Amazon WorkDocs

Developer Guide

Amazon WorkDocs: Developer Guide

Table of Contents

What Is Amazon WorkDocs?

Amazon WorkDocs is a fully managed, secure, enterprise storage and sharing service with strong administrative controls and feedback capabilities that improve user productivity. Your files are stored in the cloud, safely and securely. Amazon WorkDocs even includes a synchronization application that always keeps selected folders on your local computer in sync with your cloud folders. Your files are only visible to you, and your designated contributors and viewers. Other members of your organization do not have access to any of your files unless you specifically grant them access.

You can share your files with other members of your organization for collaboration or review. The Amazon WorkDocs client applications can be used to view many different types of files, depending on the Internet media type of the file. Amazon WorkDocs supports all common document and image formats, and support for additional media types is constantly being added.

For more information, see Amazon WorkDocs.

Accessing

End users use the client applications to access their files. Non-administrative users never need to use the Amazon WorkDocs console or the administration dashboard. Amazon WorkDocs offers several different client applications and utilities:

- A web application used for document management and reviewing.
- Native apps for mobile devices used for document review.
- A document synchronization app used to synchronize a folder on your Mac or Windows desktop with your Amazon WorkDocs files.
- Web clipper browser extensions for several popular web browsers that allow you to save an image of a web page to your Amazon WorkDocs files.

Pricing

With Amazon WorkDocs, there are no upfront fees or commitments. You pay only for active user accounts, and the storage you use. For more information, go to Pricing.

Resources

The following related resources can help you as you work with this service.

- **Classes & Workshops** – Links to role-based and specialty courses as well as self-paced labs to help sharpen your AWS skills and gain practical experience.
- **AWS Developer Tools** – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- **AWS Whitepapers** – Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.
- **AWS Support Center** – The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- **AWS Support** – The primary web page for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
- **Contact Us** – A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues.
- **AWS Site Terms** – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Getting Started

The following code snippets can help you get started using the Amazon WorkDocs SDK.

Examples

Connect to Amazon WorkDocs Using IAM User Credentials and Query for Users

The following simple code, using the AWS SDK, illustrates the steps in making API calls using IAM user's API credentials. In this case API user and Amazon WorkDocs site belong to same AWS Account.

Ensure that the IAM user has been granted Amazon WorkDocs API access through an appropriate IAM policy.

The code sample uses describeUsers API to search for users and obtain Metadata for users. User metadata provides details such as first name, last name and User ID and Root Folder ID. Root Folder ID is particularly helpful if you want to perform any content upload or download operations on behalf of the user.

```
public class GetUserDemo {

  public static void main(String[] args) throws Exception {
    AWSCredentials longTermCredentials = new BasicAWSCredentials("accessKey", "secretKey");
    AmazonWorkDocsClient workDocsClient = new
    AmazonWorkDocsClient(longTermCredentials);
    // Change this endpoint based on the region of your site.
```

```
    workDocsClient.setEndpoint("https://workdocs.us-east-1.amazonaws.com");
    List<User> wdUsers = new ArrayList<>();
    DescribeUsersRequest request = new DescribeUsersRequest ();
    // The OrganizationId used here is an example.
    // You should use an OrganizationId unique to your site.
    request.setOrganizationId("d-123456789c");
    request.setQuery("joe");
    String marker = null;
    boolean hasMore = true;
    while (hasMore) {
      request.setMarker(marker);
      DescribeUsersResult result = workDocsClient.describeUsers(request);
      wdUsers.addAll(result.getUsers());
      marker = result.getMarker();
      if (marker == null) {
        hasMore = false;
      }
    }
    System.out.println("List of users matching string: joe");
    for (User wdUser : wdUsers) {
      System.out.printf("Firstname:%s | Lastname:%s | Email:%s | RootFolderId:%s\n",
          wdUser.getGivenName(), wdUser.getSurname(), wdUser.getEmailAddress(),
          wdUser.getRootFolderId());
    }
  }
}
```

Connect to Amazon WorkDocs by Assuming a Role and Browse a User's Root Folder

This simple sample code, using the AWS Java SDK, illustrates the individual steps for assuming a role and using the role's temporary security credentials to access Amazon WorkDocs. The code sample uses `describeFolderContents` API to list of items present in a user's folder.

```
public class AssumeRoleDemo {
  private static final String DEMO_ROLE_ARN = "arn:aws:iam::111122223333:role/workdocs-
readonly-role";
  static AmazonWorkDocsClient workDocsClient;

  public static void main(String[] args) throws Exception {
    AWSCredentials longTermCredentials = new BasicAWSCredentials("accessKey", "secretKey");

    // Step 1. Use developer's long-term credentials to call the
    // AWS Security Token Service (STS) AssumeRole API, specifying
    // the ARN for the role workdocs-readonly-role in 3rd party AWS account.

    AWSSecurityTokenServiceClient stsClient =
        new AWSSecurityTokenServiceClient(longTermCredentials);

    // If you are accessing a 3rd party account, then ExternalId should
    // be set on assumeRequest using the setExternalId() function.
    AssumeRoleRequest assumeRequest = new AssumeRoleRequest()
                                      .withRoleArn(DEMO_ROLE_ARN)
                                      .withDurationSeconds(3600)
                                      .withRoleSessionName("demo");

    AssumeRoleResult assumeResult = stsClient.assumeRole(assumeRequest);

    // Step 2. AssumeRole returns temporary security credentials for
    // the IAM role.
```

```
    BasicSessionCredentials temporaryCredentials =
        new BasicSessionCredentials(
            assumeResult.getCredentials().getAccessKeyId(),
            assumeResult.getCredentials().getSecretAccessKey(),
            assumeResult.getCredentials().getSessionToken()
            );

    workDocsClient = new AmazonWorkDocsClient(temporaryCredentials);
    // Change this endpoint based on the region of your site.
    workDocsClient.setEndpoint("https://workdocs.us-west-2.amazonaws.com");

    // Step 3. Make WorkDocs service calls using the temporary security credentials
    // obtained for workdocs-readonly-role
    // In this case a call has been made to get metadata of Folders and Documents
    // present in a user's root folder.

          describeFolder("folder-id");
  }

  private static void describeFolder(String folderId) {
    DescribeFolderContentsRequest request = new DescribeFolderContentsRequest();
    request.setFolderId(folderId);
    request.setLimit(2);
    List<DocumentMetadata> docs = new ArrayList<>();
    List<FolderMetadata> folders = new ArrayList<>();
    String marker = null;
    boolean hasMore = true;
    while (hasMore) {
      request.setMarker(marker);
      DescribeFolderContentsResult result = workDocsClient.describeFolderContents(request);
      docs.addAll(result.getDocuments());
      folders.addAll(result.getFolders());
      marker = result.getMarker();
      if (marker == null) {
        hasMore = false;
        break;
      }
    }
    for (FolderMetadata folder : folders)
      System.out.println("Folder:"+folder.getName());
    for (DocumentMetadata doc : docs)
      System.out.println("Document:"+doc.getLatestVersionMetadata().getName());
  }
}
```

Upload a Document

Use the following procedure to upload a document to Amazon WorkDocs.

To upload a document

1. Create an instance of `AmazonWorkDocsClient` as follows:

```
BasicSessionCredentials awsCreds = new BasicSessionCredentials (
  credentials.getAccessKeyId(),
  credentials.getSecretAccessKey(),
  credentials.getSessionToken()
);

AmazonWorkDocsClient amazonWorkDocsClient = new AmazonWorkDocsClient(awsCreds);
```

```
amazonWorkDocsClient.setEndpoint("url");
```

2. Get the signed URL for the upload as follows:

```
InitiateDocumentVersionUploadRequest request = new
 InitiateDocumentVersionUploadRequest();
request.setParentFolderId("parent-folder-id");
request.setName("my-document-name");
request.setContentType("application/octet-stream")
InitiateDocumentVersionUploadResult result =
 amazonWorkDocsClient.initiateDocumentVersionUpload(request);
UploadMetadata uploadMetadata = result.getUploadMetadata();
String documentId = result.getMetadata().getId();
String documentVersionId = result.getMetadata().getLatestVersionMetadata().getId();
String uploadUrl = uploadMetadata.getUploadUrl();
```

3. Upload the document using the signed URL as follows:

```
URL url = new URL(uploadUrl);
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
connection.setDoOutput(true);
connection.setRequestMethod("PUT");
// Content-Type supplied here should match with the Content-Type set
// in the InitiateDocumentVersionUpload request.
connection.setRequestProperty("Content-Type","application/octet-stream");
connection.setRequestProperty("x-amz-server-side-encryption", "AES256");
File file = new File("/path/to/file.txt");
FileInputStream fileInputStream = new FileInputStream(file);
OutputStream outputStream = connection.getOutputStream();
com.amazonaws.util.IOUtils.copy(fileInputStream, outputStream);
connection.getResponseCode();
```

4. Complete the upload process by changing the document status to `ACTIVE` as follows:

```
UpdateDocumentVersionRequest request = new UpdateDocumentVersionRequest();
request.setDocumentId("document-id");
request.setVersionId("document-version-id");
request.setVersionStatus(DocumentVersionStatus.ACTIVE);
amazonWorkDocsClient.updateDocumentVersion(request);
```

Download a Document

To download a document from Amazon WorkDocs, get a URL for the download as follows, and then use the API actions provided by your development platform to download the file using the URL.

```
GetDocumentVersionRequest request = new GetDocumentVersionRequest();
request.setDocumentId("document-id");
request.setVersionId("document-version-id");
request.setFields("SOURCE");
GetDocumentVersionResult result = amazonWorkDocsClient.getDocumentVersion(request);
String downloadUrl =
 result.getMetadata().getSource().get(DocumentSourceType.ORIGINAL.name());
```

Subscribe to Notifications

You can subscribe to notifications that Amazon WorkDocs sends when specific actions occur.

To subscribe to WorkDocs notifications

1. Prepare your endpoint to process Amazon SNS messages. For more information, see Make sure your endpoint is ready to process Amazon SNS messages in the *Amazon Simple Notification Service Developer Guide*.
2. Enable notifications for the IAM role that your application is using. See Managing Notifications for an IAM User or a Role (p. 12).
3. Create the subscription request as follows:

```
CreateNotificationSubscriptionRequest request = new
 CreateNotificationSubscriptionRequest();
request.setOrganizationId("d-1234567890");
request.setProtocol(SubscriptionProtocolType.Https);
request.setEndpoint("https://my-webhook-service.com/webhook");
request.setSubscriptionType(SubscriptionType.ALL);
CreateNotificationSubscriptionResult result =
 amazonWorkDocsClient.createNotificationSubscription(request);
System.out.println("WorkDocs notifications subscription-id: "
 result.getSubscription().getSubscriptionId());
```

SNS Notifications

The message includes the following information:

- `organizationId` — The ID of the organization.
- `parentEntityType` — The type of the parent (`Document` | `DocumentVersion` | `Folder`).
- `parentEntityId` — The ID of the parent.
- `entityType` — The type of the entity (`Document` | `DocumentVersion` | `Folder`).
- `entityId` — The ID of the entity.
- action — The action, which can be one of the following values:
  - `delete_document`
  - `move_document`
  - `recycle_document`
  - `rename_document`
  - `revoke_share_document`
  - `share_document`
  - `upload_document_version`

You can obtain a Amazon WorkDocs Organization ID from the AWS console using the following steps:

Obtaining an Organization Id

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. The **Directory ID** corresponding to your Amazon WorkDocs site is the Organization ID for that site.

Creating a New User

The following code snippet demonstrates the request construction for creating a new user in Amazon WorkDocs. Note that this is not a valid operation for a Connected AD configuration. If you want to add a new user to Amazon WorkDocs in the Connected AD configuration then the user must already be

present in the enterprise directory and then you would make a call to the ActivateUser API to activate the specified user in Amazon WorkDocs.

In the example below we are creating a new user with a storage quota of 1 gigabytes.

```
CreateUserRequest request = new CreateUserRequest();
    request.setGivenName("GivenName");
    request.setOrganizationId("d-12345678c4");
    // Passwords should:
    //   Be between 8 and 64 characters
    //   Contain three of the four below:
    //   A Lowercase Character
    //   An Uppercase Character
    //   A Number
    //   A Special Character
    request.setPassword("Badpa$$w0rd");
    request.setSurname("surname");
    request.setUsername("UserName");
    StorageRuleType storageRule = new StorageRuleType();
    storageRule.setStorageType(StorageType.QUOTA);
    storageRule.setStorageAllocatedInBytes(new Long(1048576l));
    request.setStorageRule(storageRule);
    CreateUserResult result = workDocsClient.createUser(request);
```

You can obtain a Amazon WorkDocs Organization ID from the AWS console using the following steps:

Obtaining an Organization Id

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. The **Directory ID** corresponding to your Amazon WorkDocs site is the Organization ID for that site.

Adding Permissions to User on a Resource

The following code snippet illustrates the request construction for adding permissions for a user on a resource. In this example we are adding `CONTRIBUTOR` permissions to a `USER` on a resource. This API can also be used to give permissions to a User or Group for a Folder or Document.

```
AddResourcePermissionsRequest request = new AddResourcePermissionsRequest();
    request.setResourceId("resource-id");
    Collection<SharePrincipal> principals = new ArrayList<>();;
    SharePrincipal principal = new SharePrincipal();
    principal.setId("user-id");
    principal.setType(PrincipalType.USER);
    principal.setRole(RoleType.CONTRIBUTOR);
    principals.add(principal);
    request.setPrincipals(principals);
    AddResourcePermissionsResult result = workDocsClient.addResourcePermissions(request);
```

Authentication and Access Control for Amazon WorkDocs APIs

Amazon WorkDocs admin APIs are authenticated and authorized through IAM policies. IAM administrators can create an IAM policy and attach it to an IAM role or user that can be used by the developer to access the API.

The following are provided as examples:

Tasks

Grant Permission to the Amazon WorkDocs API for a Developer on the AWS Account

As an IAM administrator if you want to grant Amazon WorkDocs API access to an IAM user from the same AWS account, you will need to create a Amazon WorkDocs API permission policy and attach it to the IAM user. The following is a sample Amazon WorkDocs API policy that grants permission to read-only APIs (List and Describe APIs).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "WorkDocsAPIReadOnly",
            "Effect": "Allow",
            "Action": [
                "workdocs:Get*",
      "workdocs:Describe*"
            ],
            "Resource": [
```

```
                "*"
            ]
        }
    ]
}
```

Grant Permission to Amazon WorkDocs API for Third Party Developer AWS Account

If you want to grant access to third party developers or users from your own organization but from a different AWS account, then you will have to create an IAM role and attach Amazon WorkDocs API allow policies.

This form of access is required in the following scenarios:

- Developer belongs to the same organization but the developer's AWS account is different from the Amazon WorkDocs AWS account.
- When an enterprise would like to grant Amazon WorkDocs API access to third party application developers.

In both of these scenarios, there are two AWS accounts involved, a developer's AWS account and a different account hosting a Amazon WorkDocs site.

The developer will need to provide the following information so the account administrator can create the IAM role:

- Your AWS account ID
- A unique `External ID` that your customer will use to identify you. For more information, see How to Use an External ID When Granting Access to Your AWS Resources to a Third Party.
- A list of Amazon WorkDocs APIs your application needs access to. IAM based policy control provides granular control, the ability to define allow or deny policies at the individual API level. For the list of Amazon WorkDocs APIs, see Amazon WorkDocs API Reference.

The following procedure describes steps involved in configuring IAM for cross-account access.

Configuring IAM for Cross-Account Access

1. Create a Amazon WorkDocs API permission policy, call it `WorkDocsAPIReadOnly` policy.
2. Create a new role in the IAM console of the AWS account hosting the Amazon WorkDocs site:
    a. Sign in to the IAM console at https://console.aws.amazon.com/iam/.
    b. In the navigation pane of the console, click **Roles** and then click **Create New Role**.
    c. For **Role name**, type a role name to help identify the purpose of this role, for example `workdocs_app_role`. Role names must be unique within your AWS account. After you enter the name, click **Next Step**.
    d. On the **Select Role Type** page, select the **Role for Cross-Account Access** section, and then select the type of role that you want to create:
        - Select **Provide access between AWS accounts you own** if you are the administrator of both the user account and the resource account, or both accounts belong to the same company. This is also the option to select when the users, role, and resource to be accessed are all in the same account.

- Select **Provide access between your AWS account and a third party AWS account** if you are the administrator of the account that owns the Amazon WorkDocs site and you want to grant permissions to users from an Application developer account. This option requires you to specify an external ID (which the third party must provide to you) to provide additional control over the circumstances in which the third party can use the role to access your resources. For more information, see How to Use an External ID When Granting Access to Your AWS Resources to a Third Party.

e. On the next page, specify the AWS account ID to which you want to grant access to your resources and also enter **External ID** in case of third party access.

f. Click **Next Step** to attach a policy.

3. On the **Attach Policy** page, search for the Amazon WorkDocs API permission policy that was created earlier and select the box next to the policy and click **Next Step**.
4. Review the details, copy the role ARN for future reference and click **Create Role** to complete the creation of the role.
5. Share the role ARN with the developer. The following is an example of the role ARN:

```
arn:aws:iam::AWS-ACCOUNT-ID:role/workdocs_app_role
```

Grant Permission to a Developer to Assume an IAM Role Given by a Amazon WorkDocs Customer

As an adminstrator of a developer's AWS account, to grant a user permission to switch to a role you can create a new policy and attach it to a user.

A policy that grants a user permission to assume a role must include a statement with the `Allow` effect on the `sts:AssumeRole` action and the Amazon Resource Name (ARN) of the role in a `Resource` element, as shown in the following example. Users that get the policy (either through group membership or directly attached) are allowed to switch to the specified role.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::<aws_account_id>:role/ workdocs_app_role"
  }
}
```

Restricting Access to a Specific Amazon WorkDocs Instance

If you have multiple Amazon WorkDocs sites on an AWS account and you want to grant API access to a specific site, you can do so by defining a condition element. The `Condition` element lets you specify conditions for when a policy is in effect.

The following is an example of a condition element:

```
    "Condition":
```

```
{
            "StringEquals": {
                "Resource.OrganizationId": "d-123456789c5"
            }
}
```

With the above condition in place in a policy, users are allowed to access only the Amazon WorkDocs instance with Id `d-123456789c5`. Amazon WorkDocs Instance Id is sometimes referred as Organization Id or Directory Id.

An Organization ID is also referred to as a Directory ID or an Instance ID. It can be used to restrict access to one or more Amazon WorkDocs sites on an account. For more information, see Restricting Access to a Specific Amazon WorkDocs Instance (p. 11).

You can obtain a Amazon WorkDocs Organization ID from the AWS console using the following steps:

Obtaining an Organization Id

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. The **Directory ID** corresponding to your Amazon WorkDocs site is the Organization ID for that site.

Managing Notifications for an IAM User or a Role

IAM Administrators can enable or disable notifications in Amazon WorkDocs through the IAM Console. Note that even if there is an explicit allow policy attached to a user or role granting access to Notification APIs, for example by adding `workdocs:CreateNotificationSubscription` in the allowed action, Administrators still have to explicitly enable Notifications for this specific User or Role ARN through the IAM console.

Unless Notifications are explicitly enabled for a user or a role ARN through the IAM console, the applications using the user or role credentials would not be able to make calls to `CreateNotificationSubscription` to subscribe and receive notifications.

To enable notifications

1. Open the Amazon WorkDocs console at https://console.aws.amazon.com/zocalo/.
2. On the **Manage Your WorkDocs Sites** page, select the desired directory and choose **Actions** and then **Manage Notifications**.
3. On the **Manage Notifications** page, choose **Enable Notifications**.
4. Enter the ARN for the user or role you want to allow to receive notifications from your Amazon WorkDocs site.

To disable notifications

1. Open the Amazon WorkDocs console at https://console.aws.amazon.com/zocalo/.
2. On the **Manage Your WorkDocs Sites** page, select the desired directory and choose **Actions** and then **Manage Notifications**.
3. On the **Manage Notifications** page, select the ARN that you wish to disable notifications for and choose **Disable Notifications**.