always be a must be a UTC time, and must be formatted in ISO 8601 round-trip format. For example, `"2009-06-15T13:45:30.0000000Z"`.

RequestedCallThrottling

A **string** value indicating whether the WOPI host is experiencing capacity problems and would like to reduce the frequency at which the WOPI clients make calls to the host. Each WOPI application may choose how best to respect the expressed desire from the host. WOPI applications may respond in manners such as reducing the frequency of CheckFileInfo calls and extending the window between when a user makes a change and the updated document gets saved back to the WOPI host. It is advised that hosts sharing this value between responses for distinct users of the same document at any given time may yield more deterministic results from the clients.

Possible Values:

Normal The WOPI host is healthy and does not want any additional request throttling.

Minor The WOPI host is requesting a small amount of throttling from the WOPI client.

Medium The WOPI host is requesting a medium amount of throttling from the WOPI client.

Major The WOPI host is requesting a large amount of throttling from the WOPI client.

Critical The WOPI host is requesting the WOPI client to apply the largest amount of throttling possible.

If no value is specified, or an invalid value is provided, the default behavior is to assume the WOPI host intended the value of None.

SHA256 A 256 bit SHA-2-encoded [FIPS 180-2] hash of the file contents, as a Base64-encoded **string**. Used for caching purposes in WOPI clients.

Office for the web Tip

See Optimizing document viewing for high volume for more details on how this property is used in Office for the web.

SharingStatus

A **string** value indicating whether the current document is shared with other users. The value can change upon adding or removing permissions to other users. Clients should use this value to help decide when to enable collaboration features as a document must be Shared in order to multi-user collaboration on the document.

Possible Values:

Private Only the document owner has permission to the file.

Shared At least one other user has access to the file via direct permissions or a sharing link.

If no value is specified, or an invalid value is provided, the default behavior is to assume the WOPI host intended the value of None.

UniqueContentId

In special cases, a host may choose to not provide a *SHA256*, but still have some mechanism for identifying that two different files contain the same content in the same manner as the *SHA256* is used.

This **string** value can be provided rather than a *SHA256* value if and only if the host can guarantee that two different files with the same content will have the same UniqueContentId value.

Office for the web Tip

See Optimizing document viewing for high volume for more details on how this property is used in Office for the web.

5.10 Unused and future properties

The following properties are defined as valid CheckFileInfo response properties. However, they are not used, either because they are pending deprecation or they are designated for future features of WOPI clients and WOPI servers.

ClientUrl A user-accessible URI directly to the file intended for opening the file through a client. Can be a DAV URL (**RFC 5323**), but may be any URL that can be handled by a client that can open a file of the given type.

CobaltCapabilities A **array of strings** that contains the Cobalt capabilities supported by the host. If *SupportsCobalt* is set to `false`, this property must be ignored by the WOPI client. Any value other than the possible values listed below must be ignored by the WOPI client.

Possible Values:

DownloadStreaming This type of CobaltCapabilities indicates the host supports Cobalt streaming for download at the moment it handles the request.

DisableBrowserCachingOfUserContent A **Boolean** value that indicates that the WOPI client should disable caching of file contents in the browser cache. Note that this has important performance implications for web browser-based WOPI clients.

EditAndReplyUrl *Not yet documented.*

HostAuthenticationId A **string** value uniquely identifying the user currently accessing the file.

> **Warning:** This property should not be used. Hosts should use the *UserId* property instead.

HostEmbeddedEditUrl A URI to a web page that provides access to an editing experience for the file that can be embedded in another HTML page. For example, a page that provides an HTML snippet that can be inserted into the HTML of a blog.

HostNotes A **string** that is used by the host to pass arbitrary information to the WOPI client. The client must ignore this string if it does not recognize its contents. A host must not require that a client understand the contents of this string to operate.

HostRestUrl A URI that is the base URI for REST operations for the file.

IrmPolicyDescription A **string** that the WOPI client should display to the user indicating the IRM (Information Rights Management) policy for the file. This value should be combined with *IrmPolicyTitle*.

IrmPolicyTitle A **string** that the WOPI client should display to the user indicating the IRM policy for the file. This value should be combined with *IrmPolicyDescription*.

PresenceProvider A **string** that identifies the provider of information that a WOPI client may use to discover information about the user's online status (for example, whether a user is available via instant messenger).

PresenceUserId A **string** that identifies the user in the context of the *PresenceProvider*.

ProtectInClient A **Boolean** value that indicates that the WOPI client should take measures to prevent copying and printing of the file. This is intended to help enforce IRM.

SignInUrl A URI that will allow the user to sign in using the host's authentication system. This property can be used when supporting anonymous users. If this property is not provided, no sign in UI will be shown in Office Online.

See also:

SignoutUrl

SupportsFileCreation A **Boolean** value that indicates that the host supports creating new files using the WOPI client.

SupportsScenarioLinks A **Boolean** value that indicates that the host supports scenarios where users can operate on files in limited ways via restricted URLs.

SupportsSecureStore A **Boolean** value that indicates that the host supports calls to a secure data store utilizing credentials stored in the file.

TenantId A **string** value uniquely identifying the user's 'tenant,' or group/organization to which they belong.

> **Caution:** The presence of this property does not remove the uniqueness and consistency requirements listed above. User properties are expected to be unique *per user* and consistent over time regardless of the presence of a *TenantId*.

TimeZone A **string** that is used to pass time zone information to a WOPI client. The format of this value is determined by the host.

UserPrincipalName A **string** value uniquely identifying the user currently accessing the file.

WebEditingDisabled A **Boolean** value that indicates that the WOPI client must not allow the user to edit the file.

> **Note:** This does not mean that the user doesn't have rights to edit the file. Hosts should use the *UserCanWrite* property for that purpose.

5.10.1 Workflow properties

Pre-release Feature

This documentation is for an upcoming feature and may undergo dramatic changes prior to final release. Pre-release features are available in the Test environment only.

WorkflowType

An **array of strings** containing the workflow types available for the document. Possible values are:

- `Assign`
- `Submit`

This property must always be specified if *WorkflowUrl* or WorkflowPostMessage are provided. If this property is *not* supplied, then both *WorkflowUrl* and WorkflowPostMessage must be ignored by the WOPI client.

Conversely, if the WorkflowType property is provided but neither *WorkflowUrl* nor WorkflowPostMessage are provided, then the WorkflowType value must be ignored by the WOPI client.

Important: While this property is an array of strings, note that specific values of WorkflowType may be mutually exclusive depending on the WOPI client. WOPI clients must use the following guidelines when handling values in the WorkflowType array:

- If no supported values are provided, the WOPI client must behave as though the WorkflowType property was not provided.
- If multiple values are provided and the WOPI client does not support multiple values, the client may use the first supported value provided in the array or behave as though the WorkflowType property was not provided.

WorkflowUrl

A URI to a location that allows the user to participate in a workflow for the file.

Important: This value will be ignored if *WorkflowType* is not provided.

5.11 Deprecated properties

The following properties are deprecated and should no longer be used by WOPI clients or servers.

BreadcrumbDocUrl Deprecated since version 2014.06.01.

EditingCannotSave Deprecated since version 2014.06.01.

HostName Deprecated since version 2016.01.12.

PrivacyUrl Deprecated since version 2015.06.01.

SupportsCoauth Deprecated since version 2016.01.27.

TermsOfUseUrl Deprecated since version 2015.06.01.

CHAPTER

SIX

GETFILE

Required for

GET **/wopi/files/** (*file_id*) **/contents**
The GetFile operation retrieves a file from a host.

Office for the web Tip

By default, Office for the web will use the GetFile WOPI request to retrieve files from the host. However, hosts can override this behavior by providing a direct URL to the file using the *FileUrl* property in the *CheckFileInfo* response.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-MaxExpectedSize* – An **integer** specifying the upper bound of the expected size of the file being requested. Optional. The host should use the maximum value of a 4-byte integer if this value is not set in the request. If the file requested is larger than this value, the host must respond with a 412 Precondition Failed.

Response Headers

- *X-WOPI-ItemVersion* – An optional **string** value indicating the version of the file. Its value should be the same as *Version* value in *CheckFileInfo*.

Response Body The response body must be the full binary contents of the file.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 412 Precondition Failed – File is larger than **X-WOPI-MaxExpectedSize**
- 500 Internal Server Error – Server error

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

CHAPTER

SEVEN

LOCK

Required for

POST /wopi/files/ (*file_id*)

The Lock operation locks a file for editing by the WOPI client application instance that requested the lock. To support editing files, WOPI clients require that the WOPI host support locking files. When locked, a file should not be writable by other applications.

If the file is currently unlocked, the host should lock the file and return 200 OK.

If the file is currently locked and the **X-WOPI-Lock** value matches the lock currently on the file, the host should treat the request as if it is a *RefreshLock* request. That is, the host should refresh the lock timer and return 200 OK.

In all other cases, the host must return a "lock mismatch" response (409 Conflict) and include an **X-WOPI-Lock** response header containing the value of the current lock on the file.

In the case where the file is locked by someone other than a WOPI client, hosts should still always include the current lock ID in the **X-WOPI-Lock** response header. However, if the current lock ID is not representable as a WOPI lock (for example, it is longer than the *maximum lock length*), the **X-WOPI-Lock** response header should be set to the empty string or omitted completely.

See *Lock* for more general information regarding locks.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `LOCK`. Required.
- *X-WOPI-Lock* – A **string** provided by the WOPI client that the host must use to identify the lock on the file. Required.

Response Headers

- *X-WOPI-Lock* – A **string** value identifying the current lock on the file. This header must always be included when responding to the request with 409 Conflict. It should not be included when responding to the request with 200 OK.

- *X-WOPI-LockFailureReason* – An optional **string** value indicating the cause of a lock failure. This header may be included when responding to the request with 409 Conflict. There is no standard for how this string is formatted, and it must only be used for logging purposes.
- *X-WOPI-LockedByOtherInterface* – Deprecated since version 2015-12-15: This header is deprecated and should be ignored by WOPI clients.
- *X-WOPI-ItemVersion* – An optional **string** value indicating the version of the file. Its value should be the same as *Version* value in *CheckFileInfo*.

Status Codes

- 200 OK – Success
- 400 Bad Request – **X-WOPI-Lock** was not provided or was empty
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Lock mismatch/locked by another interface; an **X-WOPI-Lock** response header containing the value of the current lock on the file must always be included when using this response code
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

CHAPTER

EIGHT

GETLOCK

POST /wopi/files/ (*file_id*)

The GetLock operation retrieves a lock on a file. Note that this operation *does not create a new lock.* Rather, this operation always returns the current lock value in the **X-WOPI-Lock** response header. Because of this, its semantics differ slightly from the other lock-related operations.

If the file is currently not locked, the host must return a 200 OK and include an **X-WOPI-Lock** response header set to the empty string.

If the file is currently locked, the host should return a 200 OK and include an **X-WOPI-Lock** response header containing the value of the current lock on the file. If the current lock ID is not representable as a WOPI lock (for example, it is longer than the *maximum lock length*), the host should return a 409 Conflict and set the **X-WOPI-Lock** response header to the empty string or omit it completely.

Tip: While a 409 Conflict is technically a valid response to this operation, it is rarely needed in practice, and hosts should respond with a 200 OK in most cases.

See *Lock* for more general information regarding locks.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `GET_LOCK`. Required.

Response Headers

- *X-WOPI-Lock* – A **string** value identifying the current lock on the file. Unlike other lock operations, this header is required when responding to the request with either 200 OK or 409 Conflict.
- *X-WOPI-LockFailureReason* – An optional **string** value indicating the cause of a lock failure. This header may be included when responding to the request with 409 Conflict. There is no standard for how this string is formatted, and it must only be used for logging purposes.
- *X-WOPI-LockedByOtherInterface* – Deprecated since version 2015-12-15: This header is deprecated and should be ignored by WOPI clients.

Status Codes

- 200 OK – Success; an **X-WOPI-Lock** response header containing the value of the current lock on the file must always be included when using this response code
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Lock mismatch/locked by another interface; an **X-WOPI-Lock** response header containing the value of the current lock on the file must always be included when using this response code
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

CHAPTER

NINE

REFRESHLOCK

Required for

POST /wopi/files/ (*file_id*)

The RefreshLock operation refreshes the lock on a file by resetting its automatic expiration timer to 30 minutes. The refreshed lock must expire automatically after 30 minutes unless it is modified by a subsequent WOPI operation, such as *Unlock* or *RefreshLock*.

WOPI clients will usually make a *Lock* request to lock a file prior to calling this operation. The WOPI client will pass the lock ID established by that previous *Lock* operation in the **X-WOPI-Lock** request header.

If the file is currently locked and the **X-WOPI-Lock** value does not match the lock currently on the file, or if the file is unlocked, the host must return a "lock mismatch" response (409 Conflict) and include an **X-WOPI-Lock** response header containing the value of the current lock on the file. In the case where the file is unlocked, the host must set **X-WOPI-Lock** to the empty string.

In the case where the file is locked by someone other than a WOPI client, hosts should still always include the current lock ID in the **X-WOPI-Lock** response header. However, if the current lock ID is not representable as a WOPI lock (for example, it is longer than the *maximum lock length*), the **X-WOPI-Lock** response header should be set to the empty string or omitted completely.

See *Lock* for more general information regarding locks.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** REFRESH_LOCK. Required.
- *X-WOPI-Lock* – A **string** provided by the WOPI client that the host must use to identify the existing lock on the file. Required.

Response Headers

- *X-WOPI-Lock* – A **string** value identifying the current lock on the file. This header must always be included when responding to the request with 409 Conflict. It should not be included when responding to the request with 200 OK.
- *X-WOPI-LockFailureReason* – An optional **string** value indicating the cause of a lock failure. This header may be included when responding to the request with 409 Conflict. There is no standard for how this string is formatted, and it must only be used for logging purposes.

- *X-WOPI-LockedByOtherInterface* – Deprecated since version 2015-12-15: This header is deprecated and should be ignored by WOPI clients.

Status Codes

- 200 OK – Success
- 400 Bad Request – **X-WOPI-Lock** was not provided or was empty
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Lock mismatch/locked by another interface; an **X-WOPI-Lock** response header containing the value of the current lock on the file must always be included when using this response code
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

CHAPTER

TEN

UNLOCK

Required for

POST /wopi/files/ (*file_id*)

The Unlock operation releases the lock on a file.

WOPI clients will usually make a *Lock* request to lock a file prior to calling this operation. The WOPI client will pass the lock ID established by that previous *Lock* operation in the **X-WOPI-Lock** request header.

If the file is currently locked and the **X-WOPI-Lock** value does not match the lock currently on the file, or if the file is unlocked, the host must return a "lock mismatch" response (409 Conflict) and include an **X-WOPI-Lock** response header containing the value of the current lock on the file. In the case where the file is unlocked, the host must set **X-WOPI-Lock** to the empty string.

In the case where the file is locked by someone other than a WOPI client, hosts should still always include the current lock ID in the **X-WOPI-Lock** response header. However, if the current lock ID is not representable as a WOPI lock (for example, it is longer than the *maximum lock length*), the **X-WOPI-Lock** response header should be set to the empty string or omitted completely.

See *Lock* for more general information regarding locks.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `UNLOCK`. Required.
- *X-WOPI-Lock* – A **string** provided by the WOPI client that the host must use to identify the lock on the file. Required.

Response Headers

- *X-WOPI-Lock* – A **string** value identifying the current lock on the file. This header must always be included when responding to the request with 409 Conflict. It should not be included when responding to the request with 200 OK.
- *X-WOPI-LockFailureReason* – An optional **string** value indicating the cause of a lock failure. This header may be included when responding to the request with 409 Conflict. There is no standard for how this string is formatted, and it must only be used for logging purposes.

- *X-WOPI-LockedByOtherInterface* – Deprecated since version 2015-12-15: This header is deprecated and should be ignored by WOPI clients.
- *X-WOPI-ItemVersion* – An optional **string** value indicating the version of the file. Its value should be the same as *Version* value in *CheckFileInfo*.

Status Codes

- 200 OK – Success
- 400 Bad Request – **X-WOPI-Lock** was not provided or was empty
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Lock mismatch/locked by another interface; an **X-WOPI-Lock** response header containing the value of the current lock on the file must always be included when using this response code
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

CHAPTER

ELEVEN

UNLOCKANDRELOCK

Required for

POST /wopi/files/ (*file_id*)

The UnlockAndRelock operation releases a lock on a file, and then immediately takes a new lock on the file.

Important: This operation must be atomic.

UnlockAndRelock is very similar semantically to the *Lock* operation. The two operations share the same **X-WOPI-Override** value. Thus, hosts must differentiate the two operations based on the presence, or lack of, the **X-WOPI-OldLock** request header.

Unlike the *Lock* operation, the UnlockAndRelock operation passes the current expected lock ID in the **X-WOPI-OldLock** request header. The **X-WOPI-Lock** value is the lock ID for the new lock.

If the file is currently locked and the **X-WOPI-OldLock** value does not match the lock currently on the file, or if the file is unlocked, the host must return a "lock mismatch" response (409 Conflict) and include an **X-WOPI-Lock** response header containing the value of the current lock on the file. In the case where the file is unlocked, the host must set **X-WOPI-Lock** to the empty string.

In the case where the file is locked by someone other than a WOPI client, hosts should still always include the current lock ID in the **X-WOPI-Lock** response header. However, if the current lock ID is not representable as a WOPI lock (for example, it is longer than the *maximum lock length*), the **X-WOPI-Lock** response header should be set to the empty string or omitted completely.

See *Lock* for more general information regarding locks.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `LOCK`. Required.
- *X-WOPI-Lock* – A **string** provided by the WOPI client that the host must use to identify the new lock on the file. The maximum length of a lock ID is 1024 ASCII characters. Required.
- *X-WOPI-OldLock* – A **string** provided by the WOPI client that is the existing lock on the file. Required. Note that if **X-WOPI-OldLock** is not provided, the request is identical to a *Lock* request.

Response Headers

- *X-WOPI-Lock* – A **string** value identifying the current lock on the file. This header must always be included when responding to the request with 409 Conflict. It should not be included when responding to the request with 200 OK.
- *X-WOPI-LockFailureReason* – An optional **string** value indicating the cause of a lock failure. This header may be included when responding to the request with 409 Conflict. There is no standard for how this string is formatted, and it must only be used for logging purposes.
- *X-WOPI-LockedByOtherInterface* – Deprecated since version 2015-12-15: This header is deprecated and should be ignored by WOPI clients.

Status Codes

- 200 OK – Success
- 400 Bad Request – **X-WOPI-Lock** was not provided or was empty
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Lock mismatch/locked by another interface; an **X-WOPI-Lock** response header containing the value of the current lock on the file must always be included when using this response code
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

CHAPTER

TWELVE

PUTFILE

Required for

POST /wopi/files/ (*file_id*) **/contents**

The PutFile operation updates a file's binary contents.

WOPI clients will usually make a *Lock* request to lock a file prior to calling this operation. The WOPI client will pass the lock ID established by that previous *Lock* operation in the **X-WOPI-Lock** request header.

When a host receives a PutFile request on a file that is not locked, the host must check the current size of the file. If it is 0 bytes, the PutFile request should be considered valid and should proceed. If it is any value other than 0 bytes, or is missing altogether, the host should respond with a 409 Conflict. For more information, see Creating new files using Office for the web.

If the file is currently locked and the **X-WOPI-Lock** value does not match the lock currently on the file the host must return a "lock mismatch" response (409 Conflict) and include an **X-WOPI-Lock** response header containing the value of the current lock on the file. In the case where the file is unlocked, the host must set **X-WOPI-Lock** to the empty string.

In the case where the file is locked by someone other than a WOPI client, hosts should still always include the current lock ID in the **X-WOPI-Lock** response header. However, if the current lock ID is not representable as a WOPI lock (for example, it is longer than the *maximum lock length*), the **X-WOPI-Lock** response header should be set to the empty string or omitted completely.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `PUT`. Required.
- *X-WOPI-Lock* – A **string** provided by the WOPI client in a previous *Lock* request. Note that this header will not be included during document creation.
- *X-WOPI-Editors* – A comma-delimited **string** of *UserId* values representing all the users who contributed changes to the document in this PutFile request.

Request Body The request body must be the full binary contents of the file.

Response Headers

- *X-WOPI-Lock* – A **string** value identifying the current lock on the file. This header must always be included when responding to the request with 409 Conflict. It should not be included when responding to the request with 200 OK.
- *X-WOPI-LockFailureReason* – An optional **string** value indicating the cause of a lock failure. This header may be included when responding to the request with 409 Conflict. There is no standard for how this string is formatted, and it must only be used for logging purposes.
- *X-WOPI-LockedByOtherInterface* – Deprecated since version 2015-12-15: This header is deprecated and should be ignored by WOPI clients.
- *X-WOPI-ItemVersion* – An optional **string** value indicating the version of the file. Its value should be the same as *Version* value in *CheckFileInfo*.

Tip: For PutFile responses, this should be the version of the file *after* the PutFile operation.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Lock mismatch/locked by another interface; an **X-WOPI-Lock** response header containing the value of the current lock on the file must always be included when using this response code
- 413 Request Entity Too Large – File is too large; the maximum file size is host-specific
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

CHAPTER

THIRTEEN

PUTRELATIVEFILE

Required for

`POST /wopi/files/` (*file_id*)

The PutRelativeFile operation creates a new file on the host based on the current file. The host must use the content in the POST body to create a new file.

The PutRelativeFile operation has two distinct modes: *specific* and *suggested*. The primary difference between the two modes is whether the WOPI client expects the host to use the file name provided exactly (*specific* mode) or if the host can adjust the file name in order to make the request succeed (*suggested* mode).

Hosts can determine the mode of the operation based on which of the mutually exclusive **X-WOPI-RelativeTarget** (indicates *specific* mode) or **X-WOPI-SuggestedTarget** (indicates *suggested* mode) request headers is used. The expected behavior for each mode is described in detail below.

The PutRelativeFile operation may be called on a file that is not locked, so the **X-WOPI-Lock** request header is not included in this operation. An example of when this might occur is if a user uses the *Save As* feature when viewing a document in read-only mode. The source file will not be locked in this case, but the PutRelativeFile operation will be invoked on it.

Note, however, that a file matching the target name might be locked, and in *specific* mode, the host must respond with a 409 Conflict and include a **X-WOPI-Lock** response header as described below.

Important: If a WOPI host sets the *SupportsUpdate* property in *CheckFileInfo* to `true`, then the host is expected to implement the PutRelativeFile operation. However, a host may choose not to implement this operation even though *SupportsUpdate* is `true`, but they must do the following:

1. Set the *UserCanNotWriteRelative* property to `true` always.
2. Return a 501 Not Implemented to all PutRelativeFile requests.

Excel for the web Note

Excel for the web uses this operation in the following two ways:

1. As part of the *Save As* feature. If PutRelativeFile is not supported, the *Save As* feature will not work in Excel for the web.
2. To support editing of some Excel files in Excel for the web. Some files may contain content that is not currently supported in Excel for the web. In this case, Excel for the web will prompt the user to save an editable copy of the document, removing all unsupported content so that the file can be edited in Excel for the web. If PutRelativeFile is not supported, files with unsupported content will not be editable in Excel Online.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `PUT_RELATIVE`. Required.
- *X-WOPI-SuggestedTarget* – A UTF-7 encoded **string** specifying either a file extension or a full file name, including the file extension. Hosts can differentiate between full file names and extensions as follows:
  - If the string begins with a period (`.`), it is a file extension.
  - Otherwise, it is a full file name.

  If only the extension is provided, the name of the initial file without extension should be combined with the extension to create the proposed name.

  The response to a request including this header must never result in a 400 Bad Request or 409 Conflict. Rather, the host must modify the proposed name as needed to create a new file that is both legally named and does not overwrite any existing file, while preserving the file extension.

  This header must be present if **X-WOPI-RelativeTarget** is not present; the two headers are mutually exclusive. If both headers are present the host should respond with a 400 Bad Request or 501 Not Implemented.
- *X-WOPI-RelativeTarget* – A UTF-7 encoded **string** that specifies a full file name including the file extension. The host must not modify the name to fulfill the request.

  If the specified name is illegal, the host must respond with a 400 Bad Request.

  If a file with the specified name already exists, the host must respond with a 409 Conflict, unless the **X-WOPI-OverwriteRelativeTarget** request header is set to `true`. When responding with a 409 Conflict for this reason, the host may include an **X-WOPI-ValidRelativeTarget** specifying a file name that is valid.

  If the **X-WOPI-OverwriteRelativeTarget** request header is set to `true` and a file with the specified name already exists and is locked, the host must respond with a 409 Conflict and include an **X-WOPI-Lock** response header containing the value of the current lock on the file.

  This header must be present if **X-WOPI-SuggestedTarget** is not present; the two headers are mutually exclusive. If both headers are present the host should respond with a 400 Bad Request or 501 Not Implemented.
- *X-WOPI-OverwriteRelativeTarget* – A **Boolean** value that specifies whether the host must overwrite the file name if it exists. The default value is `false`. In other words, if **X-WOPI-OverwriteRelativeTarget** is not explicitly included on the request, hosts must behave as though its value is `false`.

  This header is only valid if the **X-WOPI-RelativeTarget** is also included on the request. It must be ignored in all other cases.

  If the user is not authorized to overwrite the target file, the host must respond with a 501 Not Implemented.

- *X-WOPI-Size* – An **integer** that specifies the size of the file in bytes.
- *X-WOPI-FileConversion* – A header whose presence indicates that the request is being made in the context of a binary document conversion. This header will only be included on the request in that case. Thus, if **X-WOPI-FileConversion** is not explicitly included on the request, hosts must behave as if the PutRelativeFile request is not being made as part of a binary document conversion.

  See Editing binary document formats for more information on the conversion process and how this header can be used.

Request Body The request body must be the full binary contents of the file.

Response Headers

- *X-WOPI-ValidRelativeTarget* – A UTF-7 encoded **string** that specifies a full file name including the file extension. This header may be used when responding with a 409 Conflict because a file with the requested name already exists, or when responding with a 400 Bad Request because the requested name contained invalid characters. If this response header is included, the WOPI client should automatically retry the PutRelativeFile operation using the contents of this header as the **X-WOPI-RelativeTarget** value and should not display an error message to the user.
- *X-WOPI-Lock* – A **string** value identifying the current lock on the file. This header must always be included when responding to the request with 409 Conflict. It should not be included when responding to the request with 200 OK.
- *X-WOPI-LockFailureReason* – An optional **string** value indicating the cause of a lock failure. This header may be included when responding to the request with 409 Conflict. There is no standard for how this string is formatted, and it must only be used for logging purposes.
- *X-WOPI-LockedByOtherInterface* – Deprecated since version 2015-12-15: This header is deprecated and should be ignored by WOPI clients.

Status Codes

- 200 OK – Success
- 400 Bad Request – Specified name is illegal
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Target file already exists or the file is locked; if the target file is locked, an **X-WOPI-Lock** response header containing the value of the current lock on the file must always be included
- 413 Request Entity Too Large – File is too large; the maximum size is host-specific
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported; if the host sets the *SupportsUpdate* and *UserCanNotWriteRelative* properties to `true` in *CheckFileInfo*, this response code must be used when this operation is called

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

13.1 Response

The response to a PutRelativeFile call is JSON (as specified in **RFC 4627**) containing a number of properties, some of which are optional.

All optional values default to the following values based on their type:

Type	Default value
Boolean	`false`
String	The empty string
Integer/Long	Varies; see individual properties for details
Array	Empty array

Important: No properties should be set to `null`. If you do not wish to set a property, simply omit it from the response and WOPI clients will use the default value.

13.2 Required response properties

The following properties must be present in all PutRelativeFile responses:

Name The **string** name of the file, including extension, without a path.

Url A **string** URI of the form `http://server/<...>/wopi/files/(file_id)?access_token=(access token)`, of the newly created file on the host. This URL is the *WOPISrc* for the new file with an *access token* appended. Or, stated differently, it is the URL to the host's *Files endpoint* for the new file, along with an *access token*. A GET request to this URL will invoke the *CheckFileInfo* operation.

> **Danger:** This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

13.3 Optional response properties

HostViewUrl The *HostViewUrl*, as a **string**, for the newly created file.

HostEditUrl The *HostEditUrl*, as a **string**, for the newly created file.

CHAPTER

FOURTEEN

RENAMEFILE

POST /wopi/files/ (*file_id*)

The RenameFile operation renames a file.

Important: Renaming the file must not cause the *File ID*, and by extension, the *WOPISrc*, to change.

If the host cannot rename the file because the name requested is invalid or conflicts with an existing file, the host should try to generate a different name based on the requested name that meets the file name requirements.

If the host cannot generate a different name, it should return an HTTP status code 400 Bad Request. The response must include an **X-WOPI-InvalidFileNameError** header that describes why the file name was invalid.

If the file is currently unlocked, the host should respond with a 200 OK and proceed with the rename.

If the file is currently locked and the **X-WOPI-Lock** value does not match the lock currently on the file the host must return a "lock mismatch" response (409 Conflict) and include an **X-WOPI-Lock** response header containing the value of the current lock on the file.

Office for the web Tip

Office for the web includes contains UI that users can use to rename files. In order to activate this UI in Office for the web, you must implement the RenameFile operation, and also do the following:

- Set *SupportsRename* and *UserCanRename* to true in your *CheckFileInfo* response.
- Set a *FileNameMaxLength* value if the default value is not correct for your WOPI host.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** RENAME_FILE. Required.
- *X-WOPI-Lock* – A **string** provided by the WOPI client that the host must use to identify the lock on the file.

- *X-WOPI-RequestedName* – A UTF-7 encoded **string** that is a file name, *not including the file extension.*

Response Headers

- *X-WOPI-InvalidFileNameError* – A **string** describing the reason the rename operation could not be completed. This header should only be included when the response code is 400 Bad Request. This value must only be used for logging purposes.
- *X-WOPI-Lock* – A **string** value identifying the current lock on the file. This header must always be included when responding to the request with 409 Conflict. It should not be included when responding to the request with 200 OK.
- *X-WOPI-LockFailureReason* – An optional **string** value indicating the cause of a lock failure. This header may be included when responding to the request with 409 Conflict. There is no standard for how this string is formatted, and it must only be used for logging purposes.
- *X-WOPI-LockedByOtherInterface* – Deprecated since version 2015-12-15: This header is deprecated and should be ignored by WOPI clients.

Status Codes

- 200 OK – Success
- 400 Bad Request – Specified name is illegal
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Lock mismatch/locked by another interface; an **X-WOPI-Lock** response header containing the value of the current lock on the file must always be included when using this response code
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

14.1 Response

The response to a RenameFile call is JSON (as specified in **RFC 4627**) containing a single required property:

Name The **string** name of the renamed file *without a path or file extension.*

Important: The **Name** property returned must *not* include the file extension. This is different than other similar WOPI operations such as *PutRelativeFile* and *CreateChildFile*.

CHAPTER

FIFTEEN

DELETEFILE

Required for

POST /wopi/files/ (*file_id*)

The DeleteFile operation deletes a file from a host.

If the file is currently locked, the host should return a 409 Conflict and include an **X-WOPI-Lock** response header containing the value of the current lock on the file. If the current lock ID is not representable as a WOPI lock (for example, it is longer than the *maximum lock length*), the host should return a 409 Conflict and set the **X-WOPI-Lock** response header to the empty string or omit it completely.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `DELETE`. Required.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Lock mismatch/locked by another interface; an **X-WOPI-Lock** response header containing the value of the current lock on the file must always be included when using this response code
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

CHAPTER

SIXTEEN

ENUMERATEANCESTORS (FILES)

Required for

See also:

EnumerateAncestors (containers)

GET /wopi/files/ (*file_id*) **/ancestry**

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Response Headers

- *X-WOPI-EnumerationIncomplete* – An optional header indicating that the enumeration of the container's ancestry is incomplete. If set, the value of this header must be the string `true`.

  A WOPI client may choose to issue additional EnumerateAncestors requests to complete the enumeration.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

16.1 Response

The response to a EnumerateAncestors call is JSON (as specified in **RFC 4627**) containing the following required properties:

AncestorsWithRootFirst An array of JSON-formatted objects containing the following properties:

Name The name of the container without a path. This value should match the Name property in a *CheckContainerInfo* response. Required.

Url A URI to the container, including a valid *access token*. Required.

> **Danger:** This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

The array must always be ordered such that the ancestor closest to the root is the first element.

If there are no ancestor containers, this property should be an empty array.

16.2 Sample response

Consider a file in the following container hierarchy: `/root/grandparent/parent/myfile.docx`. When called on this file, the EnumerateAncestors operation should return the following:

```
{
  "AncestorsWithRootFirst": [
    {
      "Url": "http://.../wopi*/containers/<containerId>?access_token=<per_container_
↪token>",
      "Name": "root"
    },
    {
      "Url": "http://.../wopi*/containers/<containerId2>?access_token=<per_container_
↪token2>",
      "Name": "grandparent"
    },
    {
      "Url": "http://.../wopi*/containers/<containerId3>?access_token=<per_container_
↪token3>",
      "Name": "parent"
    }
  ]
}
```

CHAPTER

SEVENTEEN

GETSHAREURL (FILES)

Required for

See also:

GetShareUrl (containers)

`POST /wopi/files/` (*file_id*)

The GetShareUrl operation returns a *Share URL* that is suitable for viewing a shared file when launched in a web browser. A host can support multiple Share URL types, as described by the *SupportedShareUrlTypes* property. The **X-WOPI-UrlType** request header contains the Share URL type that should be returned.

If the **X-WOPI-UrlType** header is not present or contains a value that is invalid or not supported by the host, the host should respond with a 501 Not Implemented.

Parameters

- **`file_id`** (*`string`*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **`access_token`** (*`string`*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `GET_SHARE_URL`. Required.
- *X-WOPI-UrlType* – A **string** indicating what Share URL type to return. Required.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

17.1 Response

The response to a GetShareUrl call is JSON (as specified in **RFC 4627**) containing the following required properties:

ShareUrl A URI that points to a webpage that allows the user to access the file. Required.

See also:

Share Url

CHAPTER

EIGHTEEN

PUTUSERINFO

POST /wopi/files/ (*file_id*)

The PutUserInfo operation stores some basic user information on the host. When a host receives this request, they must store the UserInfo string which is contained in the body of the request. The UserInfo string should be associated with a particular user, and should be passed back to the WOPI client in subsequent *CheckFileInfo* responses in the *UserInfo* property.

The UserInfo string is provided in the body of the request, and has a maximum size of 1024 ASCII characters.

Note that WOPI clients will only call this WOPI operation if the host sets the *SupportsUserInfo* property to `true` in the *CheckFileInfo* response.

New in version 2015.08.03.

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `PUT_USER_INFO`. Required.

Body The request body must be the full UserInfo string.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

CHAPTER

NINETEEN

CHECKCONTAINERINFO

Required for

GET /wopi/containers/ (*container_id*)
The CheckContainerInfo operation is similar to the the *CheckFileInfo* operation, but operates on *containers* instead of files. CheckContainerInfo returns information about a container and a user's permissions on that container.

Parameters

- **container_id** (*string*) – A string that specifies a container ID of a container managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

19.1 Response

The response to a CheckContainerInfo call is JSON (as specified in **RFC 4627**).

All optional values default to the following values based on their type:

Type	Default value
Boolean	`false`
String	The empty string
Integer/Long	Varies; see individual properties for details
Array	Empty array

Important: No properties should be set to `null`. If you do not wish to set a property, simply omit it from the response and WOPI clients will use the default value.

19.2 Required response properties

The following properties must be present in all CheckContainerInfo responses:

Name The name of the container without a path. This value will be displayed in the WOPI client UI.

19.3 Other response properties

HostUrl A URI to a webpage for the container.

IsAnonymousUser A **Boolean** value indicating whether the user is authenticated with the host or not. This should match the *IsAnonymousUser* value returned in *CheckFileInfo*.

New in version 2017.02.15.

IsEduUser A **Boolean** value indicating whether the user is an education user or not. This should match the *IsEduUser* value returned in *CheckFileInfo*.

LicenseCheckForEditIsEnabled A **Boolean** value indicating whether the user is a business user or not. This should match the *LicenseCheckForEditIsEnabled* value returned in *CheckFileInfo*.

SharingUrl A URI to a webpage to allow the user to control sharing of the container. This is analogous to the *FileSharingUrl* in *CheckFileInfo*.

SupportedShareUrlTypes An **array** of strings containing the *Share URL* types supported by the host. The types indicate the sharing options available for the container itself and not on the files in the container.

These types can be passed in the **X-WOPI-UrlType** request header to signify which Share URL type to return for the *GetShareUrl (containers)* operation.

Possible Values:

ReadOnly This type of Share URL allows users to view the container using the URL, but does not give them permission to make changes to the container.

ReadWrite This type of Share URL allows users to both view and make changes to the container using the URL.

UserCanCreateChildContainer A **Boolean** value that indicates the user has permission to create a new container in the container.

UserCanCreateChildFile A **Boolean** value that indicates the user has permission to create a new file in the container.

UserCanDelete A **Boolean** value that indicates the user has permission to delete the container.

UserCanRename A **Boolean** value that indicates the user has permission to rename the container.

CHAPTER

TWENTY

CREATECHILDCONTAINER

Required for

`POST /wopi/containers/`(*container_id*)

The CreateChildContainer operation creates a new child *container* in the provided parent container.

The CreateChildContainer operation has two distinct modes: *specific* and *suggested*. The primary difference between the two modes is whether the WOPI client expects the host to use the container name provided exactly (*specific* mode) or if the host can adjust the container name in order to make the request succeed (*suggested* mode).

Hosts can determine the mode of the operation based on which of the mutually exclusive **X-WOPI-RelativeTarget** (indicates *specific* mode) or **X-WOPI-SuggestedTarget** (indicates *suggested* mode) request headers is used. The expected behavior for each mode is described in detail below.

Parameters

- **`container_id`** (*string*) – A string that specifies a container ID of a container managed by host. This string must be URL safe.

Query Parameters

- **`access_token`** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `CREATE_CHILD_CONTAINER`. Required.
- *X-WOPI-SuggestedTarget* – A UTF-7 encoded **string** that specifies a full container name. Required.

  The response to a request including this header must never result in a 400 Bad Request or 409 Conflict. Rather, the host must modify the proposed name as needed to create a new container that is legally named.

  This header must be present if **X-WOPI-RelativeTarget** is not present; the two headers are mutually exclusive. If both headers are present the host should respond with a 501 Not Implemented.
- *X-WOPI-RelativeTarget* – A UTF-7 encoded **string** that specifies a full container name. The host must not modify the name to fulfill the request.

  If the specified name is illegal, the host must respond with a 400 Bad Request.

  If a container with the specified name already exists, the host must respond with a 409 Conflict. When responding with a 409 Conflict for this reason, the host may include an **X-WOPI-ValidRelativeTarget** specifying a container name that is valid.

This header must be present if **X-WOPI-SuggestedTarget** is not present; the two headers are mutually exclusive. If both headers are present the host should respond with a 501 Not Implemented.

Response Headers

- *X-WOPI-InvalidContainerNameError* – A **string** describing the reason the CreateChildContainer operation could not be completed. This header should only be included when the response code is 400 Bad Request. This string is only used for logging purposes.

Status Codes

- 200 OK – Success
- 400 Bad Request – Specified name is illegal
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Target container already exists
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

20.1 Response

The response to a CreateChildContainer call is JSON (as specified in **RFC 4627**).

All optional values default to the following values based on their type:

Type	Default value
Boolean	`false`
String	The empty string
Integer/Long	Varies; see individual properties for details
Array	Empty array

Important: No properties should be set to `null`. If you do not wish to set a property, simply omit it from the response and WOPI clients will use the default value.

20.2 Required response properties

The following properties must be present in all CreateChildContainer responses:

20.2.1 ContainerPointer

A JSON-formatted object containing the following properties:

Name The name of the container without a path. This value should match the Name property in a *CheckContainerInfo* response. Required.

Url A URI to the container, including a valid *access token*. Required.

> **Danger:** This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

20.3 Other response properties

20.3.1 ContainerInfo

Hosts can optionally include the ContainerInfo property, which should match the *CheckContainerInfo* response for the newly created container.

If not provided, the WOPI client will call *CheckContainerInfo* to retrieve it. Including this property in the response is strongly recommended so that the WOPI client does not need to make an additional call to *CheckContainerInfo*.

20.4 Sample response

```
{
  "ContainerPointer" : {
    "Url" : "http://.../wopi*/containers/<containerId>?access_token=<per_container_
↪token>",
    "Name" : "Container Name"
  },
  "ContainerInfo" : {
    "Name" : "Container Name",
    "HostUrl" : "",
    "SharingUrl" : "",
    "UserCanCreateChildContainer" : false,
    "UserCanCreateChildFile" : false,
    "UserCanDelete" : false,
    "UserCanRename" : false
  }
}
```

CHAPTER

TWENTYONE

CREATECHILDFILE

Required for

`POST /wopi/containers/`(*container_id*)

The CreateChildFile operation creates a new file in the provided *container*. **The resulting file must be zero bytes in length.**

The CreateChildFile operation has two distinct modes: *specific* and *suggested*. The primary difference between the two modes is whether the WOPI client expects the host to use the file name provided exactly (*specific* mode) or if the host can adjust the file name in order to make the request succeed (*suggested* mode).

Hosts can determine the mode of the operation based on which of the mutually exclusive **X-WOPI-RelativeTarget** (indicates *specific* mode) or **X-WOPI-SuggestedTarget** (indicates *suggested* mode) request headers is used. The expected behavior for each mode is described in detail below.

Note that a file matching the target name might be locked, and in *specific* mode, the host must respond with a 409 Conflict and include a **X-WOPI-Lock** response header as described below.

Parameters

- **`container_id`** (*`string`*) – A string that specifies a container ID of a container managed by host. This string must be URL safe.

Query Parameters

- **`access_token`** (*`string`*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `CREATE_CHILD_FILE`. Required.
- *X-WOPI-SuggestedTarget* – A UTF-7 encoded **string** specifying either a file extension or a full file name, including the file extension. Hosts can differentiate between full file names and extensions as follows:
  - If the string begins with a period (`.`), it is a file extension.
  - Otherwise, it is a full file name.

  If only the extension is provided, the name of the initial file without extension should be combined with the extension to create the proposed name.

  The response to a request including this header must never result in a 400 Bad Request or 409 Conflict. Rather, the host must modify the proposed name as needed to create a new file that is both legally named and does not overwrite any existing file, while preserving the file extension.

This header must be present if **X-WOPI-RelativeTarget** is not present; the two headers are mutually exclusive. If both headers are present the host should respond with a 501 Not Implemented.

- *X-WOPI-RelativeTarget* – A UTF-7 encoded **string** that specifies a full file name including the file extension. The host must not modify the name to fulfill the request.

  If the specified name is illegal, the host must respond with a 400 Bad Request.

  If a file with the specified name already exists, the host must respond with a 409 Conflict, unless the **X-WOPI-OverwriteRelativeTarget** request header is set to `true`. When responding with a 409 Conflict for this reason, the host may include an **X-WOPI-ValidRelativeTarget** specifying a file name that is valid.

  If the **X-WOPI-OverwriteRelativeTarget** request header is set to `true` and a file with the specified name already exists and is locked, the host must respond with a 409 Conflict and include an **X-WOPI-Lock** response header containing the value of the current lock on the file.

  This header must be present if **X-WOPI-SuggestedTarget** is not present; the two headers are mutually exclusive. If both headers are present the host should respond with a 501 Not Implemented.

- *X-WOPI-OverwriteRelativeTarget* – A **Boolean** value that specifies whether the host must overwrite the file name if it exists. The default value is `false`. In other words, if **X-WOPI-OverwriteRelativeTarget** is not explicitly included on the request, hosts must behave as though its value is `false`.

  This header is only valid if the **X-WOPI-RelativeTarget** is also included on the request. It must be ignored in all other cases.

  If the user is not authorized to overwrite the target file, the host must respond with a 501 Not Implemented.

Response Headers

- *X-WOPI-InvalidFileNameError* – A **string** describing the reason the CreateChildFile operation could not be completed. This header should only be included when the response code is 400 Bad Request. This string is only used for logging purposes.
- *X-WOPI-ValidRelativeTarget* – A UTF-7 encoded **string** that specifies a full file name including the file extension. This header may be used when responding with a 409 Conflict because a file with the requested name already exists, or when responding with a 400 Bad Request because the requested name contained invalid characters. If this response header is included, the WOPI client should automatically retry the CreateChildFile operation using the contents of this header as the **X-WOPI-RelativeTarget** value and should not display an error message to the user.

Status Codes

- 200 OK – Success
- 400 Bad Request – Specified name is illegal
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Target file already exists
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

21.1 Response

The response to a CreateChildFile call is JSON (as specified in **RFC 4627**).

All optional values default to the following values based on their type:

Type	Default value
Boolean	`false`
String	The empty string
Integer/Long	Varies; see individual properties for details
Array	Empty array

Important: No properties should be set to `null`. If you do not wish to set a property, simply omit it from the response and WOPI clients will use the default value.

21.2 Required response properties

The following properties must be present in all CreateChildFile responses:

Name The **string** name of the file, including extension, without a path.

Url A **string** URI of the form `http://server/<...>/wopi/files/(file_id)?access_token=(access token)`, of the newly created file on the host. This URL is the *WOPISrc* for the new file with an *access token* appended. Or, stated differently, it is the URL to the host's *Files endpoint* for the new file, along with an *access token*. A GET request to this URL will invoke the *CheckFileInfo* operation.

> **Danger:** This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

21.3 Optional response properties

HostViewUrl The *HostViewUrl*, as a **string**, for the newly created file. This should match the value returned in *CheckFileInfo*.

HostEditNewUrl A URI to the host page that loads the `editnew` WOPI action.

HostEditUrl The *HostEditUrl*, as a **string**, for the newly created file. This should match the value returned in *CheckFileInfo*.

CHAPTER

TWENTYTWO

DELETECONTAINER

Required for

POST /wopi/containers/(*container_id*)

The DeleteContainer operation deletes a *container*.

Parameters

- **container_id** (*string*) – A string that specifies a container ID of a container managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** DELETE_CONTAINER. Required.

Status Codes

- 200 OK – Success
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Container has child files/containers
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

CHAPTER

TWENTYTHREE

ENUMERATEANCESTORS (CONTAINERS)

Required for

GET /wopi/containers/(*container_id*)**/ancestry**

The EnumerateAncestors operation enumerates all the parents of a given *container*, up to and including the *root container*.

Parameters

- **container_id** (*string*) – A string that specifies a container ID of a container managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Response Headers

- *X-WOPI-EnumerationIncomplete* – An optional header indicating that the enumeration of the container's ancestry is incomplete. If set, the value of this header must be the string `true`.

  A WOPI client may choose to issue additional EnumerateAncestors requests to complete the enumeration.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

23.1 Response

The response to a EnumerateAncestors call is JSON (as specified in **RFC 4627**) containing the following required properties:

AncestorsWithRootFirst An array of JSON-formatted objects containing the following properties:

Name The name of the container without a path. This value should match the Name property in a *CheckContainerInfo* response. Required.

Url A URI to the container, including a valid *access token*. Required.

> **Danger:** This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

The array must always be ordered such that the ancestor closest to the root is the first element.

If there are no ancestor containers, this property should be an empty array.

23.2 Sample response

Consider a file in the following container hierarchy: `/root/grandparent/parent/mycontainer`. When called on this container, the EnumerateAncestors operation should return the following:

```
{
  "AncestorsWithRootFirst": [
    {
      "Url": "http://.../wopi*/containers/<containerId>?access_token=<per_container_
↪token>",
      "Name": "root"
    },
    {
      "Url": "http://.../wopi*/containers/<containerId2>?access_token=<per_container_
↪token2>",
      "Name": "grandparent"
    },
    {
      "Url": "http://.../wopi*/containers/<containerId3>?access_token=<per_container_
↪token3>",
      "Name": "parent"
    }
  ]
}
```

CHAPTER

TWENTYFOUR

ENUMERATECHILDREN (CONTAINERS)

Required for

GET /wopi/containers/(*container_id*)**/children**

The EnumerateChildren operation enumerates all the immediate children of a given *container*. Note that paging is deliberately not supported by this operation. Responses are expected to include all immediate children of the container.

Parameters

- **container_id** (*string*) – A string that specifies a container ID of a container managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-FileExtensionFilterList* – A **string** value that the host must use to filter the returned child files. This header must be a list of comma-separated file extensions with a leading dot (`.`). There must be no whitespace and no trailing comma in the string. Wildcard characters are not permitted.

  If this header is included, the host must only return child files whose file extensions match the filter list, based on a case-insensitive match. For example, if this header is set to the value `.doc,.docx`, the host should include only files with the `doc` or `docx` file extension.

  This header value only affects the child files that are returned; it does not have any effect on child containers.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

24.1 Response

The response to a EnumerateChildren call is JSON (as specified in **RFC 4627**).

All optional values default to the following values based on their type:

Type	Default value
Boolean	`false`
String	The empty string
Integer/Long	Varies; see individual properties for details
Array	Empty array

Important: No properties should be set to `null`. If you do not wish to set a property, simply omit it from the response and WOPI clients will use the default value.

24.2 Required response properties

The following properties must be present in all EnumerateChildren responses:

24.2.1 ChildContainers

An array of JSON-formatted objects containing the following properties:

Name The name of the container without a path. This value should match the Name property in a *CheckContainerInfo* response. Required.

Url A URI to the container, including a valid *access token*. Required.

> **Danger:** This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

If there are no child containers, this property should be an empty array.

24.2.2 ChildFiles

An array of JSON-formatted objects containing the following properties:

Name Should match the *BaseFileName* property in *CheckFileInfo*. Required.

Size Should match the *Size* property in *CheckFileInfo*. Required.

Url A URI to the file, including a valid *access token*. Required.

> **Danger:** This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

Version Should match the *Version* property in *CheckFileInfo*. Required.

LastModifiedTime Should match the *LastModifiedTime* property in *CheckFileInfo*. Optional.

If there are no child files, this property should be an empty array.

24.3 Sample response

```
{
  "ChildContainers": [
    {
      "Url": "http://.../wopi*/containers/<containerId>?access_token=<per_file_token>
↪",
      "Name": "FolderName"
    },
    {
      "Url": "http://.../wopi*/containers/<containerId2>?access_token=<per_file_
↪token2>",
      "Name": "FolderName2"
    }
  ],
  "ChildFiles": [
    {
      "Url": "http://.../wopi*/files/<fileId>?access_token=<per_file_token>",
      "Name": "FileName",
      "Version": "version1",
      "Size": 7,
      "LastModifiedTime": "time last modified"
    }
  ]
}
```

CHAPTER

TWENTYFIVE

GETSHAREURL (CONTAINERS)

Required for

See also:

GetShareUrl (files)

`POST /wopi/containers/` (*container_id*)

The GetShareUrl operation returns a *Share URL* that is suitable for viewing a shared container when launched in a web browser. A host can support multiple Share URL types, as described by the *SupportedShareUrlTypes* property. The **X-WOPI-UrlType** request header contains the Share URL type that should be returned.

If the **X-WOPI-UrlType** header is not present or contains a value that is invalid or not supported by the host, the host should respond with a 501 Not Implemented.

Parameters

- **`container_id`** (*`string`*) – A string that specifies a container ID of a container managed by host. This string must be URL safe.

Query Parameters

- **`access_token`** (*`string`*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `GET_SHARE_URL`. Required.
- *X-WOPI-UrlType* – A **string** indicating what Share URL type to return. Required.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

25.1 Response

The response to a GetShareUrl call is JSON (as specified in **RFC 4627**) containing the following required properties:

ShareUrl A URI that points to a webpage that allows the user to access the container. Required.

See also:

Share Url

CHAPTER

TWENTYSIX

RENAMECONTAINER

Required for

POST /wopi/containers/(*container_id*)

The RenameContainer operation renames a container.

Parameters

- **container_id** (*string*) – A string that specifies a container ID of a container managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** RENAME_CONTAINER. Required.
- *X-WOPI-RequestedName* – A UTF-7 encoded **string** that is a container name. Required.

Response Headers

- *X-WOPI-InvalidContainerNameError* – A **string** describing the reason the RenameContainer operation could not be completed. This header should only be included when the response code is 400 Bad Request. This string is only used for logging purposes.

Status Codes

- 200 OK – Success
- 400 Bad Request – Specified name is illegal
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 409 Conflict – Target container already exists
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

26.1 Response

The response to a RenameContainer call is JSON (as specified in **RFC 4627**) containing the following required properties:

Name The **string** name of the renamed container.

CHAPTER

TWENTYSEVEN

CHECKECOSYSTEM

Required for

`GET /wopi/ecosystem`
The CheckEcosystem operation is similar to the the *CheckFileInfo* operation, but does not require a file or *container* ID.

Query Parameters

- **`access_token`** (*`string`*) – An *access token* that the host will use to determine whether the request is authorized.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

27.1 Response

The response to a CheckEcosystem call is JSON (as specified in **RFC 4627**).

All optional values default to the following values based on their type:

Type	Default value
Boolean	`false`
String	The empty string
Integer/Long	Varies; see individual properties for details
Array	Empty array

Important: No properties should be set to `null`. If you do not wish to set a property, simply omit it from the response and WOPI clients will use the default value.

27.2 Optional response properties

SupportsContainers Should match the *SupportsContainers* property in *CheckFileInfo*.

Note: Since all properties in the CheckEcosystem response are optional, an empty JSON response is valid.

CHAPTER

TWENTYEIGHT

GETECOSYSTEM (FILES)

Required for

See also:

GetEcosystem (containers)

GET /wopi/files/ (*file_id*) **/ecosystem_pointer**
The GetEcosystem operation returns the URI for the WOPI server's *Ecosystem endpoint*, given a file ID.

See also:

GetEcosystem (containers)

Parameters

- **file_id** (*string*) – A string that specifies a *file ID* of a file managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

28.1 Response

The response to a GetEcosystem call is JSON (as specified in **RFC 4627**) containing the following required properties:

Url A **string** URI for the WOPI server's *Ecosystem endpoint*, with an *access token* appended. A GET request to this URL will invoke the *CheckEcosystem* operation.

> **Danger:** This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

CHAPTER

TWENTYNINE

GETECOSYSTEM (CONTAINERS)

Required for

GET /wopi/containers/ (*container_id*) **/ecosystem_pointer**
The GetEcosystem operation returns the URI for the WOPI server's *Ecosystem endpoint*, given a *container* ID.

See also:

GetEcosystem (files)

Parameters

- **container_id** (*string*) – A string that specifies a container ID of a container managed by host. This string must be URL safe.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

29.1 Response

The response to a GetEcosystem call is JSON (as specified in **RFC 4627**) containing the following required properties:

Url A **string** URI for the WOPI server's *Ecosystem endpoint*, with an *access token* appended. A GET request to this URL will invoke the *CheckEcosystem* operation.

Danger: This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

CHAPTER

THIRTY

GETFILEWOPISRC (ECOSYSTEM)

POST /wopi/ecosystem

> **Warning:** This operation should not be called by WOPI clients at this time. It is reserved for future use and subject to change.

The GetFileWopiSrc operation is used to convert a host-specific file identifier into a *WopiSrc* value.

The WOPI client passes the host-specific file identifier in the **X-WOPI-HostNativeFileName** header. The host, in turn, returns a WopiSrc URL with an *access token* appended.

This operation is useful in cases where a WOPI client receives a file identifier that is not a proper WopiSrc, but can be translated into a valid WopiSrc value by the WOPI host.

For example, iOS applications can open files in Office for iOS using URL schemes. However, it may not be feasible for an application to generate a WopiSrc value itself. As long as it can generate a string value that can later be converted to a WopiSrc value by calling this operation, then the application can pass this value and rely on Office for iOS to call this operation to convert the file identifier into a WopiSrc.

Important: While a WOPI host can accept any string value as input, hosts are strongly recommended to support the following values in **X-WOPI-HostNativeFileName**:

- *HostEditUrl*
- *HostViewUrl*
- Any value the host returns in response to a *GetShareUrl (files)* call

---

See also:

This operation is also exposed as a *shortcut operation*.

Query Parameters

- **access_token** (*string*) – An *access token* that the host will use to determine whether the request is authorized.

Request Headers

- *X-WOPI-Override* – The **string** `GET_WOPI_SRC_WITH_ACCESS_TOKEN`. Required.
- *X-WOPI-HostNativeFileName* – A **string** representing the host-specific file identifier for a file.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

30.1 Response

The response to a GetFileWopiSrc call is JSON (as specified in **RFC 4627**) containing the following required properties:

Url A URI that represents a *WopiSrc* value with an *access token* appended.

> **Danger:** This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

30.2 Sample response

```
{
  "Url": "http://.../wopi*/[containers|files]/<id>?access_token=<file|container_token>
↪&access_token_ttl=<timestamp>"
}
```

CHAPTER

THIRTYONE

GETROOTCONTAINER (ECOSYSTEM)

Required for

`GET /wopi/ecosystem/root_container_pointer`
The GetRootContainer operation returns the *root container*. A WOPI client can use this operation to get a reference to the root container, from which the client can call *EnumerateChildren (containers)* to navigate a container hierarchy.

See also:

This operation is also exposed as a *shortcut operation*.

Query Parameters

- **`access_token`** (*`string`*) – An *access token* that the host will use to determine whether the request is authorized.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Invalid *access token*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

See also:

Standard WOPI request and response headers In addition to the request/response headers listed here, this operation may also use the *Standard WOPI request and response headers*.

31.1 Response

The response to a GetRootContainer call is JSON (as specified in **RFC 4627**).

All optional values default to the following values based on their type:

Type	Default value
Boolean	`false`
String	The empty string
Integer/Long	Varies; see individual properties for details
Array	Empty array

Important: No properties should be set to `null`. If you do not wish to set a property, simply omit it from the response and WOPI clients will use the default value.

31.2 Required response properties

The following properties must be present in all GetRootContainer responses:

31.2.1 ContainerPointer

A JSON-formatted object containing the following properties:

Name The name of the container without a path. This value should match the Name property in a *CheckContainerInfo* response. Required.

Url A URI to the container, including a valid *access token*. Required.

> **Danger:** This property includes an *access token* and thus has important security implications. See *Preventing 'token trading'* for more details.

31.3 Other response properties

31.3.1 ContainerInfo

Hosts can optionally include the ContainerInfo property, which should match the *CheckContainerInfo* response for the root container.

If not provided, the WOPI client will call *CheckContainerInfo* to retrieve it. Including this property in the response is strongly recommended so that the WOPI client does not need to make an additional call to *CheckContainerInfo*.

31.4 Sample response

```
{
  "ContainerPointer" : {
    "Url" : "http://.../wopi*/containers/<containerId>?access_token=<per_container_
↪token>",
    "Name" : "Container Name"
  },
  "ContainerInfo" : {
    "Name" : "Container Name",
    "HostUrl" : "",
    "SharingUrl" : "",
    "UserCanCreateChildContainer" : false,
    "UserCanCreateChildFile" : false,
    "UserCanDelete" : false,
    "UserCanRename" : false
  }
}
```

CHAPTER

THIRTYTWO

BOOTSTRAP

Required for

`GET /wopibootstrapper`

The Bootstrap operation is used to 'convert' an OAuth token into appropriate WOPI *access tokens* and provides access to WOPI operations for applications using OAuth tokens, like Office for iOS. For more information see native.

Important: Connections to the bootstrapper must be made using TLS (Transport Layer Security).

Request Headers

- Authorization – A **string** in the format `Bearer: <TOKEN>` where `<TOKEN>` is a Base64-encoded OAuth 2.0 token. If this header is missing or blank, or if the token provided is invalid, the host must respond with a 401 Unauthorized response and include the WWW-Authenticate header as described below.

Response Headers

- WWW-Authenticate – A **string** value formatted as described in *WWW-Authenticate response header format*.
- Content-Type – Must be `application/json` for *authenticated responses*.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Authorization failure; when responding with this status code, hosts must include a WWW-Authenticate response header with values as described in *WWW-Authenticate response header format*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error

32.1 Response

The response to a Bootstrap operation differs based on whether it is **authenticated** or **unauthenticated**. When possible, the WOPI client will provide authentication state information, as defined by OAuth 2.0 protocol, in the HTTP header in every request to the *Bootstrapper endpoint*. This authentication state will be sent in the form of the OAuth 2.0 Access token and will be contained in the Authorization header as described previously.

The host must verify that the provided token is valid. If it is not, the host must respond as described in the *unauthenticated response* section below. If the provided token is valid, then the host must respond as described in the *authenticated response* section below.

32.1.1 Authenticated response

When an **authenticated** request (i.e. a valid OAuth 2.0 access token is included in the Authorization HTTP header) is made to this endpoint, it returns a 200 OK response with a JSON (as specified in **RFC 4627**) response body. The response must include a single `Bootstrap` property, with the following properties nested within it as needed.

Required response properties

The following properties must be present in the `Bootstrap` property in all 200 OK Bootstrap responses:

EcosystemUrl A **string** URI for the WOPI server's *Ecosystem endpoint*, with a WOPI *access token* appended. A GET request to this URL will invoke the *CheckEcosystem* operation.

UserId A **string** value uniquely identifying the user making the request. This value should match the *UserId* value provided in *CheckFileInfo*. This ID is expected to be unique per user and consistent over time. See *Requirements for user identity properties* for more information.

SignInName A **string** value identifying the user making the request. This value is used to distinguish a user's account in the event a user has multiple accounts with a given host. This value is often an email address, though it is not required to be.

Optional response properties

UserFriendlyName A **string** that is the name of the user. This value should match the *UserFriendlyName* value provided in *CheckFileInfo*.

Sample response

```
{
  "Bootstrap": {
    "EcosystemUrl": "http://.../wopi*/ecosystem?access_token=<ecosystem_token>",
    "UserId": "User ID",
    "SignInName": "user@contoso.com",
    "UserFriendlyName": "User Name"
  }
}
```

32.1.2 Unauthenticated response

When an **unauthenticated** request (i.e. no access token is attached in the Authorization HTTP header) is made to this endpoint, it returns a 401 Unauthorized response containing sufficient information to facilitate user authentication with the host.

WWW-Authenticate response header format

The response must contain sufficient information for a WOPI client to perform the necessary authentication/authorization/token issuance flows with the host's identity provider, and result in an authenticated call to the same Bootstrapper endpoint.

The information for the successful authentication/authorization/token issuance flows must be returned in the WWW-Authenticate header of the 401 Unauthorized response with type "Bearer." The information that must be returned in a 401 Unauthorized response to an unauthenticated request is as follows:

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 Microsoft Office) that uniquely identifies the host. Allowed characters: [a-z, A-Z, 0-9]	No	`tp_contoso`
UrlSchemes	URL scheme your app uses. This is an ordered list by platform. Omit any platforms you do not support. Office will attempt to invoke these URL schemes in order before falling back to the webview auth.	No	{"iOS" : ["contoso","contoso-EMM"], "Android" : ["contoso","contoso-EMM"] "UWP": ["contoso","contoso-EMM"]} The value itself must be URL encoded

These parameters and their values must be formatted as follows:

- Values are contained within double-quotes (").
- Contiguous parameters are separated by commas with no comma after the trailing parameter/value pair.
- If no value is known/required for an optional parameter, it may be omitted from the WWW-Authenticate header.
- Multiple instances of WWW-Authenticate HTTP headers may exist in the response to an unauthenticated request to the Bootstrapper endpoint. However, there must be exactly one instance of the WWW-Authenticate header with the `Bearer` qualifier.

Sample unauthenticated response

```
HTTP/1.1 401 Unauthorized
<removed for brevity>
WWW-Authenticate: Bearer authorization_uri="https://contoso.com/api/oauth2/authorize",
↪tokenIssuance_uri="https://contoso.com/api/oauth2/token",providerId="tp_contoso",␣
↪UrlSchemes="%7B%22iOS%22%20%3A%20%5B%22contoso%22%2C%22contoso-EMM%22%5D%2C%20
↪%22Android%22%20%3A%20%5B%22contoso%22%2C%22contoso-EMM%22%5D%2C%20%22UWP%22%3A%20
↪%5B%22contoso%22%2C%22contoso-EMM%22%5D%7D"
Date: Wed, 24 Jun 2015 21:52:44 GMT
```

CHAPTER

THIRTYTHREE

GETNEWACCESSTOKEN

Required for

`POST /wopibootstrapper`

The GetNewAccessToken operation is used to retrieve a fresh WOPI *access token* for a given resource (i.e. a file or *container*), provided the caller has a valid OAuth 2.0 token.

This operation is called by OAuth-capable WOPI clients, such as Office for iOS, to refresh WOPI access tokens when they expire.

Request Headers

- *X-WOPI-EcosystemOperation* – The **string** `GET_NEW_ACCESS_TOKEN`. Required.
- *X-WOPI-WopiSrc* – The *WopiSrc* (a **string**) for the file or *container*
- Authorization – A **string** in the format `Bearer: <TOKEN>` where `<TOKEN>` is a Base64-encoded OAuth 2.0 token. If this header is missing, or the token provided is invalid, the host must respond with a 401 Unauthorized response and include the WWW-Authenticate header as described in *WWW-Authenticate response header format*.

Response Headers

- WWW-Authenticate – A **string** value formatted as described in *WWW-Authenticate response header format*. This header should only be included when responding with a 401 Unauthorized.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Authorization failure; when responding with this status code, hosts must include a WWW-Authenticate response header with values as described in *WWW-Authenticate response header format*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error

33.1 Response

The response to a GetNewAccessToken call is JSON (as specified in **RFC 4627**) containing the following required properties:

33.1.1 Bootstrap

The contents of this property should be the response to a *Bootstrap* call.

33.1.2 AccessTokenInfo

The contents of this property should be a the nested JSON-formatted object with the following properties.

AccessToken A **string** *access token* for the file specified in the **X-WOPI-WopiSrc** request header.

AccessTokenExpiry A **long** value representing the time that the *access token* provided in the response will expire. See *access_token_ttl* for more information on how this value is defined.

33.2 Sample response

```
{
  "Bootstrap": {
    "EcosystemUrl": "http://.../wopi*/ecosystem?access_token=<ecosystem_token>",
    "UserId": "User ID",
    "SignInName": "user@contoso.com",
    "UserFriendlyName": "User Name"
  },
  "AccessTokenInfo": {
    "AccessToken": "1234567890abcdef",
    "AccessTokenExpiry": 1234567890
  }
}
```

CHAPTER

THIRTYFOUR

SHORTCUT OPERATIONS

The following operations are defined on the *Bootstrapper endpoint* in addition to the *Ecosystem endpoint*. They are provided on the *Bootstrapper endpoint* as shortcuts to reduce HTTP round trips for native applications that use the Bootstrapper. They are equivalent to calling *Bootstrap* then using the *EcosystemUrl* to immediately execute an operation on the *Ecosystem endpoint*.

For example, in order to get the root container, a WOPI client might execute the following operations:

1. Call *Bootstrap*
2. Using the *EcosystemUrl* from the *Bootstrap* response, call the *GetRootContainer (ecosystem)* operation

However, using the *GetRootContainer (bootstrapper)* shortcut operation, the WOPI client can skip calling *Bootstrap*, and instead get the root container directly using an OAuth 2.0 token.

Important: Implementing shortcut operations is optional. If a host does not implement them, they should simply respond to the request as though it is a standard *Bootstrap* request. In other words, a host that doesn't implement shortcut operations can simply ignore the **X-WOPI-EcosystemOperation** header and treat the request as a standard *Bootstrap* request.

WOPI clients must thus expect that some shortcut operations will not include the data expected, and should fall back to calling the appropriate operation on the *Ecosystem endpoint* in such cases.

With this in mind, the responses for each 'shortcut' operation are a combination of the *Bootstrap* response and the response for the corresponding operation on the *Ecosystem endpoint*. The Bootstrapper response is contained in the `Bootstrap` property, and the operation's response is contained in a separate property with a name specific to the operation.

For example, the response to the *GetRootContainer (bootstrapper)* shortcut operation looks like this:

```
{
  "Bootstrap": {
    "EcosystemUrl": "http://.../wopi*/ecosystem?access_token=<ecosystem_token>",
    "UserId": "User ID",
    "SignInName": "user@contoso.com",
    "UserFriendlyName": "User Name"
  },
  "RootContainerInfo": {
    "ContainerPointer" : {
      "Url" : "http://.../wopi*/containers/<containerId>?access_token=<per_container_
↪token>",
      "Name" : "Container Name"
    },
    "ContainerInfo" : {
      "Name" : "Container Name",
```

(continues on next page)

(continued from previous page)

```
      "HostUrl" : "http://contoso/324332",
      "SharingUrl" : "http://contoso/323432/efewos",
      "UserCanCreateChildContainer" : false,
      "UserCanCreateChildFile" : false,
      "UserCanDelete" : false,
      "UserCanRename" : false
    }
  }
}
```

Note that since these operations are exposed through the *Bootstrapper endpoint*, they will be called using OAuth 2.0 access tokens, not WOPI access tokens.

34.1 GetFileWopiSrc (bootstrapper)

Warning: This operation should not be called by WOPI clients at this time. It is reserved for future use and subject to change.

POST /wopibootstrapper

This operation is equivalent to the *GetFileWopiSrc (ecosystem)* operation.

Important: The request/response semantics for this operation differ slightly from its counterpart on the *Ecosystem endpoint* since it is exposed on the *Bootstrapper endpoint* and thus will use OAuth 2.0 access tokens for authorization instead of WOPI access tokens.

Request Headers

- *X-WOPI-EcosystemOperation* – The **string** `GET_WOPI_SRC_WITH_ACCESS_TOKEN`. Required.
- *X-WOPI-HostNativeFileName* – A **string** representing the host-specific file identifier for a file.
- Authorization – A **string** in the format `Bearer: <TOKEN>` where `<TOKEN>` is a Base64-encoded OAuth 2.0 token. If this header is missing, or the token provided is invalid, the host must respond with a 401 Unauthorized response and include the WWW-Authenticate header as described in *WWW-Authenticate response header format*.

Response Headers

- WWW-Authenticate – A **string** value formatted as described in *WWW-Authenticate response header format*. This header should only be included when responding with a 401 Unauthorized.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Authorization failure; when responding with this status code, hosts must include a WWW-Authenticate response header with values as described in *WWW-Authenticate response header format*
- 404 Not Found – Resource not found/user unauthorized

- 500 Internal Server Error – Server error

34.1.1 Response

The response to a GetFileWopiSrc call is JSON (as specified in **RFC 4627**) containing the following required properties:

Bootstrap The contents of this property should be the response to a *Bootstrap* operation.

WopiSrcInfo The contents of this property should be the response to a *GetFileWopiSrc (ecosystem)* operation.

34.1.2 Sample response

```
{
  "Bootstrap": {
    "EcosystemUrl": "http://.../wopi*/ecosystem?access_token=<ecosystem_token>",
    "UserId": "User ID",
    "SignInName": "user@contoso.com",
    "UserFriendlyName": "User Name"
  },
  "WopiSrcInfo": {
    "Url": "http://.../wopi*/[containers|files]/<id>?access_token=<file|container_
↪token>&access_token_ttl=<timestamp>"
  }
}
```

34.2 GetRootContainer (bootstrapper)

Required for

POST /wopibootstrapper

This operation is equivalent to the *GetRootContainer (ecosystem)* operation.

Important: The request/response semantics for this operation differ slightly from its counterpart on the *Ecosystem endpoint* since it is exposed on the *Bootstrapper endpoint* and thus will use OAuth 2.0 access tokens for authorization instead of WOPI access tokens.

Request Headers

- *X-WOPI-EcosystemOperation* – The **string** `GET_ROOT_CONTAINER`. Required.
- Authorization – A **string** in the format `Bearer: <TOKEN>` where `<TOKEN>` is a Base64-encoded OAuth 2.0 token. If this header is missing, or the token provided is invalid, the host must respond with a 401 Unauthorized response and include the WWW-Authenticate header as described in *WWW-Authenticate response header format*.

Response Headers

- WWW-Authenticate – A **string** value formatted as described in *WWW-Authenticate response header format*. This header should only be included when responding with a 401 Unauthorized.

Status Codes

- 200 OK – Success
- 401 Unauthorized – Authorization failure; when responding with this status code, hosts must include a WWW-Authenticate response header with values as described in *WWW-Authenticate response header format*
- 404 Not Found – Resource not found/user unauthorized
- 500 Internal Server Error – Server error
- 501 Not Implemented – Operation not supported

34.2.1 Response

The response to a GetRootContainer call is JSON (as specified in **RFC 4627**) containing the following required properties:

Bootstrap The contents of this property should be the response to a *Bootstrap* call.

RootContainerInfo The contents of this property should be the response to a *GetRootContainer (ecosystem)* call.

34.2.2 Sample response

```
{
  "Bootstrap": {
    "EcosystemUrl": "http://.../wopi*/ecosystem?access_token=<ecosystem_token>",
    "UserId": "User ID",
    "SignInName": "user@contoso.com",
    "UserFriendlyName": "User Name"
  },
  "RootContainerInfo": {
    "ContainerPointer" : {
      "Url" : "http://.../wopi*/containers/<containerId>?access_token=<per_container_
↪token>",
      "Name" : "Container Name"
    },
    "ContainerInfo" : {
      "Name" : "Container Name",
      "HostUrl" : "http://contoso/324332",
      "SharingUrl" : "http://contoso/323432/efewos",
      "UserCanCreateChildContainer" : false,
      "UserCanCreateChildFile" : false,
      "UserCanDelete" : false,
      "UserCanRename" : false
    }
  }
}
```

CHAPTER

THIRTYFIVE

GLOSSARY

WOPI client A WOPI client is an application or service that executes WOPI operations on a WOPI server by issuing HTTP requests to the WOPI server's WOPI endpoints. Office for the web is an example of a WOPI client.

WOPI host

WOPI server A WOPI server, often called a WOPI host or just simply a *host*, is an application or service that implements WOPI endpoints and operations as described in this documentation, such as *CheckFileInfo* and *GetFile*. OneDrive for Business is an example of a WOPI host.

HTTP ROUTING TABLE

/wopi

/wopibootstrapper

INDEX