개발자 가이드

Amazon WorkDocs

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon WorkDocs: 개발자 가이드

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

..... iv
Amazon WorkDocs란 무엇입니까? 1
WorkDocs 액세스 1
요금 1
리소스 1
시작 3
IAM 사용자 자격 증명을 사용하여 WorkDocs에 연결 3
역할을 수임하여 WorkDocs에 연결 5
문서 업로드 7
문서 다운로드 9
알림 설정 10
사용자 생성 12
사용자에게 리소스에 대한 권한 부여 13
관리 애플리케이션에 대한 인증 및 액세스 제어 15
개발자에게 WorkDocs API에 대한 권한 부여 15
타사 개발자에게 WorkDocs APIs에 대한 권한 부여 16
사용자에게 IAM 역할을 맡을 수 있는 권한 부여 17
특정 WorkDocs 인스턴스에 대한 액세스 제한 18
사용자 애플리케이션에 대한 인증 및 액세스 제어 19
WorkDocs APIs를 호출할 수 있는 권한 부여 19
API 직접 호출 시 폴더 ID 사용 20
애플리케이션 만들기 21
애플리케이션 범위 22
권한 부여 23
WorkDocs APIs 호출 24
WorkDocs 콘텐츠 관리자 26
WorkDocs 콘텐츠 관리자 구성 26
문서 다운로드 27
문서 업로드 28

참고: Amazon WorkDocs에서는 새 고객 가입 및 계정 업그레이드를 더 이상 사용할 수 없습니다. 여기에서 마이그레이션 단계에 대해 알아봅니다. WorkDocs에서 데이터를 마이그레이션하는 방법.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.

Amazon WorkDocs란 무엇입니까?

Amazon WorkDocs는 문서 저장, 협업 및 공유 시스템입니다. WorkDocs는 완전 관리형의 안전한 엔터프라이즈 규모입니다. 강력한 관리 제어 기능과 사용자 생산성 향상에 도움이 되는 피드백 기능을 제공합니다. 파일은 클라우드에 안전하게 저장됩니다. 사용자의 파일은 소유자, 소유자가 지정한 기고자 및 최종 사용자에게만 표시됩니다. 소유자가 명확하게 액세스 권한을 부여하지 않는 한, 조직의 다른 구성원은 다른 사용자의 파일에 액세스할 수 없습니다.

사용자는 공동 작업이나 검토를 위해 조직의 다른 구성원과 파일을 공유할 수 있습니다. WorkDocs 클라이언트 애플리케이션은 파일의 인터넷 미디어 유형에 따라 다양한 유형의 파일을 보는 데 사용할 수 있습니다. WorkDocs는 모든 일반 문서 및 이미지 형식을 지원하며 추가 미디어 유형에 대한 지원이 지속적으로 추가되고 있습니다.

자세한 내용은 Amazon WorkDocs를 참조하세요.

WorkDocs 액세스

최종 사용자는 클라이언트 애플리케이션을 사용하여 관련 파일에 액세스합니다. 관리자가 아닌 사용자는 WorkDocs 콘솔 또는 관리 대시보드를 사용할 필요가 없습니다. WorkDocs는 다음과 같은 다양한 클라이언트 애플리케이션 및 유틸리티를 제공합니다.

- 문서 관리 및 검토에 사용되는 웹 애플리케이션입니다.
- 문서 검토에 사용되는 모바일 기기용 기본 앱입니다.
- Mac 또는 Windows 데스크톱의 폴더를 WorkDocs 파일과 동기화하는 데 사용되는 WorkDocs Drive 입니다.

요금

WorkDocs를 사용하면 선결제 요금이나 약정이 없습니다. 활성 사용자 계정 및 사용하는 스토리지에 대해서만 요금을 지불하면 됩니다. 자세한 내용은 요금을 참조하세요.

리소스

다음의 관련 리소스는 이 서비스 사용 시 도움이 될 수 있습니다.

- 클래스 및 워크숍 - AWS 기술을 연마하고 실용적인 경험을 얻는 데 도움이 되는 자기 주도형 실습 외에도 역할 기반 및 특수 과정으로 연결되는 링크입니다.

- AWS 개발자 센터 - 자습서를 살펴보고, 도구를 다운로드하고, AWS 개발자 이벤트에 대해 알아봅니다.
- AWS 개발자 도구 - AWS 애플리케이션 개발 및 관리를 위한 개발자 도구, SDKs, IDE 툴킷 및 명령줄 도구에 대한 링크입니다.
- 리소스 센터 시작하기 -를 설정하고 AWS 계정, AWS 커뮤니티에 가입하고, 첫 번째 애플리케이션을 시작하는 방법을 알아봅니다.
- 실습 튜토리얼 – 단계별 튜토리얼에 따라 AWS에서 첫 번째 애플리케이션을 시작합니다.
- AWS 백서 - 아키텍처, 보안 및 경제와 같은 주제를 다루고 Solutions Architects 또는 기타 기술 전문가가 작성한 AWS 포괄적인 기술 AWS 백서 목록 링크입니다.
- AWS Support 센터 - AWS Support 사례를 생성하고 관리하기 위한 허브입니다. 포럼, 기술 FAQs, 서비스 상태 및 같은 기타 유용한 리소스에 대한 링크도 포함되어 있습니다 AWS Trusted Advisor.
- 지원 - 클라우드에서 애플리케이션을 구축하고 실행하는 데 도움이 지원되는 one-on-one 빠른 응답 지원 채널에 대한 정보를 제공하는 기본 웹 페이지입니다.
- Contact Us(문의처) - AWS 결제, 계정, 이벤트, 침해 및 기타 문제에 대해 문의할 수 있는 중앙 연락 창구입니다.
- AWS 사이트 약관 - 저작권 및 상표, 계정, 라이선스, 사이트 액세스 및 기타 주제에 대한 자세한 정보입니다.

시작

다음 코드 조각은 WorkDocs SDK 사용을 시작하는 데 도움이 될 수 있습니다.

> **Note**
> 보안을 강화하려면 가급적 IAM 사용자 대신 페더레이션 사용자를 생성하세요.

예시

- IAM 사용자 자격 증명을 사용하여 WorkDocs에 연결하고 사용자를 쿼리합니다.
- 역할을 수임하여 WorkDocs에 연결
- 문서 업로드
- 문서 다운로드
- 알림 설정
- 사용자 생성
- 사용자에게 리소스에 대한 권한 부여

IAM 사용자 자격 증명을 사용하여 WorkDocs에 연결하고 사용자를 쿼리합니다.

다음 코드는 IAM 사용자의 API 보안 인증 정보를 사용하여 API 직접 호출을 수행하는 방법을 보여줍니다. 이 경우 API 사용자와 WorkDocs 사이트는 동일한 AWS 계정에 속합니다.

> **Note**
> 보안을 강화하려면 가급적 IAM 사용자 대신 페더레이션 사용자를 생성하세요.

적절한 IAM 정책을 통해 IAM 사용자에게 WorkDocs API 액세스 권한이 부여되었는지 확인합니다.

코드 샘플은 DescribeUsers API를 사용하여 사용자를 검색하고 사용자에 대한 메타데이터를 얻습니다. 사용자 메타데이터에는 이름, 성, 사용자 ID, 루트 폴더 ID 같은 세부 정보가 들어 있습니다. 루트 폴더 ID는 사용자 대신 콘텐츠를 업로드하거나 다운로드하는 작업을 수행하려고 할 때 특히 유용합니다.

코드를 사용하려면 WorkDocs 조직 ID를 얻어야 합니다.

AWS 콘솔에서 WorkDocs 조직 ID를 얻으려면 다음 단계를 따르세요.

조직 ID를 가져오려면

1. AWS Directory Service 콘솔 탐색 창에서 디렉터리를 선택합니다.
2. WorkDocs 사이트에 해당하는 디렉터리 ID 값을 기록해 둡니다. 이것이 사이트의 조직 ID입니다.

다음 예시는 IAM 보안 인증 정보를 사용하여 API 직접 호출을 수행하는 방법을 보여줍니다.

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

역할을 수임하여 WorkDocs에 연결

이 예제에서는 AWS Java SDK를 사용하여 역할을 수임하고 역할의 임시 보안 자격 증명을 사용하여 WorkDocs에 액세스합니다. 코드 샘플은 DescribeFolderContents API를 사용하여 사용자 폴더의 항목을 나열합니다.

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

문서 업로드

> **Note**
>
> 이 섹션의 단계를 완료하려면 소프트웨어 개발자여야 합니다. WorkDocs를 사용하여 파일을 업로드하는 방법에 대한 자세한 내용은 WorkDocs 사용 설명서의 파일 업로드를 참조하세요.

다음 절차에 따라 WorkDocs에 문서를 업로드합니다.

문서를 업로드하려면

1. 다음과 같이 `AmazonWorkDocsClient`의 인스턴스를 만듭니다.

   IAM 사용자 보안 인증 정보를 사용하는 경우 IAM 사용자 자격 증명을 사용하여 WorkDocs에 연결하고 사용자를 쿼리합니다.을 참조하세요. IAM 역할을 맡는 경우 자세한 내용은 역할을 수임하여 WorkDocs에 연결을 참조하세요.

   > **Note**
   > 보안을 강화하려면 가급적 IAM 사용자 대신 페더레이션 사용자를 생성하세요.

```
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. 업로드에 대하여 다음과 같이 서명한 URL을 얻으십시오.

```
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. 다음과 같이 서명한 URL을 사용하여 문서를 업로드하십시오.

```
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. 문서 상태를 다음과 같이 ACTIVE로 바꾸어 업로드 프로세스를 완료합니다.

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

문서 다운로드

> **Note**
>
> 이 섹션의 단계를 완료하려면 소프트웨어 개발자여야 합니다. WorkDocs를 사용하여 파일을 다운로드하는 방법에 대한 자세한 내용은 WorkDocs 사용 설명서의 파일 다운로드를 참조하세요.

WorkDocs에서 문서를 다운로드하려면 다음과 같이 다운로드 URL을 가져온 다음 개발 플랫폼에서 제공하는 API 작업을 사용하여 URL을 사용하여 파일을 다운로드합니다.

```
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());
```

알림 설정

다음 프로세스에 따라 알림을 설정합니다.

1. 호출자가 알림 구독 관리 APIs에 액세스할 수 있도록 IAM 사용자 또는 역할 권한을 설정합니다.
2. 알림 구독 APIs를 호출하여 엔드포인트에 SNS 메시지 게시를 활성화하거나 비활성화합니다.

> **Note**
> 보안을 강화하려면 가급적 IAM 사용자 대신 페더레이션 사용자를 생성하세요.

IAM 사용자 권한을 설정하려면

- IAM 콘솔을 사용하여 사용자에게 다음 권한을 설정합니다.

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

알림을 활성화하려면

알림을 활성화하면 알림을 구독한 후 CreateNotificationSubscription을 직접적으로 호출할 수 있습니다.

1. https://console.aws.amazon.com/zocalo/ WorkDocs 콘솔을 엽니다.
2. Manage Your WorkDocs Sites(WorkDocs 사이트 관리) 페이지에서 원하는 디렉터리를 선택하고 Actions(작업)을 선택한 후 Manage Notifications(알림 관리)를 선택합니다.
3. Manage Notifications(알림 관리) 페이지에서 Enable Notifications(알림 활성화)를 선택합니다.
4. 가 WorkDocs 사이트에서 알림을 수신하도록 허용할 사용자 또는 역할의 ARN을 입력합니다.

WorkDocs에서 알림을 사용하도록 설정하는 방법에 대한 자세한 내용은 AWS SDK for Python 및 AWS Lambda와 함께 Amazon WorkDocs API 사용을 참조하세요. 알림을 활성화한 후 당신과 당신의 사용자는 알림을 구독할 수 있습니다.

WorkDocs 알림을 구독하려면

1. Amazon SNS 메시지를 처리하도록 엔드포인트를 준비합니다. 자세한 내용은 Amazon Simple Notification Service 개발자 안내서에서 HTTP/S 엔드포인트로 팬아웃을 참조하세요.

   > **Important**
   >
   > SNS는 구성된 엔드포인트에 확인 메시지를 보냅니다. 알림을 받으려면 이 메시지를 반드시 확인해야 합니다. 또한 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하세요.

2. 다음을 수행합니다.

   - 조직 ID 가져오기
     1. AWS Directory Service 콘솔 탐색 창에서 디렉터리를 선택합니다.
     2. Amazon WorkDocs 사이트에 해당하는 디렉터리 ID는 해당 사이트의 조직 ID로도 사용됩니다.
   - 다음과 같이 구독 요청을 생성합니다.

```
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 알림

메시지에 포함되는 정보는 다음과 같습니다.

- `organizationId` — 조직의 ID.
- `parentEntityType` — 상위의 유형(`Document` | `DocumentVersion` | `Folder`).
- `parentEntityId` — 상위의 ID.
- `entityType` — 엔터티의 유형(`Document` | `DocumentVersion` | `Folder`).
- `entityId` — 엔터티의 ID.
- 작업 — 이 작업은 다음 값 중 하나일 수 있습니다.
  - `delete_document`
  - `move_document`
  - `recycle_document`
  - `rename_document`
  - `revoke_share_document`
  - `share_document`
  - `upload_document_version`

알림을 비활성화하려면

1. https://console.aws.amazon.com/zocalo/ WorkDocs 콘솔을 엽니다.
2. Manage Your WorkDocs Sites(WorkDocs 사이트 관리) 페이지에서 원하는 디렉터리를 선택하고 Actions(작업)을 선택한 후 Manage Notifications(알림 관리)를 선택합니다.
3. 알림 관리 페이지에서 알림을 비활성화할 ARN을 선택하고 알림 비활성화를 선택합니다.

사용자 생성

다음 예제에서는 WorkDocs에서 사용자를 생성하는 방법을 보여줍니다.

> **Note**
>
> Connected AD 구성에는 유효한 작업이 아닙니다. Connected AD 구성에서 사용자를 생성하려면 사용자가 이미 엔터프라이즈 디렉터리에 있어야 합니다. 그런 다음 ActivateUser API를 호출하여 WorkDocs에서 사용자를 활성화해야 합니다.

다음 예시에서는 스토리지 할당량이 1GB인 사용자를 만드는 방법을 보여줍니다.

```
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 콘솔에서 WorkDocs 조직 ID를 얻으려면 다음 단계를 따르세요.

조직 ID를 가져오려면

1. AWS Directory Service 콘솔 탐색 창에서 디렉터리를 선택합니다.
2. WorkDocs 사이트에 해당하는 디렉터리 ID 값을 기록해 둡니다. 이것이 사이트의 조직 ID입니다.

사용자에게 리소스에 대한 권한 부여

다음 예시는 AddResourcePermissions API를 사용하여 리소스의 USER에 CONTRIBUTOR 권한을 부여하는 방법을 보여줍니다. API를 사용하여 사용자 또는 그룹에 폴더나 문서에 대한 권한을 부여할 수도 있습니다.

```
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);
```

관리 애플리케이션에 대한 인증 및 액세스 제어

WorkDocs 관리 APIs는 IAM 정책을 통해 인증되고 승인됩니다. IAM 관리자는 IAM 정책을 생성하여 이를 개발자가 API에 액세스하는 데 사용할 수 있도록 IAM 역할 또는 사용자에게 연결할 수 있습니다.

다음은 예로 제공된 것입니다.

업무

- 개발자에게 WorkDocs API에 대한 권한 부여
- 타사 개발자에게 WorkDocs APIs에 대한 권한 부여
- 사용자에게 IAM 역할을 맡을 수 있는 권한 부여
- 특정 WorkDocs 인스턴스에 대한 액세스 제한

개발자에게 WorkDocs API에 대한 권한 부여

> **Note**
> 보안을 강화하려면 가급적 IAM 사용자 대신 페더레이션 사용자를 생성하세요.

IAM 관리자인 경우 동일한 AWS 계정의 IAM 사용자에게 WorkDocs API 액세스 권한을 부여할 수 있습니다. 이렇게 하려면 WorkDocs API 권한 정책을 생성하여 IAM 사용자에게 연결합니다. 다음 API 정책은 다양한 `Describe` API에 읽기 전용 권한을 부여합니다.

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

```
        }
    ]
}
```

타사 개발자에게 WorkDocs APIs에 대한 권한 부여

타사 개발자 또는 다른 AWS 계정을 사용하는 사용자에게 액세스 권한을 부여할 수 있습니다. 이렇게 하려면 IAM 역할을 생성하고 WorkDocs API 허용 정책을 연결합니다.

이런 형태의 액세스가 필요한 경우는 다음과 같습니다.

- 개발자는 동일한 조직에 속하지만 개발자의 AWS 계정이 WorkDocs AWS 계정과 다릅니다.
- 기업이 타사 애플리케이션 개발자에게 WorkDocs API 액세스 권한을 부여하려는 경우.

이 두 시나리오에는 개발자 AWS 계정과 WorkDocs 사이트를 호스팅하는 다른 계정이라는 AWS 두 개의 계정이 관여합니다.

계정 관리자가 IAM 역할을 생성할 수 있도록 개발자는 다음 정보를 제공해야 합니다.

- AWS 계정 ID
- 고객이 개발자 확인에 사용하는 고유 `External ID`. 자세한 내용은 AWS 리소스에 대한 액세스 권한을 타사에 부여할 때 외부 ID를 사용하는 방법을 참조하세요.
- 애플리케이션이 액세스해야 하는 WorkDocs APIs. IAM 기반의 정책 제어는 세심한 제어가 가능하며 개별 API 수준에서 정책의 허용 또는 거부를 정의할 수 있습니다. WorkDocs APIs 목록은 WorkDocs API 참조를 참조하세요.

다음은 교차 계정 액세스를 위한 IAM 구성에 수반되는 단계를 설명하는 절차입니다.

크로스 계정 액세스를 위해 IAM을 구성하려면

1. WorkDocs API 권한 정책을 생성하고 이를 `WorkDocsAPIReadOnly` 정책이라고 합니다.
2. WorkDocs 사이트를 호스팅하는 AWS 계정의 IAM 콘솔에서 새 역할을 생성합니다.
    a. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
    b. 콘솔의 탐색 창에서 역할을 클릭한 다음 새 역할 생성을 클릭합니다.

c. Role name(역할 이름)에 이 역할의 목적을 확인할 수 있는 역할 이름을 입력합니다(예: `workdocs_app_role`). 역할 이름은 AWS 계정 내에서 고유해야 합니다. 이름을 입력한 후 다음 단계를 클릭합니다.

d. 역할 유형 선택 페이지에서 Role for Cross-Account Access(교차 계정 액세스의 역할)를 선택한 다음 만들 역할의 유형을 선택합니다.

   - 사용자 AWS 계정과 리소스 계정 모두의 관리자이거나 두 계정이 동일한 회사에 속하는 경우 소유한 계정 간에 액세스 권한 제공을 선택합니다. 사용자, 역할 및 액세스할 리소스가 모두 같은 계정에 있을 때도 이 옵션을 선택합니다.
   - WorkDocs 사이트를 소유한 AWS 계정의 관리자이고 애플리케이션 개발자 AWS 계정의 사용자에게 권한을 부여하려는 경우 계정과 타사 계정 간에 액세스 권한 제공을 선택합니다. 이 옵션을 선택하면 외부 ID(타사가 제공해야 함)를 지정하여 타사가 리소스에 액세스하기 위해 해당 역할을 사용할 수 있는 상황을 추가로 제어해야 합니다. 자세한 내용은 AWS 리소스에 대한 액세스를 타사에 부여할 때 외부 ID를 사용하는 방법을 참조하십시오.

e. 다음 페이지에서 리소스에 대한 액세스 권한을 부여할 AWS 계정 ID를 지정하고 타사 액세스의 경우 외부 ID도 입력합니다.

f. 다음 단계를 클릭하여 정책을 연결합니다.

3. 정책 연결 페이지에서 이전에 생성된 WorkDocs API 권한 정책을 검색하고 정책 옆의 상자를 선택하고 다음 단계를 클릭합니다.
4. 세부 정보를 검토하고 향후 참조를 위해 역할 ARN을 복사한 후 Create Role(역할 만들기)를 클릭하여 역할 생성을 완료합니다.
5. 역할 ARN을 개발자와 공유합니다. 다음은 역할 ARN의 예입니다.

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

사용자에게 IAM 역할을 맡을 수 있는 권한 부여

관리 AWS 계정이 있는 개발자는 사용자가 IAM 역할을 수임하도록 허용할 수 있습니다. 이를 위해 새 정책을 생성하여 해당 사용자에게 연결합니다.

정책에는 다음 예시와 같이 `sts:AssumeRole` 작업에 대한 `Allow`의 영향, 그리고 `Resource` 요소에서 역할의 Amazon 리소스 이름(ARN)이 작성된 문이 포함되어야 합니다. 그룹 멤버십 또는 직접 첨부를 통해 그 정책을 가져오는 사용자는 지정된 역할로 전환할 수 있습니다.

```
{
```

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

특정 WorkDocs 인스턴스에 대한 액세스 제한

AWS 계정에 여러 WorkDocs 사이트가 있고 특정 사이트에 API 액세스 권한을 부여하려는 경우 `Condition` 요소를 정의할 수 있습니다. `Condition` 요소를 사용하여 정책의 효력이 발생하는 시점에 대한 조건을 지정할 수 있습니다.

다음 예는 조건 요소를 보여줍니다.

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

정책에 위 조건을 적용하면 사용자는 ID가 인 WorkDocs 인스턴스에만 액세스할 수 있습니다`d-123456789c5`. WorkDocs 인스턴스 ID를 조직 ID 또는 디렉터리 ID라고도 합니다. 자세한 내용은 특정 WorkDocs 인스턴스에 대한 액세스 제한 단원을 참조하십시오.

AWS 콘솔에서 WorkDocs 조직 ID를 얻으려면 다음 단계를 따르세요.

조직 ID를 가져오려면

1. AWS Directory Service 콘솔 탐색 창에서 디렉터리를 선택합니다.
2. WorkDocs 사이트에 해당하는 디렉터리 ID 값을 기록해 둡니다. 이것이 사이트의 조직 ID입니다.

사용자 애플리케이션에 대한 인증 및 액세스 제어

WorkDocs 사용자 수준 애플리케이션은 WorkDocs 콘솔을 통해 등록 및 관리됩니다. 개발자는 각 애플리케이션에 고유한 IDs를 제공하는 WorkDocs 콘솔의 `My Applications` 페이지에 애플리케이션을 등록해야 합니다. 등록 과정에서 개발자는 액세스 토큰과 애플리케이션 범위를 받을 리디렉션 URI를 지정해야 합니다.

현재 애플리케이션은 등록된 동일한 AWS 계정 내의 WorkDocs 사이트에만 액세스할 수 있습니다.

내용

- WorkDocs APIs를 호출할 수 있는 권한 부여
- API 직접 호출 시 폴더 ID 사용
- 애플리케이션 만들기
- 애플리케이션 범위
- 권한 부여
- WorkDocs APIs 호출

WorkDocs APIs를 호출할 수 있는 권한 부여

명령줄 인터페이스 사용자는 WorkDocs 및에 대한 전체 권한을 가지고 있어야 합니다 AWS Directory Service. 권한이 없는 경우 모든 API 직접 호출은 UnauthorizedResourceAccessException 메시지를 반환합니다. 다음 정책은 모든 권한을 부여합니다.

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

읽기 전용 권한을 부여하려면 이 정책을 사용하세요.

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

정책에서 첫 번째 작업은 모든 WorkDocs `Describe` 작업에 대한 액세스 권한을 부여합니다. `DescribeDirectories` 작업은 AWS Directory Service 디렉터리에 대한 정보를 가져옵니다. Amazon EC2 작업을 사용하면 WorkDocs가 VPCs 및 서브넷 목록을 가져올 수 있습니다.

API 직접 호출 시 폴더 ID 사용

API 직접 호출로 폴더에 액세스할 때마다 폴더 이름이 아닌 폴더 ID를 사용해야 합니다. 예를 들어 `client.get_folder(FolderId='MyDocs')`를 통과하면 API 직접 호출은 UnauthorizedResourceAccessException 메시지와 다음 404 메시지를 반환합니다.

```
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.
```

이를 방지하려면 폴더의 URL에 있는 ID를 사용하세요.

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

해당 ID를 전달하면 올바른 결과가 반환됩니다.

```
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'}}
```

애플리케이션 만들기

WorkDocs 관리자는 다음 단계를 사용하여 애플리케이션을 생성합니다.

애플리케이션 생성

1. https://console.aws.amazon.com/zocalo/ WorkDocs 콘솔을 엽니다.
2. My Applications(내 애플리케이션), 애플리케이션 생성을 선택합니다.
3. 다음 값을 입력합니다.

   애플리케이션 이름

   애플리케이션의 이름을 지정합니다.

   이메일

   애플리케이션과 연결할 이메일 주소입니다.

   애플리케이션 설명

   애플리케이션에 대한 설명입니다.

   리디렉션 URI

   WorkDocs가 트래픽을 리디렉션할 위치입니다.

   애플리케이션 범위

   애플리케이션의 읽기 또는 쓰기 범위입니다. 자세한 내용은 애플리케이션 범위를 참조하세요.
4. 생성(Create)을 선택합니다.

애플리케이션 범위

WorkDocs는 다음 애플리케이션 범위를 지원합니다.

- 콘텐츠 읽기(`workdocs.content.read`)는 애플리케이션에 다음 WorkDocs APIs에 대한 액세스 권한을 부여합니다.
  - Get*
  - Describe*
- 콘텐츠 쓰기(`workdocs.content.write`)는 애플리케이션에 다음 WorkDocs APIs에 대한 액세스 권한을 부여합니다.
  - Create*
  - Update*
  - Delete*

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

권한 부여

애플리케이션 등록이 완료되면 애플리케이션은 WorkDocs 사용자를 대신하여 권한 부여를 요청할 수 있습니다. 이렇게 하려면 애플리케이션이 WorkDocs OAuth 엔드포인트인를 방문하여 다음 쿼리 파라미터를 `https://auth.amazonworkdocs.com/oauth`제공해야 합니다.

- [필수] `app_id`—애플리케이션 등록 시 생성되는 애플리케이션 ID.
- [필수] `auth_type`—요청에 대한 OAuth 유형. 지원되는 값은 `ImplicitGrant`입니다.
- [필수] `redirect_uri`—액세스 토큰을 받기 위하여 애플리케이션에 등록한 리디렉션 URI.
- [선택 사항] `scopes`—쉼표로 구분되는 범위 목록. 지정하지 않은 경우 등록 과정에서 선택한 범위 목록이 사용됩니다.
- [선택 사항] `state`—액세스 토큰과 함께 반환되는 스트링.

> **Note**
>
> 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하세요.

액세스 토큰을 얻기 위해 OAuth 흐름을 시작하는 샘플 GET 요청:

```
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
```

OAuth 권한 부여 흐름 중에는 다음 작업이 진행됩니다.

1. 애플리케이션 사용자에게 WorkDocs 사이트 이름을 입력하라는 메시지가 표시됩니다.
2. 사용자는 자격 증명을 입력하기 위해 WorkDocs 인증 페이지로 리디렉션됩니다.

3. 인증에 성공하면 사용자가 애플리케이션에 WorkDocs에 액세스할 수 있는 권한을 부여하거나 거부할 수 있는 동의 화면이 표시됩니다.
4. 동의 화면에서 사용자가 `Accept`를 선택하면 쿼리 파라미터와 마찬가지로 액세스 토큰, 리전 정보와 더불어 애플리케이션의 콜백 URL로 브라우저가 리디렉션됩니다.

WorkDocs의 샘플 GET 요청:

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

액세스 토큰 외에도 WorkDocs OAuth 서비스는 선택한 WorkDocs 사이트에 대한 쿼리 파라미터`region`로 도 반환합니다. WorkDocs 외부 애플리케이션은 `region` 파라미터를 사용하여 WorkDocs 서비스 엔드포인트를 결정해야 합니다.

명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하세요.

WorkDocs APIs 호출

액세스 토큰을 얻은 후 애플리케이션은 WorkDocs 서비스에 대한 API 호출을 수행할 수 있습니다.

> **Important**

이 예시는 curl GET 요청을 사용하여 문서의 메타데이터를 가져오는 방법을 보여줍니다.

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

사용자의 루트 폴더를 설명하기 위한 JavaScript 함수 샘플:

```
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 기반 API 호출의 샘플은 아래 설명이 나와 있습니다.

```
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());
}
```

WorkDocs 콘텐츠 관리자

WorkDocs Content Manager는 콘텐츠를 업로드하거나 WorkDocs 사이트에서 다운로드하는 상위 수준 유틸리티 도구입니다.

주제

- WorkDocs 콘텐츠 관리자 구성
- 문서 다운로드
- 문서 업로드

WorkDocs 콘텐츠 관리자 구성

관리 및 사용자 애플리케이션에 WorkDocs Content Manager를 사용할 수 있습니다.

사용자 애플리케이션의 경우 개발자는 익명 AWS 자격 증명과 인증 토큰을 사용하여 WorkDocs Content Manager를 구성해야 합니다.

관리 애플리케이션의 경우 WorkDocs 클라이언트를 (IAM) 자격 증명으로 AWS Identity and Access Management 초기화해야 합니다. 또한 인증 토큰은 후속 API 호출에서 생략되어야 합니다.

다음 코드는 Java 또는 C#을 사용하여 사용자 애플리케이션에 대한 WorkDocs Content Manager를 초기화하는 방법을 보여줍니다.

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);
```

문서 다운로드

개발자는 WorkDocs Content Manager를 사용하여 WorkDocs에서 문서의 특정 버전 또는 최신 버전을 다운로드할 수 있습니다. 다음 예에서는 Java 및 C#을 사용하여 특정 버전의 문서를 다운로드하는 방법을 보여 줍니다.

> **Note**
>
> 최신 버전의 문서를 다운로드하려면 `GetDocumentStream` 요청을 구성할 때 `VersionId`를 지정하지 마십시오.

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();
```

문서 업로드

WorkDocs Content Manager는 WorkDocs 사이트에 콘텐츠를 업로드하기 위한 API를 제공합니다. 다음 예에서는 Java 및 C#을 사용하여 문서를 업로드하는 방법을 보여 줍니다.

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();
```