Amazon WorkDocs

Developer Guide

Amazon WorkDocs: Developer Guide

Table of Contents

What is Amazon WorkDocs?

Amazon WorkDocs is a document storage, collaboration, and sharing system. Amazon WorkDocs is fully managed, secure, and enterprise scale. It provides strong administrative controls, plus feedback capabilities that help improve user productivity. Files are stored in the cloud, safely and securely. Your user's files are only visible to them, and their designated contributors and viewers. Other members of your organization do not have access to other user's files unless they are specifically granted access.

Users can share their 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 Amazon WorkDocs

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.
- Amazon WorkDocs Drive used to synchronize a folder on your Mac or Windows desktop with 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, in addition to self-paced labs to help sharpen your AWS skills and gain practical experience.
- AWS Developer Center – Explore tutorials, download tools, and learn about AWS developer events.
- AWS Developer Tools – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- Getting Started Resource Center – Learn how to set up your AWS account, join the AWS community, and launch your first application.
- Hands-On Tutorials – Follow step-by-step tutorials to launch your first application on AWS.
- 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 webpage 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.

> **Note**
> For greater security, create federated users instead of IAM users whenever possible.

Examples

Connect to Amazon WorkDocs with IAM user credentials and query for users

The following code shows how to use an IAM user's API credentials to make API calls. In this case the API user and the Amazon WorkDocs site belong to the same AWS account.

> **Note**
> For greater security, create federated users instead of IAM users whenever possible.

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

The code sample uses the DescribeUsers API to search for users, and obtain metadata for users. User metadata provides details such as first name, last name, 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.

The code requires that you obtain an Amazon WorkDocs Organization ID.

Follow these steps to obtain a Amazon WorkDocs organization ID from the AWS console:

To get an organization ID

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Note the **Directory ID** value that corresponds to your Amazon WorkDocs site. That is the Organization ID for the site.

The following example shows how to use IAM credentials to make API calls.

```
import java.util.ArrayList;
import java.util.List;
```

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.workdocs.AmazonWorkDocs;
import com.amazonaws.services.workdocs.AmazonWorkDocsClient;
import com.amazonaws.services.workdocs.model.DescribeUsersRequest;
import com.amazonaws.services.workdocs.model.DescribeUsersResult;
import com.amazonaws.services.workdocs.model.User;

public class GetUserDemo {

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

    AmazonWorkDocs workDocs =
        AmazonWorkDocsClient.builder().withCredentials(staticCredentialProvider)
            .withRegion(Regions.US_WEST_2).build();

    List<User> wdUsers = new ArrayList<>();
    DescribeUsersRequest request = new DescribeUsersRequest();

    // The OrganizationId used here is an example and it should be replaced
    // with the OrganizationId of your WorkDocs site.
    request.setOrganizationId("d-123456789c");
    request.setQuery("joe");

    String marker = null;
    do {
      request.setMarker(marker);
      DescribeUsersResult result = workDocs.describeUsers(request);
      wdUsers.addAll(result.getUsers());
      marker = result.getMarker();
    } while (marker != null);

    System.out.println("List of users matching the query string: joe ");

 for (User wdUser : wdUsers) {
      System.out.printf("Firstname:%s | Lastname:%s | Email:%s | root-folder-id:%s\n",
          wdUser.getGivenName(), wdUser.getSurname(), wdUser.getEmailAddress(),
          wdUser.getRootFolderId());
    }
  }
}
```

Connecting to Amazon WorkDocs by assuming a role

This example uses the AWS Java SDK to assume a role and and use the role's temporary security credentials to access Amazon WorkDocs. The code sample uses the DescribeFolderContents API to list the items in a user's folder.

```
import java.util.ArrayList;
import java.util.List;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
```

```
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.AssumeRoleResult;
import com.amazonaws.services.workdocs.AmazonWorkDocs;
import com.amazonaws.services.workdocs.AmazonWorkDocsClient;
import com.amazonaws.services.workdocs.model.DescribeFolderContentsRequest;
import com.amazonaws.services.workdocs.model.DescribeFolderContentsResult;
import com.amazonaws.services.workdocs.model.DocumentMetadata;
import com.amazonaws.services.workdocs.model.FolderMetadata;

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

  public static void main(String[] args) throws Exception {

    AWSCredentials longTermCredentials =
        new BasicAWSCredentials("accessKey", "secretKey");

    // 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.

    AWSSecurityTokenService stsClient =
        AWSSecurityTokenServiceClientBuilder.standard()
            .withCredentials(new AWSStaticCredentialsProvider(longTermCredentials))
            .withRegion(Regions.DEFAULT_REGION.getName()).build();;

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

    AssumeRoleResult assumeResult = stsClient.assumeRole(assumeRequest);

    // AssumeRole returns temporary security credentials for the
 // workdocs-readonly-role

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

    // Build WorkDocs client using the temporary credentials.
    workDocs =
        AmazonWorkDocsClient.builder()
            .withCredentials(new AWSStaticCredentialsProvider(temporaryCredentials))
            .withRegion(Regions.US_WEST_2).build();


    // Invoke 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("root-folder-id");
  }

  private static void describeFolder(String folderId) {
    DescribeFolderContentsRequest request = new DescribeFolderContentsRequest();
```

```
    request.setFolderId(folderId);
    request.setLimit(2);
    List<DocumentMetadata> documents = new ArrayList<>();
    List<FolderMetadata> folders = new ArrayList<>();

    String marker = null;

    do {
      request.setMarker(marker);
      DescribeFolderContentsResult result = workDocs.describeFolderContents(request);
      documents.addAll(result.getDocuments());
      folders.addAll(result.getFolders());
      marker = result.getMarker();
    } while (marker != null);

    for (FolderMetadata folder : folders)
      System.out.println("Folder:" + folder.getName());
    for (DocumentMetadata document : documents)
      System.out.println("Document:" + document.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:

   If you use IAM user credentials, refer to Connect to Amazon WorkDocs with IAM user credentials and query for users (p. 3). If you assume an IAM role, refer to Connecting to Amazon WorkDocs by assuming a role (p. 4) for more information.

   Note
   For greater security, create federated users instead of IAM users whenever possible.

```
AWSCredentials longTermCredentials =
  new BasicAWSCredentials("accessKey", "secretKey");
AWSStaticCredentialsProvider staticCredentialProvider =
  new AWSStaticCredentialsProvider(longTermCredentials);

// Use the region specific to your WorkDocs site.
AmazonWorkDocs amazonWorkDocsClient =
  AmazonWorkDocsClient.builder().withCredentials(staticCredentialProvider)
    .withRegion(Regions.US_WEST_2).build();
```

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());
```

Setting up notifications for an IAM user or role

To create and manage notifications in Amazon WorkDocs, administrators use the IAM and Amazon WorkDocs consoles. You use the IAM console to set user permissions, and you use the Amazon WorkDocs console to enable notifications. Once you enable notifications, you then subscribe to them. Follow these steps.

Note
For greater security, create federated users instead of IAM users whenever possible.

To set IAM user permissions

- Use the IAM console to set the following permissions for the user:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
        "Effect": "Allow",
        "Action": [
            "workdocs:CreateNotificationSubscription",
            "workdocs:DeleteNotificationSubscription",
            "workdocs:DescribeNotificationSubscriptions"
            ],
        "Resource": "*"
        }
    ]
}
```

To enable notifications

Enabling notifications allows you to call CreateNotificationSubscription after you subscribe to 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.

For information about enabling Amazon WorkDocs to use notifications, see Using the Amazon WorkDocs API with the AWS SDK for Python and AWS Lambda. Once you enable notifications, you and your user can subscribe to them.

To subscribe to WorkDocs notifications

1. Prepare your endpoint to process Amazon SNS messages. For more information, see Fanout to HTTP/S endpoints in the *Amazon Simple Notification Service Developer Guide*.

   Important
   SNS sends a confirmation message to your configured endpoint. You *must* confirm this message in order to receive notifications. Also, if you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

2. Do the following:

   - Get an organization ID
     1. In the AWS Directory Service console navigation pane, select **Directories**.
     2. The **Directory ID** corresponding to your Amazon WorkDocs site also serves as the Organization ID for that site.
   - 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`

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**.

Creating a user

The following example show to create a user in Amazon WorkDocs.

> **Note**
> This is not a valid operation for a Connected AD configuration. To create a user in the Connected AD configuration, the user must already be present in the enterprise directory. Then, you must make a call to the ActivateUser API to activate the user in Amazon WorkDocs.

The following example demonstrates how to create a user with a storage quota of 1 gigabyte.

```
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);
```

Follow these steps to obtain a Amazon WorkDocs organization ID from the AWS console:

To get an organization ID

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Note the **Directory ID** value that corresponds to your Amazon WorkDocs site. That is the Organization ID for the site.

Granting users permissions to a resource

The following example shows how to use the AddResourcePermissions API to grant `CONTRIBUTOR` permissions to a USER on a resource. You can also use the API to give permissions to a user or group on 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 administrative applications

Amazon WorkDocs administrative 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

Granting developers permissions to the Amazon WorkDocs API

Note
For greater security, create federated users instead of IAM users whenever possible.

If you are an IAM administrator, you can grant Amazon WorkDocs API access to an IAM user from the same AWS account. To do this, create a Amazon WorkDocs API permission policy and attach it to the IAM user. The following API policy grants read-only permission to the various `Describe` APIs.

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

Granting third-party developers permission to the Amazon WorkDocs APIs

You can grant access to third-party developers, or to users who are using a different AWS account. To do this, 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.

To configure 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 AWS Management Console and open 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
```

Granting users permission to assume an IAM role

A developer with an administrative AWS account can allow a user to assume an IAM role. To do that, you create a new policy and attach it to that user.

The policy must include a statement with the `Allow` effect on the `sts:AssumeRole` action, plus 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 direct attachment, can 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 define a `Condition` element. The `Condition` element lets you specify conditions for when a policy is in effect.

The following example shows a condition element:

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

With the above condition in place in a policy, users can only access the Amazon WorkDocs instance with the ID of `d-123456789c5`. Amazon WorkDocs Instance ID is sometimes referred as Organization ID or Directory ID. For more information, see Restricting access to a specific Amazon WorkDocs instance (p. 13).

Follow these steps to obtain a Amazon WorkDocs organization ID from the AWS console:

To get an organization ID

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Note the **Directory ID** value that corresponds to your Amazon WorkDocs site. That is the Organization ID for the site.

Authentication and access control for user applications

Amazon WorkDocs user level applications are registered and managed through the Amazon WorkDocs console. Developers should register their applications on the `My Applications` page on the Amazon WorkDocs console which will provide unique IDs for each application. During registration, developers should specify redirect URIs where they will receive access tokens as well as application scopes.

Currently, applications can only access Amazon WorkDocs sites within the same AWS account where they are registered.

Contents

Granting permissions to call the Amazon WorkDocs APIs

Command line interface users must have full permissions to Amazon WorkDocs and AWS Directory Service. Without the permissions, any API calls return **UnauthorizedResourceAccessException** messages. The following policy grants full permissions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
          "workdocs:*",
          "ds:*",
          "ec2:CreateVpc",
          "ec2:CreateSubnet",
          "ec2:CreateNetworkInterface",
          "ec2:CreateTags",
          "ec2:CreateSecurityGroup",
          "ec2:DescribeVpcs",
          "ec2:DescribeSubnets",
          "ec2:DescribeNetworkInterfaces",
          "ec2:DescribeAvailabilityZones",
          "ec2:AuthorizeSecurityGroupEgress",
          "ec2:AuthorizeSecurityGroupIngress",
          "ec2:DeleteSecurityGroup",
          "ec2:DeleteNetworkInterface",
          "ec2:RevokeSecurityGroupEgress",
```

```
          "ec2:RevokeSecurityGroupIngress"
          ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

If you want to grant read-only permissions, use this policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
          "workdocs:Describe*",
          "ds:DescribeDirectories",
          "ec2:DescribeVpcs",
          "ec2:DescribeSubnets"
          ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

In the policy, the first action grants access to all the Amazon WorkDocs `Describe` operations. The `DescribeDirectories` action obtains information about your AWS Directory Service directories. The Amazon EC2 operations enable Amazon WorkDocs to obtain a list of your VPCs and subnets.

Using folder IDs in API calls

Whenever an API call accesses a folder, you must use the folder ID, not the folder name. For example, if you pass `client.get_folder(FolderId='MyDocs')`, the API call returns an **UnauthorizedResourceAccessException** message and the following 404 message.

```
client.get_folder(FolderId='MyDocs')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Users\user-name\AppData\Local\Programs\Python\Python36-32\lib\site-packages
\botocore\client.py", line 253, in _api_call
    return self._make_api_call(operation_name, kwargs)
  File "C:\Users\user-name\AppData\Local\Programs\Python\Python36-32\lib\site-packages
\botocore\client.py", line 557, in _make_api_call
    raise error_class(parsed_response, operation_name)
botocore.errorfactory.UnauthorizedResourceAccessException: An error occurred
 (UnauthorizedResourceAccessException) when calling the GetFolder operation:
Principal [arn:aws:iam::395162986870:user/Aman] is not allowed to execute
 [workdocs:GetFolder] on the resource.
```

To avoid that, use the ID in the folder's URL.

site.workdocs/index.html#/folder/
abc123def456ghi789jkl789mno4be7024df198736472dd50ca970eb22796082e3d489577.

Passing that ID returns a correct result.

```
client.get_folder(FolderId='abc123def456ghi789jkl789mno4be7024df198736472dd50ca970eb22796082e3d489577')
```

```
{'ResponseMetadata': {'RequestId': 'f8341d4e-4047-11e7-9e70-afa8d465756c',
 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'f234564e-1234-56e7-89e7-
a10fa45t789c', 'cache-control': 'private, no-cache, no-store, max-age=0',
 'content-type': 'application/json', 'content-length': '733', 'date':
 'Wed, 24 May 2017 06:12:30 GMT'}, 'RetryAttempts': 0}, 'Metadata': {'Id':
 'abc123def456ghi789jkl789mno4be7024df198736472dd50ca970eb22796082e3d489577', 'Name':
 'sentences', 'CreatorId': 'S-1-5-21-2125721135-1643952666-3011040551-2105&d-906724f1ce',
 'ParentFolderId': '0a811a922403ae8e1d3c180f4975f38f94372c3d6a2656c50851c7fb76677363',
 'CreatedTimestamp': datetime.datetime(2017, 5, 23, 12, 59, 13, 8000, tzinfo=tzlocal()),
 'ModifiedTimestamp': datetime.datetime(2017, 5, 23, 13, 13, 9, 565000, tzinfo=tzlocal()),
 'ResourceState': 'ACTIVE', 'Signature': 'b7f54963d60ae1d6b9ded476f5d20511'}}
```

Create an application

As an Amazon WorkDocs administrator, create your application using the following steps.

To create an application

1. Open the Amazon WorkDocs console at https://console.aws.amazon.com/zocalo/.
2. Choose **My Applications**, **Create an Application**.
3. Enter the following values:

   Application Name

   Name for the application.

   Email

   Email address to associate with the application.

   Application Description

   Description for the application.

   Redirect URIs

   The location that you want Amazon WorkDocs to redirect traffic to.

   Application Scopes

   The scope, either read or write, that you wish your application to have. For more details, see Application scopes (p. 16).
4. Choose **Create**.

Application scopes

Amazon WorkDocs supports the following application scopes:

- Content Read (`workdocs.content.read`), which gives your application access to the following Amazon WorkDocs APIs:
  - Get*
  - Describe*
- Content Write (`workdocs.content.write`), which gives your application access to the following Amazon WorkDocs APIs:
  - Create*
  - Update*

- Delete*
- Initiate*
- Abort*
- Add*
- Remove*

Authorization

After application registration is complete, an application can request authorization on behalf of any Amazon WorkDocs user. To do this, the application should visit the Amazon WorkDocs OAuth endpoint, `https://auth.amazonworkdocs.com/oauth`, and provide the following query parameters:

- [Required] `app_id`—Application ID generated when an application is registered.
- [Required] `auth_type`—The OAuth type for the request. Supported value is `ImplicitGrant`.
- [Required] `redirect_uri`—The redirect URI registered for an application to receive an access token.
- [Optional] `scopes`—A comma-deliminated list of scopes. If not specified, the list of scopes selected during registration will be used.
- [Optional] `state`—A string which is returned along with an access token.

> **Note**
> If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

A sample GET request to initiate the OAuth flow to obtain an access token:

```
GET https://auth.amazonworkdocs.com/oauth?app_id=my-app-
id&auth_type=ImplicitGrant&redirect_uri=https://myapp.com/
callback&scopes=workdocs.content.read&state=xyz
```

The following takes place during the OAuth authorization flow:

1. The application user is prompted to enter the Amazon WorkDocs site name.
2. The user is redirected to the Amazon WorkDocs authentication page to enter their credentials.
3. After successful authentication, the user is presented with the consent screen that allows the user to either grant or deny your application the authorization to access Amazon WorkDocs.
4. After the user chooses `Accept` on the consent screen, their browser is redirected to your application's callback URL along with the access token and region information as query parameters.

A sample GET request from Amazon WorkDocs:

```
GET https://myapp.com/callback?acessToken=accesstoken&region=us-east-1&state=xyz
```

In addition to the access token, the Amazon WorkDocs OAuth service also returns `region` as a query parameter for the selected Amazon WorkDocs site. External applications should use the `region` parameter to determine the Amazon WorkDocs service endpoint.

If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

Invoking Amazon WorkDocs APIs

After obtaining the access token, your application can make API calls to Amazon WorkDocs services.

Important

This example shows how to use a curl GET request to obtain a document's metadata.

```
Curl "https://workdocs.us-east-1.amazonaws.com/api/v1/documents/{document-id}" -H "Accept:
 application/json" -H "Authentication: Bearer accesstoken"
```

A sample JavaScript function to describe a user's root folders:

```
function printRootFolders(accessToken, siteRegion) {
    var workdocs = new AWS.WorkDocs({region: siteRegion});
    workdocs.makeUnauthenticatedRequest("describeRootFolders", {AuthenticationToken:
 accessToken}, function (err, folders) {
        if (err) console.log(err);
        else console.log(folders);
    });
}
```

A sample Java-based API invocation is described below:

```
AWSCredentialsProvider credentialsProvider = new AWSCredentialsProvider() {
  @Override
  public void refresh() {}

  @Override
  public AWSCredentials getCredentials() {
    new AnonymousAWSCredentials();
  }
};

// Set the correct region obtained during OAuth flow.
workDocs =
    AmazonWorkDocsClient.builder().withCredentials(credentialsProvider)
        .withRegion(Regions.US_EAST_1).build();

DescribeRootFoldersRequest request = new DescribeRootFoldersRequest();
request.setAuthenticationToken("access-token-obtained-through-workdocs-oauth");
DescribeRootFoldersResult result = workDocs.describeRootFolders(request);

for (FolderMetadata folder : result.getFolders()) {
  System.out.printf("Folder name=%s, Id=%s \n", folder.getName(), folder.getId());
}
```

Amazon WorkDocs Content Manager

Amazon WorkDocs Content Manager is a high-level utility tool that uploads content or downloads it from an Amazon WorkDocs site.

Topics

Constructing Amazon WorkDocs Content Manager

You can use Amazon WorkDocs Content Manager for administrative and user applications.

For user applications, a developer must construct Amazon WorkDocs Content Manager with anonymous AWS credentials and an authentication token.

For administrative applications, the Amazon WorkDocs client must be initialized with AWS Identity and Access Management (IAM) credentials. In addition, the authentication token must be omitted in subsequent API calls.

The following code demonstrates how to initialize Amazon WorkDocs Content Manager for user applications using Java or C#.

Java:

```
AWSStaticCredentialsProvider credentialsProvider = new AWSStaticCredentialsProvider(new
 AnonymousAWSCredentials());

AmazonWorkDocs client =
 AmazonWorkDocsClient.builder().withCredentials(credentialsProvider).withRegion("region").build();

ContentManager contentManager =
 ContentManagerBuilder.standard().withWorkDocsClient(client).withAuthenticationToken("token").build();
```

C#:

```
AmazonWorkDocsClient client = new AmazonWorkDocsClient(new AnonymousAWSCredentials(),
 "region");
ContentManagerParams params = new ContentManagerParams
{
WorkDocsClient = client,
AuthenticationToken = "token"
};
IContentManager workDocsContentManager = new ContentManager(param);
```

Downloading a document

Developers can use Amazon WorkDocs Content Manager to download a specific version or the latest version of a document from Amazon WorkDocs. The following examples demonstrate how to download a specific version of a document using Java and C#.

Note
To download the latest version of a document, do not specify the `VersionId` when constructing the `GetDocumentStream` request.

Java

```
ContentManager contentManager =
 ContentManagerBuilder.standard().withWorkDocsClient(client).withAuthenticationToken("auth-
token").build();

// Download document.
GetDocumentStreamRequest request = new GetDocumentStreamRequest();
request.setDocumentId("document-id");
request.setVersionId("version-id");

// stream contains the content of the document version.
InputStream stream = contentManager.getDocumentStream(request).getStream();
```

C#

```
ContentManager contentManager =
 ContentManagerBuilder.standard().withWorkDocsClient(client).withAuthenticationToken("auth-
token").build();

// Download document.
GetDocumentStreamRequest request = new GetDocumentStreamRequest();
request.setDocumentId("document-id");
request.setVersionId("version-id");

// stream contains the content of the document version.
InputStream stream = contentManager.getDocumentStream(request).getStream();
```

Uploading a document

Amazon WorkDocs Content Manager provides an API for uploading content to an Amazon WorkDocs site. The following examples demonstrate how to upload a document using Java and C#.

Java

```
File file = new File("file-path");
InputStream stream = new FileInputStream(file);
UploadDocumentStreamRequest request = new UploadDocumentStreamRequest();
request.setParentFolderId("destination-folder-id");
request.setContentType("content-type");
request.setStream(stream);
request.setDocumentName("document-name");
contentManager.uploadDocumentStream(request);
```

C#

```
var stream = new FileStream("file-path", FileMode.Open);

UploadDocumentStreamRequest uploadDocumentStreamRequest = new UploadDocumentStreamRequest()
{
ParentFolderId = "destination-id",
DocumentName = "document-name",
ContentType = "content-type",
Stream = stream
```

```
};

workDocsContentManager.UploadDocumentStreamAsync(uploadDocumentStreamRequest).Wait();
```

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.