aws

開発者ガイド

Amazon WorkDocs

Amazon WorkDocs: 開発者ガイド

Table of Contents

注意: Amazon では、新しい顧客のサインアップとアカウントのアップグレードは利用できなくなりました WorkDocs。移行手順については、「Amazon からデータを移行する方法 WorkDocs」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。

Amazon WorkDocs とは何ですか？

Amazon WorkDocs は、ドキュメントストレージ、コラボレーション、および共有システムです。Amazon WorkDocs はフルマネージド型の安全なエンタープライズ規模です。強力な管理制御に加え、ユーザーの生産性向上に役立つフィードバック機能も備えています。ファイルは、クラウド内に安全に保存されます。ユーザーのファイルは、ユーザーのみ、またはユーザーが指定したコントリビューターとビューワーのみが閲覧できます。ユーザーの組織のその他の方は、ユーザーが特別なアクセス許可を付与しない限り、ユーザーのいずれのファイルへもアクセスすることができません。

ユーザーはコラボレーション、または、レビューの目的で、その他の方とファイルを共用することができます。Amazon WorkDocs クライアントアプリケーションは、ファイルのインターネットメディアタイプに応じて、さまざまな種類のファイルの表示に使用されます。Amazon WorkDocs では、一般的なドキュメントおよびイメージ形式がサポートされており、追加のメディアタイプのサポートは常に追加されています。

詳細は、「Amazon WorkDocs」を参照してください。

Amazon WorkDocs へアクセスする

エンドユーザーはクライアントアプリケーションを使用してファイルにアクセスします。管理者以外のユーザーは、Amazon WorkDocs コンソールまたは管理ダッシュボードを使用する必要はありません。Amazon WorkDocs には、いくつかの異なるクライアントアプリケーションとユーティリティが提供されています。

- ドキュメント管理とレビューに使用するウェブアプリケーション。
- ドキュメントレビューに使用するモバイルデバイス用ネイティブアプリケーション。
- Amazon WorkDocs Drive は、Mac や Windows のデスクトップ上のフォルダと Amazon WorkDocs のファイルを同期するために使用します。

料金

Amazon WorkDocs には、料金前払いなどの誓約はありません。アクティブなユーザーアカウントと、使用するストレージに対する料金のみです。 詳細については、「料金」をご参照ください。

リソース

このサービスを利用する際に役立つ関連リソースは次のとおりです。

- クラスとワークショップ – AWS のスキルを磨き、実践的な経験が得るために役立つセルフペースラボに加えて、ロールベースのコースと特別コースへのリンクです。
- AWS デベロッパーセンター – チュートリアルの検索、ツールのダウンロード、AWS デベロッパーイベントの確認を行います。
- AWS デベロッパーツール - AWS アプリケーションを開発および管理するためのデベロッパーツール、SDK、IDE ツールキット、およびコマンドラインツールへのリンクです。
- ご利用開始のためのリソースセンター — AWS アカウント をセットアップする方法、AWS コミュニティに参加する方法、最初のアプリケーションを起動する方法を説明します。
- ハンズオンチュートリアル - ステップ バイ ステップのチュートリアルに従って、最初のアプリケーションを AWS で起動します。
- AWS ホワイトペーパー – アーキテクチャ、セキュリティ、エコノミクスなどのトピックについて、AWS のソリューションアーキテクトや他の技術エキスパートが記述した AWS の技術ホワイトペーパーの包括的なリストへのリンクです。
- AWS Support センター – AWS Support のケースを作成して管理するためのハブです。フォーラム、技術上のよくある質問、サービスヘルスステータス、AWS Trusted Advisor など、他の役立つリソースへのリンクも含まれています。
- AWS Support – AWS Support に関する情報のメインウェブページです。クラウド内でのアプリケーションの構築および実行を支援するために 1 対 1 での迅速な対応を行うサポートチャネルとして機能します。
- お問い合わせ - AWS の請求、アカウント、イベント、不正使用、その他の問題などに関するお問い合わせの受付窓口です。
- AWS サイトの利用規約 – 当社の著作権、商標、お客様のアカウント、ライセンス、サイトへのアクセス、その他のトピックに関する詳細情報。

はじめに

以下のコードスニペットは、Amazon WorkDocs SDK の使用開始を促進することができます。

> **Note**
>
> セキュリティを強化するために、可能な限り IAM ユーザーの代わりにフェデレーティッドユーザーを作成してください。

例

- IAM ユーザー認証情報を使用して Amazon WorkDocs に接続し、ユーザーにクエリを実行します。
- ロールを引き受けて Amazon WorkDocs に接続する
- ドキュメントをアップロードする
- ドキュメントをダウンロードする
- IAM ユーザーまたはロールの通知を設定する
- ユーザーを作成する
- リソースへのアクセス許可をユーザーに付与する

IAM ユーザー認証情報を使用して Amazon WorkDocs に接続し、ユーザーにクエリを実行します。

次のコードは、IAM ユーザーの API 認証情報を使用して API 呼び出しを行う方法を示しています。この場合、API ユーザーと Amazon WorkDocs サイトは、同じ AWS アカウントに属しています。

> **Note**
>
> セキュリティを強化するために、可能な限り IAM ユーザーではなくフェデレーティッドユーザーを作成してください。

IAM ユーザーに適切な IAM ポリシーを介して Amazon WorkDocs API アクセスが付与されていることを確認します。

コード例は、ユーザーを検索し、ユーザーにメタデータを取得するため、describeUsers API を使用します。ユーザーメタデータは、名、姓、ユーザー ID およびルートフォルダ ID などの詳細を提供します。ルートフォルダ ID は、ユーザーに代わってコンテンツのアップロードまたはダウンロード操作を行う場合に特に役立ちます。

このコードでは、Amazon WorkDocs 組織 ID を取得する必要があります。

AWS コンソールから Amazon WorkDocs 組織 ID を取得するステップは、次のとおりです。

組織 ID を取得するには

1. AWS Directory Service コンソールのナビゲーションペインで、ディレクトリを選択します。
2. Amazon WorkDocs サイトに対応する [ディレクトリ ID] の値にご注意ください。これがサイトの組織 ID です。

次の例に、IAM 認証情報を使用して API 呼び出しを行う法を示します。

```
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());
    }
  }
}
```

ロールを引き受けて Amazon WorkDocs に接続する

この例では、AWS Java SDK を使用してロールの一時的セキュリティ認証情報を使用して Amazon WorkDocs にアクセスします。このサンプルコードでは、describeFolderContents API を使用して、ユーザーのフォルダに存在する項目のリストを作成します。

```
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());
  }
}
```

ドキュメントをアップロードする

以下の手順に従って、Amazon WorkDocs にドキュメントをアップロードします。

ドキュメントをアップロードするには

1. 次のように AmazonWorkDocsClient のインスタンスを作成します。

   IAM ユーザー認証情報を使用している場合、「IAM ユーザー認証情報を使用して Amazon WorkDocs に接続し、ユーザーにクエリを実行します。」をご参照ください。IAM ロールを割り当てている場合の詳細については「ロールを引き受けて Amazon WorkDocs に接続する」をご参照ください。

   > **Note**
   >
   > セキュリティを強化するために、可能な限り IAM ユーザーではなくフェデレーティッドユーザーを作成してください。

```
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. アップロードのため、以下のように署名付き URL を取得します。

```
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. 署名付き URL を使用して、以下のようにドキュメントをアップロードします。

```
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. 以下のようにドキュメントステータスを ACTIVE に変更して、アップロードプロセスを完了します。

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

ドキュメントをダウンロードする

Amazon WorkDocs からドキュメントをダウンロードするには、以下のようにダウンロード用の URL を取得し、その後、URL を使ってファイルをダウンロードするために、開発プラットフォームが提供する API アクションを使用します。

```
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());
```

IAM ユーザーまたはロールの通知を設定する

管理者は IAMと Amazon WorkDocs コンソールを使用して、Amazon WorkDocs で通知を作成および管理を行います。IAM コンソールを使用してユーザー権限を設定し、Amazon WorkDocs コンソールを使用して通知を有効にします。通知を有効にしてから、それをサブスクライブします。以下の手順に従ってください。

> **Note**
>
> セキュリティを強化するために、可能な限り IAM ユーザーではなくフェデレーティッドユーザーを作成してください。

IAM ユーザーの権限を設定するには

- IAM コンソールを使用して、ユーザーに対して次の権限を設定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Action": [
            "workdocs:CreateNotificationSubscription",
            "workdocs:DeleteNotificationSubscription",
            "workdocs:DescribeNotificationSubscriptions"
            ],
        "Resource": "*"
        }
    ]
}
```

通知を有効化するには

通知を有効にすると、通知をサブスクライブした後に、CreateNotificationSubscription への呼び出しを許可します。

1. Amazon WorkDocs コンソール (https://console.aws.amazon.com/zocalo/) を開きます。

2. [WorkDocs サイトを管理] ページで、目的のディレクトリを選択し、[アクション]、[通知を管理] の順に選択します。
3. [通知を管理] ページで、[通知を有効化] を選択します。
4. Amazon WorkDocs サイトから通知を受信を許可するユーザーまたはロールの ARN を入力します。

Amazon WorkDocs を有効化して、通知を使用できるようにするための情報は、「 AWS SDK for Python と AWS Lambda で Amazon WorkDocs APIを使用する」 をご参照ください。通知を有効にすると、自分と自身のユーザーは通知をサブスクライブできます。

WorkDocs の通知をサブスクライブするには

1. Amazon SNS メッセージを処理するため、エンドポイントを準備します。詳細については、「Amazon Simple Notification Service デベロッパーガイド」の「HTTP/S エンドポイントへのファンアウト」を参照してください。

   > **Important**
   >
   > SNS は、設定されたエンドポイントに確認メッセージを送信します。通知を受け取るには、このメッセージを確認する必要があります。また、コマンドラインインターフェイスまたは API を介して AWS にアクセスするときに FIPS 140-2 で検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格(FIPS) 140-2」をご参照ください。

2. 次のコマンドを実行します

   - 組織 ID を取得する
     1. [AWS Directory Service コンソール] ナビゲーションペインで、[ディレクトリ] を選択します。
     2. Amazon WorkDocs のサイトに対応する [ディレクトリ ID] は、そのサイトの組織 ID としても機能します。
   - サブスクリプションリクエストを次のように作成します。

```
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通知

メッセージには、次に示す情報が含まれます。

- organizationId — 組織の ID。
- parentEntityType — 親のタイプ (Document | DocumentVersion | Folder)。
- parentEntityId — 親の ID。
- entityType — エンティティのタイプ (Document | DocumentVersion | Folder)。
- entityId — エンティティの ID。
- アクション — 次のいずれかの値になるアクションです。
  - delete_document
  - move_document
  - recycle_document
  - rename_document
  - revoke_share_document
  - share_document
  - upload_document_version

通知を無効化するには

1. Amazon WorkDocs コンソール (https://console.aws.amazon.com/zocalo/) を開きます。
2. [WorkDocsサイトの管理] ページで、目的のディレクトリを選択し、[アクション] 、[通知を管理] の順に選択します。
3. [通知を管理] ページで、通知を無効にする ARN を選択し、[通知を無効化] を選択します。

ユーザーを作成する

次の例に、Amazon WorkDocs で新しいユーザーを作成するための法を示します。

> **Note**
>
> これは、 Connected AD 設定の有効なオペレーションではありません。Connected AD 構成でユーザーを作成するには、ユーザーがエンタープライズディレクトリにすでに存在している必要があります。次に、ActivateUser API を呼び出して、Amazon WorkDocs でユーザーをアクティブ化する必要があります。

次の例では、1 ギガバイトのストレージクォータを持つユーザーを作成する方法を説明しています。

```
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);
```

AWS コンソールから Amazon WorkDocs 組織 ID を取得するステップは、次のとおりです。

組織 ID を取得するには

1. AWS Directory Service コンソールのナビゲーションペインで、ディレクトリを選択します。
2. Amazon WorkDocs サイトに対応する [ディレクトリ ID] の値にご注意ください。これがサイトの組織 ID です。

リソースへのアクセス許可をユーザーに付与する

次の例は、AddResourcePermissions API を使用してリソースのUSERへのCONTRIBUTOR権限を付与する方法を示しています。API を使用して、フォルダーまたはドキュメントに対するユーザーまたはグループにアクセス許可を与えることもできます。

```
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);
```

管理アプリケーションの認証とアクセス制御

Amazon WorkDocs 管理 API は IAM ポリシーによって認証および承認されます。IAM管理者は IAM ポリシーを作成し、開発者が API にアクセスするために使用できる IAM ロールまたはユーザーにアタッチすることができます。

例として以下のものが用意されています。

タスク

- デベロッパーに Amazon WorkDocs API へのアクセス権を付与する
- サードパーティデベロッパーに Amazon WorkDocs API へのアクセス許可を付与する
- IAM ロールを引き受けるアクセス許可をユーザーに付与する
- 特定の Amazon WorkDocs インスタンスへのアクセスを制限する

デベロッパーに Amazon WorkDocs API へのアクセス権を付与する

> **Note**
>
> セキュリティを強化するために、可能な限り IAM ユーザーではなくフェデレーティッドユーザーを作成してください。

IAM 管理者の場合は、同じ AWS アカウントの IAM ユーザーに Amazon WorkDocs API アクセスを許可することができます。これを行うためには、Amazon WorkDocs API アクセス許可ポリシーを作成して、 ユーザーにアタッチします。次の API ポリシーは、さまざまな `Describe`API に読み取り専用許可を付与します。

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

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

サードパーティデベロッパーに Amazon WorkDocs API へのアクセス許可を付与する

アクセス権は、サードパーティーのデベロッパー、または別の AWS アカウントを使用しているユーザーに付与することができます。これを行うために、IAM ロールを作成し、Amazon WorkDocs API 許可ポリシーをアタッチします。

このアクセスのフォームは、以下のシナリオで必要です。

- 開発者は同じ組織に属しているが、開発者の AWS アカウントが Amazon WorkDocs AWS アカウントと異なっています。
- 企業がサードパーティーアプリケーション開発者に Amazon WorkDocs API アクセス権を付与したい場合。

この両方のシナリオには、開発者の AWS アカウントと、Amazon WorkDocs サイトをホストする別のアカウントの2つの AWS アカウントが関係しています。

開発者は、アカウント管理者が IAM ロールを作成できるように、以下の情報を提供する必要があります。

- あなたの AWS アカウント ID。
- 顧客がユーザーを識別するために使用するユニーク `External ID` 。詳細については、「AWS リソースへのアクセス権をサードパーティーに付与するときに外部 ID を使用する方法」を参照してください。
- アプリケーションがアクセスする必要がある Amazon WorkDocs API のリストです。IAM ベースのポリシー制御は、個々の API レベルで許可または拒否のポリシーを定義することができ、きめ細かい制御が可能です。Amazon WorkDocs API のリストについては、「Amazon WorkDocs API Reference」(Amazon WorkDocs API リファレンス) を参照してください。

次の手順は、クロスアカウントアクセスのための IAM の設定に含まれるステップについて説明しています。

クロスアカウントアクセスに対して IAM を設定するには

1. Amazon WorkDocs API アクセス許可ポリシーを作成し、これを `WorkDocsAPIReadOnly` ポリシーと呼ぶこととします。
2. Amazon WorkDocs サイトをホストしている AWS アカウントの IAM コンソールで新しいロールを作成します。

   a. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

   b. コンソールのナビゲーションペインで [Roles] (ロール) をクリックし、[Create New Role] (新しいロールの作成) をクリックします。

   c. [Role name] (ロール名) ボックスにそのロールの目的を見分けやすくするロール名を入力します。例えば、`workdocs_app_role` です。ロール名は AWS アカウント内でユニークでなければなりません。ロール名を入力したら、[Next step] (次のステップ) をクリックします。

   d. [Select Role Type] (ロールタイプの選択) ページで、[Role for Cross-Account Access] (クロスアカウントアクセスのロール) セクションを選択し、作成するロールのタイプを選択します。

      - お客様がユーザーアカウントとリソースアカウントの両方の管理者であるか、両方のアカウントが同じ会社に属している場合は、[所有する AWS アカウント間のアクセスを提供する] を選択します。このオプションは、ユーザー、ロール、アクセスされるリソースがすべて同じアカウントに属している場合にも選択します。
      - Amazon WorkDocs サイトを所有するアカウントの管理者で、アプリケーション開発者アカウントのユーザーにアクセス許可を付与したい場合は、「AWS アカウントとサードパーティの AWS アカウント間のアクセスを提供する」を選択します。このオプションでは、お客様は、第三者によって提供される外部 ID を指定して、第三者がロールを使用してお客様のリソースにアクセスできる環境に対して、管理性を強化する必要があります。詳細については、「How to Use an External ID When Granting Access to Your AWS Resources to a Third Party」(AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法) をご参照ください。

   e. 次のページで、リソースへのアクセスを付与する AWS アカウント ID を指定し、サードパーティーによるアクセスがあった場合に備えて [External ID(外部 ID)] も入力します。

   f. [Next Step] (次のステップ) をクリックして、ポリシーをアタッチします。

3. [Attach Policy] (ポリシーのアタッチ) ページで、以前に作成された Amazon WorkDocs API アクセス許可ポリシーを検索して、ポリシーの横にあるボックスをオンにし、[Next Step] (次のステップ) をクリックします。
4. 詳細を確認し、今後の参照用にロール ARN をコピーして、[Create Role] (ロールの作成) をクリックしてロールの作成を完了します。
5. ロール ARN を開発者と共有します。ロール ARN の例を以下に示します。

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

IAM ロールを引き受けるアクセス許可をユーザーに付与する

管理者AWSアカウントを持つ開発者は、ユーザーが IAM ロールを引き受けることを許可できます。そのためには、新しいポリシーを作成してそのユーザーにアタッチします。

次の例に示すように、ポリシーには、`sts:AssumeRole` アクションに対する `Allow` 効果を含むステートメントと、`Resource` 要素内のロールの Amazon リソースネーム (ARN) を含める必要があります。グループ メンバーシップまたは直接アタッチを通じてポリシーを取得したユーザーは、指定されたロールに切り替えることができます。

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

特定の Amazon WorkDocs インスタンスへのアクセスを制限する

AWS アカウントに複数の Amazon WorkDocs サイトがあり、特定のサイトへの API アクセスを許可したい場合は、`Condition` 要素を定義できます。`Condition` 要素は、ポリシーを実行するタイミングの条件を特定することができます。

次の例は、条件要素を示しています。

```
    "Condition":
```

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

上記の条件をポリシーに設定すると、ユーザーは d-123456789c5 の ID を持つ Amazon WorkDocs インスタンスにのみアクセスできます。Amazon WorkDocs インスタンス ID は、組織 ID またはディレクトリ ID と呼ばれることもあります。詳細については、「特定の Amazon WorkDocs インスタンスへのアクセスを制限する」を参照してください。

以下の手順で、 AWS コンソールから Amazon WorkDocs の組織 ID を取得します。

組織 ID を取得するには

1. AWS Directory Service コンソールのナビゲーションペインで、ディレクトリを選択します。
2. Amazon WorkDocs サイトに対応する [ディレクトリ ID] の値にご注意ください。これがサイトの組織 ID です。

ユーザーアプリケーションの認証とアクセス制御

Amazon WorkDocs ユーザーレベルのアプリケーションは、Amazon WorkDocs コンソールを介して登録および管理されています。開発者は、各アプリケーションに対する固有の ID が提供される、Amazon WorkDocs コンソールの `My Applications` ページでアプリケーションを登録する必要があります。登録時に、開発者はリダイレクト URI を指定して、アクセストークンとアプリケーションスコープを受け取る必要があります。

現在、アプリケーションは、登録されているのと同じ AWS アカウント内の Amazon WorkDocs サイトにのみアクセスできます。

目次

Amazon WorkDocs API を呼び出すためのアクセス権限の付与

コマンドラインインターフェイスのユーザーには、Amazon WorkDocs とへのフルアクセス権限が必要です。AWS Directory Service権限がないと、どの API 呼び出しでも不正リソースアクセス例外メッセージが返されます。次のポリシーは完全な権限を付与します。

```
{
  "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": "*"
      }
    ]
}
```

読み取り専用のアクセス許可を付与する場合は、このポリシーを使用します。

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

ポリシーでは、最初のアクションによってすべての Amazon WorkDocs `Describe` オペレーションへのアクセス権が付与されます。`DescribeDirectories` アクションはAWS Directory Service ディレクトリに関する情報を取得します。Amazon EC2 オペレーションにより、Amazon WorkDocs は VPC とサブネットのリストを取得できるようになります。

API 呼び出しでのフォルダー ID の使用

API 呼び出しがフォルダにアクセスするときは必ず、フォルダ名ではなくフォルダ ID を使用する必要があります。たとえば、`client.get_folder(FolderId='MyDocs')`を渡した場合、API 呼び出しは UnauthorizedResourceAccessException メッセージと次の 404 メッセージを返します。

```
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.
```

これを回避するには、フォルダーの URL にある ID を使用してください。

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

その ID を渡すと、正しい結果が返されます。

```
client.get_folder(FolderId='abc123def456ghi789jkl789mno4be7024df198736472dd50ca970eb22796082e3d
{'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'}}
```

アプリケーションの作成

Amazon WorkDocs 管理者として、以下のステップを使用してアプリケーションを作成します。

アプリケーションを作成するには

1. Amazon WorkDocs コンソール (https://console.aws.amazon.com/zocalo/) を開きます。
2. [マイアプリケーション]、[アプリケーションの作成]の順に選択します。
3. 次の値を入力します。

   アプリケーション名

   アプリケーションの名前。

   Email(メール)

   アプリケーションに関連付けるメールアドレス。

   Application Description(アプリケーションの説明)

   アプリケーションの説明。

   リダイレクト URI

   Amazon WorkDocs がトラフィックをリダイレクトする場所です。

   Application Scopes(アプリケーションスコープ)

   アプリケーションに設定するスコープ (読み取りまたは書き込みのいずれか)。詳細については、「アプリケーションスコープ」をご参照ください。
4. [作成] を選択します。

アプリケーションスコープ

Amazon WorkDocs は、以下のアプリケーションスコープをサポートしています。

- コンテンツを読み取り (`workdocs.content.read`)、アプリケーションに以下の Amazon WorkDocs API へのアクセス権を付与します。
  - 取得*
  - 説明*

- コンテンツを書き込み (workdocs.content.write)、アプリケーションに以下の Amazon WorkDocs API へのアクセス権を付与します。
  - 作成*
  - 更新*
  - 削除*
  - ジョブの開始*
  - 中止*
  - 追加*
  - 削除*

認可

アプリケーション登録が完了後、アプリケーションはすべての Amazon WorkDocs ユーザーに代わって承認をリクエストすることができます。これを行うには、アプリケーションは Amazon WorkDocs OAuth エンドポイントにアクセスし、https://auth.amazonworkdocs.com/oauth、以下のクエリパラメータを提供する必要があります。

- [必須] app_id — アプリケーションが登録されている場合に生成されるアプリケーション ID。
- [必須] auth_type — リクエストの OAuth タイプ。サポートされている値は ImplicitGrant です。
- [必須] redirect_uri — アプリケーションがアクセストークンを受信するために登録されたリダイレクト URI。
- [オプション] scopes — スコープのカンマ区切りのリスト。指定しない場合、登録時に選択されたスコープのリストが使用されます。
- [オプション] state — アクセストークンとともに返される文字列。

> **Note**
>
> コマンドラインインターフェイスまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、[連邦情報処理規格 (FIPS) 140-2] をご参照ください。

アクセストークン取得のための OAuth フローを開始するためのサンプル GET リクエスト。

```
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
```

以下は、OAuth 認証フロー中に行われます。

1. アプリケーションユーザーは、Amazon WorkDocs のサイト名を入力するように指示されます。
2. ユーザーは Amazon WorkDocs の認証ページにリダイレクトされ、認証情報を入力します。
3. 認証に成功すると、ユーザーには同意画面が表示され、ユーザーはアプリケーションに Amazon WorkDocs へのアクセス許可を付与または拒否できます。
4. ユーザーが同意画面で `Accept` を選択すると、ブラウザは、クエリパラメータとしてのアクセストークンとリージョン情報とともに、アプリケーションのコールバック URL にリダイレクトされます。

Amazon WorkDocs からの GET リクエストのサンプル。

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

アクセストークンに加えて、Amazon WorkDocsOAuth サービスは選択した Amazon WorkDocs サイトのクエリパラメータとして `region` も返します。外部アプリケーションは、Amazon WorkDocs サービスエンドポイントを確定するために、 `region` パラメータを使用する必要があります。

コマンドラインインターフェイスまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、[連邦情報処理規格 (FIPS) 140-2] をご参照ください。

Amazon WorkDocs API を呼び出す

アクセストークンの取得後、アプリケーションは Amazon WorkDocs サービスへの API 呼び出しを行うことができます。

> **Important**

この例は、curl GET リクエストを使用してドキュメントのメタデータを取得する方法を示しています。

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

ユーザーのルートフォルダを記述するサンプル JavaScript 関数。

```
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);
    });
}
```

Java ベースの API 呼び出しサンプルは、以下のとおりです。

```
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 コンテンツマネージャ

Amazon WorkDocs コンテンツマネージャーは、コンテンツをアップロード、または Amazon WorkDocs サイトからダウンロードしたりする高水準のユーティリティツールです。

トピック

Amazon WorkDocs コンテンツマネージャの構築

Amazon WorkDocs コンテンツマネージャは、管理用およびユーザーアプリケーションのに使用できます。

ユーザーアプリケーションの場合、開発者は匿名の AWS 認証情報と認証トークンを使用して Amazon WorkDocs コンテンツマネージャーを構築する必要があります。

管理アプリケーションの場合、Amazon WorkDocs クライアントは、AWS Identity and Access Management (IAM) 認証情報を使用して初期化する必要があります。さらに、それ以降の API 呼び出しでは、認証トークンを省略する必要があります。

次のコードは、 Java または C# を使用して、ユーザーアプリケーションの Amazon WorkDocs コンテンツマネージャを初期化する方法を明示しています。

Java:

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

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

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

C#:

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

ドキュメントのダウンロード

開発者は、Amazon WorkDocs コンテンツマネージャーを使用して、Amazon WorkDocs からドキュメントの特定バージョンまたは最新バージョンをダウンロードすることができます。以下の例では、Java および C# を使用してドキュメントの特定のバージョンをダウンロードします。

> **Note**
>
> ドキュメントの最新バージョンをダウンロードするには、VersionId リクエストの構築時に GetDocumentStream を指定しないでください。

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();
```

ドキュメントのアップロード

Amazon WorkDocs コンテンツマネージャーは、Amazon WorkDocs サイトにコンテンツをアップロードするための API を提供します。以下の例では、 Java および C# を使用してドキュメントをアップロードします。

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();
```