



User Guide

AWS Transfer Family



AWS Transfer Family: User Guide

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is AWS Transfer Family?	1
How AWS Transfer Family works	3
Blog posts relevant for Transfer Family	5
Prerequisites	8
Regions, endpoints and quotas	8
Sign up for AWS	8
Configure storage	9
Configure an Amazon S3 bucket	10
Configure an Amazon EFS file system	14
Create an IAM role and policy	17
Create a user role	19
How session policies work	22
Example read/write access policy	25
Transfer Family tutorials	29
Get started with server endpoints	29
Prerequisites	30
Sign in to the console	31
Create an SFTP-enabled server	31
Add a service managed user	32
Transfer a file using a client	33
Create a decryption Workflow	35
Step 1: Configure an execution role	36
Step 2: Create a managed workflow	37
Step 3: Add the workflow to a server and create a user	38
Step 4: Create a PGP key pair	40
Step 5: Store the PGP private key in AWS Secrets Manager	40
Step 6: Encrypt a file	42
Step 7: Run the workflow and view the results	42
Create and use SFTP connectors	43
Step 1: Create the necessary supporting resources	44
Step 2: Create and test an SFTP connector	49
Step 3: Send and retrieve files using the SFTP connector	53
Procedures to create a Transfer Family server to use as your remote SFTP server	56
Use a custom identity provider	59

Prerequisites	59
Step 1: Create a CloudFormation stack	60
Step 2: Check the API Gateway method configuration for your server	61
Step 3: View the Transfer Family server details	61
Step 4: Test that your user can connect to the server	63
Step 5: Test the SFTP connection and file transfer	63
Step 6: Limit access to the bucket	64
Update Lambda if using Amazon EFS	66
Set up an AS2 configuration	67
Step 1: Create certificates for AS2	69
Step 2: Create a Transfer Family server that uses the AS2 protocol	72
Step 3: Import certificates as Transfer Family certificate resources	76
Step 4: Create profiles for you and your trading partner	77
Step 5: Create an agreement between you and your partner	77
Step 6: Create a connector between you and your partner	78
Step 7: Test exchanging files over AS2 by using Transfer Family	79
Transfer Family for SFTP, FTPS, FTP	82
Identity provider options	82
AWS Transfer Family endpoint type matrix	84
Configure a Transfer Family server endpoint	88
Create an SFTP-enabled server	90
Create an FTPS-enabled server	98
Create an FTP-enabled server	106
Create a server in a VPC	114
Working with custom hostnames	135
Transfer files over server endpoint	139
Available SFTP/FTPS/FTP Commands	141
Find your Amazon VPC endpoint	143
Avoid setstat errors	144
Use OpenSSH	34
Use WinSCP	146
Use Cyberduck	33
Use FileZilla	149
Use a Perl client	151
Post upload processing	151
Manage users	152

Service-managed users	154
Directory service for MS AD	163
Directory service for Azure AD	174
Custom identity provider users	181
Use logical directories	209
Rules for using logical directories	210
Implementing logical directories and <code>chroot</code>	211
Configure logical directories example	214
Configure logical directories for Amazon EFS	215
Custom AWS Lambda response	215
SFTP connectors	217
Configure SFTP connectors	217
Create an SFTP connector	218
Store a secret for use with an SFTP connector	226
Generate and format the SFTP connector private key	227
Test an SFTP connector	230
Transfer files with SFTP connectors	232
List contents of remote directory	233
Manage SFTP connectors	235
Update SFTP connectors	236
View SFTP connector details	236
Quotas for SFTP connectors	237
Transfer Family for AS2	239
AS2 use cases	240
Configure AS2	245
Create an AS2 server using the Transfer Family console	246
Create an AS2 server using a template	249
AS2 configurations	252
AS2 quotas	253
AS2 features and capabilities	258
Configure AS2 connectors	259
Create an AS2 connector	260
AS2 connector algorithms	263
Basic authentication for AS2 connectors	264
Enable Basic authentication for AS2 connectors	266
View connector details	270

Manage AS2 partners	271
Import AS2 certificates	271
AS2 certificate rotation	273
Create AS2 profiles	275
Create AS2 agreements	275
Transfer AS2 messages	277
Send AS2 messages	278
Receive AS2 messages	279
Configure HTTPS for AS2	280
Transfer files with AS2 connectors	286
File names and locations	287
Status codes	289
Sample JSON files	290
Monitor AS2	292
AS2 Status codes	294
AS2 error codes	294
Managing file-processing workflows	307
Create a workflow	309
Configure and run a workflow	310
View workflow details	312
Use predefined steps	315
Copy file	315
Decrypt file	320
Tag file	326
Delete file	327
Named variables for workflows	328
Example tag and delete workflow	328
Use custom file-processing steps	333
Using multiple Lambda functions consecutively	334
Accessing a file after custom processing	335
Example events sent to AWS Lambda upon file upload	336
Example Lambda function for a custom workflow step	337
IAM permissions for a custom step	338
IAM policies for workflows	338
Workflow trust relationships	340
Example execution role: Decrypt, copy, and tag	341

Example execution role: Run function and delete	343
Exception handling for a workflow	343
Monitor workflow execution	344
CloudWatch logging for a workflow	344
CloudWatch metrics for workflows	347
Create workflow from template	347
Remove a workflow from a Transfer Family server	351
Restrictions and limits	352
Managing servers	355
View a list of servers	355
Delete a server	355
View SFTP server details	357
View AS2 server details	358
Edit server details	360
Edit the file transfer protocols	362
Edit custom identity provider parameters	364
Edit the server endpoint	366
Edit logging	368
Edit the security policy	368
Change the managed workflow	370
Change the display banners for your server	371
Put your server online or offline	371
Manage server host keys	372
Add an additional server host key	373
Delete a server host key	374
Rotate the server host keys	375
Additional server host key information	377
Monitor usage within console	377
Managing access controls	381
Creating an S3 bucket access policy	382
Creating a session policy	383
Preventing users from running <code>mkdir</code> in an S3 bucket	386
CloudTrail logging	388
Enabling CloudTrail logging	389
Example log entry for creating a server	390
CloudWatch logging	392

Types of CloudWatch logging for Transfer Family	392
Creating logging for servers	394
Creating logging for servers	396
Updating logging for a server	397
Viewing the server configuration	400
Managing logging for workflows	402
Configuring a role for CloudWatch	405
Viewing Transfer Family log streams	407
Creating Amazon CloudWatch alarms	411
Logging S3 API calls to S3 access logs	411
Examples to limit confused deputy problem	412
CloudWatch log structure for Transfer Family	414
JSON structured logs for Transfer Family	414
Legacy logs for Transfer Family	416
Example CloudWatch log entries	419
Example transfer sessions log entries	420
Example log entries for SFTP connectors	421
Example log entries for Key exchange algorithm failures	423
Using CloudWatch metrics	424
Transfer Family dimensions	426
User notifications	426
CloudWatch queries	426
Managing events using EventBridge	429
Transfer Family events	430
SFTP, FTPS, and FTP server events	430
SFTP connector events	431
A2S events	432
Sending Transfer Family events	432
Creating event patterns	433
Testing event patterns for Transfer Family events	434
Permissions	434
Additional resources	435
Events detail reference	435
Server events	436
Connector events	440
AS2 events	446

Security	453
Security policies for servers	455
Cryptographic algorithms	456
TransferSecurityPolicy-2024-01	465
TransferSecurityPolicy-2023-05	466
TransferSecurityPolicy-2022-03	467
TransferSecurityPolicy-2020-06 and TransferSecurityPolicy-Restricted-2020-06	468
TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11	469
TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05	471
TransferSecurityPolicy-FIPS-2023-05	472
TransferSecurityPolicy-FIPS-2020-06	474
Post Quantum security policies	475
Security policies for SFTP connectors	480
Post-Quantum security policies	482
About post-quantum hybrid key exchange in SSH	483
How to use it	483
How to test it	485
Data protection	488
Data encryption	489
Key management in Transfer Family	491
Identity and access management	506
Audience	507
Authenticating with identities	508
Managing access using policies	511
How AWS Transfer Family works with IAM	513
Identity-based policy examples	518
Tag-based policy examples	521
Troubleshooting identity and access	524
Compliance validation	526
Resilience	527
Infrastructure security	528
Web application firewall	528
Cross-service confused deputy prevention	530
Transfer Family user roles	531
Transfer Family workflow roles	533
Transfer Family logging/invocation roles	534

AWS managed policies	536
AWSTransferConsoleFullAccess	536
AWSTransferFullAccess	538
AWSTransferLoggingAccess	539
AWSTransferReadOnlyAccess	540
Policy updates	541
Troubleshooting Transfer Family	542
Troubleshoot service-managed users	542
Troubleshoot Amazon EFS service-managed users	543
Troubleshoot public key body too long	543
Troubleshoot failed to add SSH public key	543
Troubleshoot Amazon API Gateway issues	544
Too many authentication failures	544
Connection closed	545
Troubleshoot policies for encrypted Amazon S3 buckets	546
Troubleshoot authentication issues	546
Authentication failures—SSH/SFTP	547
Managed AD mismatched realms issue	547
Miscellaneous authentication issues	548
Troubleshoot managed workflows issues	548
Troubleshoot workflow-related errors using Amazon CloudWatch	548
Troubleshoot workflow copy errors	550
Troubleshoot workflow decryption issues	551
Troubleshoot error for signed encryption file	551
Troubleshoot error for a FIPS algorithm	552
Troubleshoot Amazon EFS issues	554
Troubleshoot missing POSIX profile	554
Troubleshoot logical directories with Amazon EFS	555
Troubleshoot testing your identity provider	556
Troubleshoot adding trusted host keys for your SFTP connector	556
Troubleshoot file upload issues	557
Troubleshoot Amazon S3 file upload errors	557
Troubleshoot unreadable file names	557
Troubleshoot ResourceNotFoundException exception	558
Troubleshoot SFTP connector issues	558
Key negotiation fails	559

Miscellaneous SFTP connector issues	559
Troubleshoot AS2 issues	560
API reference	561
Welcome	561
Actions	564
CreateAccess	567
CreateAgreement	574
CreateConnector	580
CreateProfile	588
CreateServer	592
CreateUser	605
CreateWorkflow	613
DeleteAccess	622
DeleteAgreement	625
DeleteCertificate	628
DeleteConnector	630
DeleteHostKey	632
DeleteProfile	635
DeleteServer	637
DeleteSshPublicKey	640
DeleteUser	643
DeleteWorkflow	646
DescribeAccess	648
DescribeAgreement	652
DescribeCertificate	655
DescribeConnector	658
DescribeExecution	661
DescribeHostKey	666
DescribeProfile	669
DescribeSecurityPolicy	672
DescribeServer	676
DescribeUser	681
DescribeWorkflow	686
ImportCertificate	691
ImportHostKey	696
ImportSshPublicKey	700

ListAccesses	705
ListAgreements	709
ListCertificates	713
ListConnectors	717
ListExecutions	720
ListHostKeys	725
ListProfiles	729
ListSecurityPolicies	733
ListServers	737
ListTagsForResource	741
ListUsers	746
ListWorkflows	751
SendWorkflowStepState	754
StartDirectoryListing	757
StartFileTransfer	763
StartServer	769
StopServer	772
TagResource	775
TestConnection	778
TestIdentityProvider	782
UntagResource	789
UpdateAccess	792
UpdateAgreement	799
UpdateCertificate	805
UpdateConnector	809
UpdateHostKey	814
UpdateProfile	818
UpdateServer	821
UpdateUser	833
Data Types	840
As2ConnectorConfig	843
CopyStepDetails	847
CustomStepDetails	849
DecryptStepDetails	851
DeleteStepDetails	854
DescribedAccess	856

DescribedAgreement	860
DescribedCertificate	864
DescribedConnector	868
DescribedExecution	872
DescribedHostKey	875
DescribedProfile	878
DescribedSecurityPolicy	881
DescribedServer	884
DescribedUser	893
DescribedWorkflow	897
EfsFileLocation	899
EndpointDetails	901
ExecutionError	905
ExecutionResults	907
ExecutionStepResult	908
FileLocation	910
HomeDirectoryMapEntry	911
IdentityProviderDetails	913
InputFileLocation	915
ListedAccess	916
ListedAgreement	919
ListedCertificate	922
ListedConnector	925
ListedExecution	927
ListedHostKey	929
ListedProfile	931
ListedServer	933
ListedUser	936
ListedWorkflow	939
LoggingConfiguration	941
PosixProfile	943
ProtocolDetails	945
S3FileLocation	949
S3InputFileLocation	951
S3StorageOptions	953
S3Tag	954

ServiceMetadata	955
SftpConnectorConfig	956
SshPublicKey	958
Tag	960
TagStepDetails	961
UserDetails	963
WorkflowDetail	965
WorkflowDetails	967
WorkflowStep	969
Making API requests	971
Transfer Family required request headers	971
Transfer Family request inputs and signing	973
Error responses	974
Available libraries	976
Common Parameters	976
Common Errors	979
Document history	981
AWS Glossary	993

What is AWS Transfer Family?

AWS Transfer Family is a secure transfer service that enables you to transfer files into and out of AWS storage services. Transfer Family is part of the AWS Cloud platform. AWS Transfer Family offers fully managed support for the transfer of files over SFTP, AS2, FTPS, and FTP directly into and out of Amazon S3 or Amazon EFS. You can seamlessly migrate, automate, and monitor your file transfer workflows by maintaining existing client-side configurations for authentication, access, and firewalls—so nothing changes for your customers, partners, and internal teams, or their applications.

See [Getting started with AWS](#) to learn more and to start building cloud applications with Amazon Web Services.

AWS Transfer Family supports transferring data from or to the following AWS storage services.

- Amazon Simple Storage Service (Amazon S3) storage. For information about Amazon S3, see [Getting started with Amazon Simple Storage Service](#).
- Amazon Elastic File System (Amazon EFS) Network File System (NFS) file systems. For information about Amazon EFS, see [What is Amazon Elastic File System?](#)

AWS Transfer Family supports transferring data over the following protocols:

- Secure File Transfer Protocol (SFTP): version 3

The official IETF document is here: [SSH File Transfer Protocol draft-ietf-secsh-filexfer-02.txt](#).

- File Transfer Protocol Secure (FTPS)
- File Transfer Protocol (FTP)
- Applicability Statement 2 (AS2)

Note

For FTP and FTPS data connections, the port range that Transfer Family uses to establish the data channel is 8192–8200.

File transfer protocols are used in data exchange workflows across different industries such as financial services, healthcare, advertising, and retail, among others. Transfer Family simplifies the migration of file transfer workflows to AWS.

The following are some common use cases for using Transfer Family with Amazon S3:

- Data lakes in AWS for uploads from third parties such as vendors and partners.
- Subscription-based data distribution with your customers.
- Internal transfers within your organization.

The following are some common use cases for using Transfer Family with Amazon EFS:

- Data distribution
- Supply chain
- Content management
- Web serving applications

The following are some common use cases for using Transfer Family with AS2:

- Workflows with compliance requirements that rely on having data protection and security features built into the protocol
- Supply chain logistics
- Payments workflows
- Business-to-business (B2B) transactions
- Integrations with enterprise resource planning (ERP) and customer relationship management (CRM) systems

With Transfer Family, you get access to a file transfer protocol-enabled server in AWS without the need to run any server infrastructure. You can use this service to migrate your file transfer-based workflows to AWS while maintaining your end users' clients and configurations as is. You first associate your hostname with the server endpoint, then add your users and provision them with the right level of access. After you do this, your users' transfer requests are serviced directly out of your Transfer Family server endpoint.

Transfer Family provides the following benefits:

- A fully managed service that scales in real time to meet your needs.
- You don't need to modify your applications or run any file transfer protocol infrastructure.
- With your data in durable Amazon S3 storage, you can use native AWS services for processing, analytics, reporting, auditing, and archival functions.
- With Amazon EFS as your data store, you get a fully managed elastic file system for use with AWS Cloud services and on-premises resources. Amazon EFS is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files. This helps eliminate the need to provision and manage capacity to accommodate growth.
- A fully managed, serverless File Transfer Workflow service that makes it easy to set up, run, automate, and monitor processing of files uploaded using AWS Transfer Family.
- There are no upfront costs, and you pay only for the use of the service.

In the following sections, you can find a description of the different features of Transfer Family, a getting started tutorial, detailed instructions on how to set up the different protocol enabled servers, how to use different types of identity providers, and the service's API reference.

To get started with Transfer Family, see the following:

- [How AWS Transfer Family works](#)
- [Prerequisites](#)
- [Getting started with AWS Transfer Family server endpoints](#)

How AWS Transfer Family works

AWS Transfer Family is a fully managed AWS service that you can use to transfer files into and out of Amazon Simple Storage Service (Amazon S3) storage or Amazon Elastic File System (Amazon EFS) file systems over the following protocols:

- Secure File Transfer Protocol (SFTP): version 3

The official IETF document is here: [SSH File Transfer Protocol draft-ietf-secsh-filexfer-02.txt](#).

- File Transfer Protocol Secure (FTPS)
- File Transfer Protocol (FTP)
- Applicability Statement 2 (AS2)

AWS Transfer Family supports up to 3 Availability Zones and is backed by an auto scaling, redundant fleet for your connection and transfer requests. For an example on how to build for higher redundancy and minimize network latency by using Latency-based routing, see the blog post [Minimize network latency with your AWS transfer for SFTP servers](#).

Transfer Family Managed File Transfer Workflows (MFTW) is a fully managed, serverless File Transfer Workflow service that makes it easy to set up, run, automate, and monitor processing of files uploaded using AWS Transfer Family. Customers can use MFTW to automate various processing steps such as copying, tagging, scanning, filtering, compressing/decompressing, and encrypting/decrypting the data that's transferred using Transfer Family. This provides end to end visibility for tracking and auditability. For more details, see [AWS Transfer Family managed workflows](#).

AWS Transfer Family supports any standard file transfer protocol client. Some commonly used clients are the following:

- [OpenSSH](#) – A Macintosh and Linux command line utility.
- [WinSCP](#) – A Windows-only graphical client.
- [Cyberduck](#) – A Linux, Macintosh, and Microsoft Windows graphical client.
- [FileZilla](#) – A Linux, Macintosh, and Windows graphical client.

AWS offers the following Transfer Family workshops.

- Build a file transfer solution that leverages AWS Transfer Family for managed SFTP/FTPS endpoints and Amazon Cognito and DynamoDB for user management. You can view the details for this workshop [here](#).
- Build a Transfer Family endpoint with AS2 enabled, and a Transfer Family AS2 connector You can view the details for this workshop [here](#).
- Build a solution that provides prescriptive guidance and a hands on lab on how you can build a scalable and secure file transfer architecture on AWS without needing to modify existing applications or manage server infrastructure. You can view the details for this workshop [here](#).

Blog posts relevant for Transfer Family

The following table lists the blog posts that contain useful information for Transfer Family customers. The table is in reverse chronological order, so that the most recent posts are at the beginning of the table.

Blog post title and link	Date
Architecting secure and compliant managed file transfers with AWS Transfer Family SFTP connectors and PGP encryption	May 16, 2024
Using Amazon Cognito as an identity provider with AWS Transfer Family and Amazon S3	May 14, 2024
How Transfer Family can help you build a secure, compliant managed file transfer solution	January 3, 2024
Detect malware threats using AWS Transfer Family	July 20, 2023
Extending SAP workloads with AWS Transfer Family	July 13, 2023
Encrypt and decrypt files with PGP and AWS Transfer Family	June 21, 2023
Authenticating to AWS Transfer Family with Azure Active Directory and AWS Lambda	December 15, 2022
Customize file delivery notifications using AWS Transfer Family managed workflows	October 14, 2022
Building a cloud-native file transfer platform using AWS Transfer Family workflows	January 5, 2022
Enabling user self-service key management with AWS Transfer Family and AWS Lambda.	December 17, 2021

Blog post title and link	Date
<u>Enhance data access control with AWS Transfer Family and Amazon S3</u>	October 5, 2021
<u>Improve throughput for internet facing file transfers using AWS Global Accelerator and AWS Transfer Family services</u>	June 7, 2021
<u>Securing AWS Transfer Family with AWS Web Application Firewall and Amazon API Gateway</u>	May 5, 2021
<u>Securing AWS Transfer Family with AWS Web Application Firewall and Amazon API Gateway</u>	January 15, 2021
<u>AWS Transfer Family support for Amazon Elastic File System</u>	January 7, 2021
<u>Enable password authentication for AWS Transfer Family using AWS Secrets Manager</u>	November 5, 2020
<u>Centralize data access using AWS Transfer Family and AWS Storage Gateway</u>	June 22, 2020
<u>Using Amazon EFS for AWS Lambda in your serverless applications</u>	June 18, 2020
<u>Use IP allow list to secure your AWS Transfer Family servers</u>	April 8, 2020
<u>Minimize network latency with your AWS transfer for SFTP servers</u>	February 19, 2020
<u>Lift and Shift migration of SFTP servers to AWS</u>	February 12, 2020
<u>Simplify your AWS SFTP Structure with chroot and logical directories</u>	September 26, 2019

Blog post title and link	Date
Using Okta as an identity provider with AWS Transfer Family	May 30, 2019

Prerequisites

The following sections describe the prerequisites required to use the AWS Transfer Family service. At a minimum, you need to create an Amazon Simple Storage Service (Amazon S3) bucket and provide access to that bucket through an AWS Identity and Access Management (IAM) role. Your role also needs to establish a trust relationship. This trust relationship allows Transfer Family to assume the IAM role to access your bucket so that it can service your users' file transfer requests.

Topics

- [Supported AWS Regions, endpoints and quotas](#)
- [Sign up for AWS](#)
- [Configure storage to use with AWS Transfer Family](#)
- [Create an IAM role and policy](#)

Supported AWS Regions, endpoints and quotas

To connect programmatically to an AWS service, you use an endpoint. For example, the endpoint for customers in US East (Ohio) region (us-east-2), is `transfer.us-east-2.amazonaws.com`. Service quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account. In this guide, you can find quotas in [AS2 quotas](#) and [Quotas for SFTP connectors](#).

For more information about supported AWS Regions, endpoints, and service quotas, see [AWS Transfer Family endpoints and quotas](#) in the *Amazon Web Services General Reference*.

Sign up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including AWS Transfer Family. You are charged only for the services that you use.

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform [tasks that require root user access](#).

For information about pricing and to use AWS Pricing Calculator to get an estimate of the cost to use Transfer Family, see [AWS Transfer Family pricing](#).

For information about AWS Region availability, see the [AWS Transfer Family endpoints and quotas](#) in the *AWS General Reference*.

Configure storage to use with AWS Transfer Family

This topic describes the storage options that you can use with AWS Transfer Family. You can use either Amazon S3 or Amazon EFS as storage for your Transfer Family servers.

Contents

- [Configure an Amazon S3 bucket](#)
 - [Amazon S3 access points](#)
 - [Amazon S3 HeadObject behavior](#)
 - [Grant ability to only write and list files](#)
 - [Large number of zero-byte objects causing latency issues](#)
- [Configure an Amazon EFS file system](#)
 - [Amazon EFS file ownership](#)
 - [Set up Amazon EFS users for Transfer Family](#)
 - [Configure Transfer Family users on Amazon EFS](#)
 - [Create an Amazon EFS root user](#)
 - [Supported Amazon EFS commands](#)

Configure an Amazon S3 bucket

AWS Transfer Family accesses your Amazon S3 bucket to service your users' transfer requests, so you need to provide an Amazon S3 bucket as part of setting up your file transfer protocol-enabled server. You can use an existing bucket, or you can create a new one.

Note

You don't have to use a server and Amazon S3 bucket that are in the same AWS Region, but we recommend this as a best practice.

When you set up your users, you assign them each an IAM role. This role determines the level of access that they have to your Amazon S3 bucket.

For information on creating a new bucket, see [How do I create an S3 bucket?](#) in the *Amazon Simple Storage Service User Guide*.

Note

You can use Amazon S3 Object Lock to prevent objects from being overwritten for a fixed amount of time or indefinitely. This works the same way with Transfer Family as with other services. If an object exists and is protected, writing to that file or deleting it is not allowed. For more details on Amazon S3 Object Lock, see [Using Amazon S3 Object Lock](#) in the *Amazon Simple Storage Service User Guide*.

Amazon S3 access points

AWS Transfer Family supports [Amazon S3 Access Points](#), a feature of Amazon S3 that allows you to easily manage granular access to shared data sets. You can use S3 Access Point aliases anywhere you use an S3 bucket name. You can create hundreds of access points in Amazon S3 for users who have different permissions to access shared data in an Amazon S3 bucket.

For example, you can use access points to allow three different teams to have access to the same shared dataset where one team can read data from S3, a second team can write data to S3, and the third team can read, write, and delete data from S3. To implement a granular access control as mentioned above, you can create an S3 access point that contains a policy that gives asymmetrical access to different teams. You can use S3 access points with your Transfer Family server to achieve

a fine-grained access control, without creating a complex S3 bucket policy that spans hundreds of use cases. To learn more about how to use S3 access points with a Transfer Family server, refer to the [Enhance data access control with AWS Transfer Family and Amazon S3](#) blog post.

Note

AWS Transfer Family does not currently support Amazon S3 Multi-Region Access Points.

Amazon S3 HeadObject behavior

Note

When you create or update a Transfer Family server, you can optimize performance for your Amazon S3 directories, which eliminates HeadObject calls.

In Amazon S3, buckets and objects are the primary resources, and objects are stored in buckets. Amazon S3 can mimic a hierarchical file system, but can sometimes behave differently than a typical file system. For example, directories are not a first-class concept in Amazon S3 but instead are based on object keys. AWS Transfer Family infers a directory path by splitting an object's key by the forward slash character (`/`), treating the last element as the file name, then grouping file names which have the same prefix together under the same path. Zero-byte objects are created to represent a folder's path when you create an empty directory using `mkdir` or by using the Amazon S3 console. The key for these objects ends in a trailing forward slash. These zero-byte objects are described in [Organizing objects in the Amazon S3 console using folders](#) in the *Amazon S3 User Guide*.

When you run an `ls` command, and some results are Amazon S3 zero-byte objects (these objects have keys that end in the forward slash character), Transfer Family issues a HeadObject request for each of these objects (see [HeadObject](#) in the *Amazon Simple Storage Service API Reference* for details). This can result in the following problems when using Amazon S3 as your storage with Transfer Family.

Grant ability to only write and list files

In some cases, you might want to offer only write access to your Amazon S3 objects. For example, you might want to provide access to write (or upload) and list objects in a bucket, but not to read

(download) objects. To perform `ls` and `mkdir` commands by using file transfer clients, you must have the Amazon S3 `ListObjects` and `PutObject` permissions. However, when Transfer Family needs to make a `HeadObject` call to either write or list files, the call fails with an error of **Access denied**, because this call requires the `GetObject` permission.

 **Note**

When you create or update a Transfer Family server, you can optimize performance for your Amazon S3 directories, which eliminates `HeadObject` calls.

In this case, you can grant access by adding an AWS Identity and Access Management (IAM) policy condition that adds the `GetObject` permission only for objects that end in a slash (`/`). This condition prevents `GetObject` calls on files (so that they can't be read), but allows the user to list and traverse folders. The following example policy offers only write and list access to your Amazon S3 buckets. To use this policy, replace *DOC-EXAMPLE-BUCKET* with the name of your bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListing",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
    },
    {
      "Sid": "AllowReadWrite",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ]
    },
    {
      "Sid": "DenyIfNotFolder",
      "Effect": "Deny",
```

```
    "Action": [
      "s3:GetObject",
      "s3:GetObjectVersion"
    ],
    "NotResource": [
      "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*/"
    ]
  }
]
```

Note

This policy doesn't allow users to append files. In other words, a user who is assigned this policy can't open files to add content to them, or to modify them. Additionally, if your use case requires a `HeadObject` call before uploading a file, this policy won't work for you.

Large number of zero-byte objects causing latency issues

If your Amazon S3 buckets contain a large number of these zero-byte objects, Transfer Family issues a lot of `HeadObject` calls, which can result in processing delays. The recommended solution for this issue is to enable **Optimized Directories** to reduce latency.

For example, suppose that you go into your home directory, and you have 10,000 subdirectories. In other words, your Amazon S3 bucket has 10,000 folders. In this scenario, if you run the `ls` (list) command, the list operation takes between six and eight minutes. However, if you optimize your directories, this operation takes only a few seconds. You set this option in the **Configure additional details** screen during the server creation or update procedure. These procedures are detailed under the [Configuring an SFTP, FTPS, or FTP server endpoint](#) topic.

Note

GUI clients may issue an `ls` command outside your control, so it is important to enable this setting if you can.

If you don't or can't optimize your directories, an alternate solution to this problem is to delete all of your zero-byte objects. Note the following:

- Empty directories will no longer exist. Directories only exist as a result of their names being in the key of an object.
- Doesn't prevent someone from calling `mkdir` and breaking things all over again. You could mitigate this by crafting a policy which prevents directory creation.
- Some scenarios make use of these 0-byte objects. For example, you have a structure like `/inboxes/customer1000` and the inbox directory gets cleaned every day.

Finally, one more possible solution is to limit the number of objects visible through a policy condition to reduce the number of `HeadObject` calls. For this to be a workable solution, you need to accept that you might only be able to view a limited set of all of your sub-directories.

Configure an Amazon EFS file system

AWS Transfer Family accesses Amazon Elastic File System (Amazon EFS) to service your users' transfer requests. So you must provide an Amazon EFS file system as part of setting up your file transfer protocol-enabled server. You can use an existing file system, or you can create a new one.

Note the following:

- When you use a Transfer Family server and an Amazon EFS file system, the server and the file system must be in the same AWS Region.
- The server and the file system don't need to be in the same account. If the server and file system are not in the same account, the file system policy must give explicit permission to the user role.

For information about how to set up multiple accounts, see [Managing the AWS accounts in your organization](#) in the *AWS Organizations User Guide*.

- When you set up your users, you assign them each an IAM role. This role determines the level of access that they have to your Amazon EFS file system.
- For details on mounting an Amazon EFS file system, see [Mounting Amazon EFS file systems](#).

For more details on how AWS Transfer Family and Amazon EFS work together, see [Using AWS Transfer Family to access files in your Amazon EFS file system](#) in the *Amazon Elastic File System User Guide*.

Amazon EFS file ownership

Amazon EFS uses the Portable Operating System Interface (POSIX) file permission model to represent file ownership.

In POSIX, users in the system are categorized into three distinct permission classes: When you allow a user to access files stored in an Amazon EFS file system using AWS Transfer Family, you must assign them a "POSIX profile." This profile is used to determine their access to files and directories in the Amazon EFS file system.

- **User (u):** Owner of the file or directory. Usually, the creator of a file or directory is also the owner.
- **Group (g):** Set of users that need identical access to files and directories that they share.
- **Others (o):** All other users that have access to the system except for the owner and group members. This permission class is also referred to as "Public."

In the POSIX permission model, every file system object (files, directories, symbolic links, named pipes, and sockets) is associated with the previously mentioned three sets of permissions. Amazon EFS objects have a Unix-style mode associated with them. This mode value defines the permissions for performing actions on that object.

Additionally, on Unix-style systems, users and groups are mapped to numeric identifiers, which Amazon EFS uses to represent file ownership. For Amazon EFS, objects are owned by a single owner and a single group. Amazon EFS uses the mapped numeric IDs to check permissions when a user attempts to access a file system object.

Set up Amazon EFS users for Transfer Family

Before you set up your Amazon EFS users, you can do either of the following:

- You can create users and set up their home folders in Amazon EFS. See [Configure Transfer Family users on Amazon EFS](#) for details.
- If you are comfortable adding a root user, you can [Create an Amazon EFS root user](#).

Note

Transfer Family servers do not support Amazon EFS access points to set POSIX permissions. Transfer Family users' POSIX profiles (described in the preceding section) offer the ability to

set POSIX permissions. These permissions are set at a user level, for granular access, based on UID, GID, and secondary GIDs.

Configure Transfer Family users on Amazon EFS

Transfer Family maps the users to the UID/GID and directories you specify. If the UID/GID/directories do not already exist in EFS, then you should create them before assigning them in Transfer to a user. The details for creating Amazon EFS users is described in [Working with users, groups, and permissions at the Network File System \(NFS\) Level](#) in the *Amazon Elastic File System User Guide*.

Steps to set up Amazon EFS users in Transfer Family

1. Map the EFS UID and GID for your user in Transfer Family using the [PosixProfile](#) fields.
2. If you want the user to start in a specific folder upon login, you can specify the EFS directory under the [HomeDirectory](#) field.

You can automate the process, by using a CloudWatch rule and Lambda function. For an example Lambda function that interacts with EFS, see [Using Amazon EFS for AWS Lambda in your serverless applications](#).

Additionally, you can configure logical directories for your Transfer Family users. For details, see the [Configure logical directories for Amazon EFS](#) section in the [Using logical directories to simplify your Transfer Family directory structures](#) topic.

Create an Amazon EFS root user

If your organization is comfortable for you to enable root user access via SFTP/FTPS for the configuration of your users, you can create a user who's UID and GID are 0 (root user), then use that root user to create folders and assign POSIX ID owners for rest of the users. The advantage of this option is that there is no need to mount the Amazon EFS file system.

Perform the steps described in [Adding Amazon EFS service-managed users](#), and for both the User ID and Group ID, enter 0 (zero).

Tip

Don't let this superuser account exist for longer than necessary. Or, if you do keep the root user account, make sure that you keep it well protected.

Supported Amazon EFS commands

The following commands are supported for Amazon EFS for AWS Transfer Family.

- `cd`
- `ls/dir`
- `pwd`
- `put`
- `get`
- `rename`
- `chown`: Only root (that is, users with `uid=0`) can change ownership and permissions of files and directories.
- `chmod`: Only root can change ownership and permissions of files and directories.
- `chgrp`: Supported either for root or for the file's owner who can only change a file's group to be one of their secondary groups.
- `ln -s/symlink`
- `mkdir`
- `rm/delete`
- `rmdir`
- `chmtime`

Create an IAM role and policy

This topic describes the types of policies and roles that can be used with AWS Transfer Family, and walks through the process of creating a user role. It also describes how session policies work and provides an example user role.

AWS Transfer Family uses the following types of roles:

- **User role** – Allows service-managed users to access the necessary Transfer Family resources. AWS Transfer Family assumes this role in the context of a Transfer Family user ARN.
- **Access role** – Provides access to only the Amazon S3 files that are being transferred. For inbound AS2 transfers, the access role uses the Amazon Resource Name (ARN) for the agreement. For outbound AS2 transfers, the access role uses the ARN for the connector.
- **Invocation role** – For use with Amazon API Gateway as the server's custom identity provider. Transfer Family assumes this role in the context of a Transfer Family server ARN.
- **Logging role** – Used to log entries into Amazon CloudWatch. Transfer Family uses this role to log success and failure details along with information about file transfers. Transfer Family assumes this role in the context of a Transfer Family server ARN. For outbound AS2 transfers, the logging role uses the connector ARN.
- **Execution role** – Allows a Transfer Family user to call and launch workflows. Transfer Family assumes this role in the context of a Transfer Family workflow ARN.

In addition to these roles, you can also use *session policies*. A session policy is used to limit access when necessary. Note that these policies are stand-alone: that is, you don't add these policies to a role. Rather, you add a session policy directly to a Transfer Family user.

Note

When you are creating a service-managed Transfer Family user, you can select **Auto-generate policy based on home folder**. This is a useful shortcut if you want to limit user access to their own folders. Also, you can view details about session policies and an example in [How session policies work](#). You can also find more information about session policies in [Session policies](#) in the *IAM User Guide*.

Topics

- [Create a user role](#)
- [How session policies work](#)
- [Example read/write access policy](#)

Create a user role

When you create a user, you make a number of decisions about user access. These decisions include which Amazon S3 buckets or Amazon EFS file systems that the user can access, what portions of each Amazon S3 bucket and which files in the file system are accessible, and what permissions the user has (for example, PUT or GET).

To set access, you create an identity-based AWS Identity and Access Management (IAM) policy and role that provide that access information. As part of this process, you provide access for your user to the Amazon S3 bucket or Amazon EFS file system that is the target or source for file operations. To do this, take the following high-level steps, described in detail later:

Create a user role

1. Create an IAM policy for AWS Transfer Family. This is described in [To create an IAM policy for AWS Transfer Family](#).
2. Create an IAM role and attach the new IAM policy. For an example, see [Example read/write access policy](#).
3. Establish a trust relationship between AWS Transfer Family and the IAM role. This is described in [To establish a trust relationship](#).

The following procedures describe how to create an IAM policy and role.

To create an IAM policy for AWS Transfer Family

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies**, and then choose **Create policy**.
3. On the **Create Policy** page, choose the **JSON** tab.
4. In the editor that appears, replace the contents of the editor with the IAM policy that you want attach to the IAM role.

You can grant read/write access or restrict users to their home directory. For more information, see [Example read/write access policy](#).

5. Choose **Review policy** and provide a name and description for your policy, and then choose **Create policy**.

Next, you create an IAM role and attach the new IAM policy to it.

To create an IAM role for AWS Transfer Family

1. In the navigation pane, choose **Roles**, and then choose **Create role**.

On the **Create role** page, make sure that **AWS service** is chosen.

2. Choose **Transfer** from the service list, and then choose **Next: Permissions**. This establishes a trust relationship between AWS Transfer Family and AWS.
3. In the **Attach permissions policies** section, locate and choose the policy that you just created, and choose **Next: Tags**.
4. (Optional) Enter a key and value for a tag, and choose **Next: Review**.
5. On the **Review** page, enter a name and description for your new role, and then choose **Create role**.

Next, you establish a trust relationship between AWS Transfer Family and AWS.

To establish a trust relationship

Note

In our examples, we use both `ArnLike` and `ArnEquals`. They are functionally identical, and therefore you may use either when you construct your policies. Transfer Family documentation uses `ArnLike` when the condition contains a wildcard character, and `ArnEquals` to indicate an exact match condition.

1. In the IAM console, choose the role that you just created.
2. On the **Summary** page, choose **Trust relationships**, and then choose **Edit trust relationship**.
3. In the **Edit Trust Relationship** editor, make sure **service** is `"transfer.amazonaws.com"`. The access policy is shown following.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      }
    }
  ],
```

```

    "Action": "sts:AssumeRole"
  }
]
}

```

We recommend that you use the `aws:SourceAccount` and `aws:SourceArn` condition keys to protect yourself against the confused deputy problem. The source account is the owner of the server and the source ARN is the ARN of the user. For example:

```

"Condition": {
  "StringEquals": {
    "aws:SourceAccount": "account_id"
  },
  "ArnLike": {
    "aws:SourceArn": "arn:aws:transfer:region:account_id:user/*"
  }
}

```

You can also use the `ArnLike` condition if you are looking to restrict to a particular server instead of any server in the user account. For example:

```

"Condition": {
  "ArnLike": {
    "aws:SourceArn": "arn:aws:transfer:region:account-id:user/server-id/*"
  }
}

```

Note

In the examples above, replace each *user input placeholder* with your own information.

For details on the confused deputy problem and more examples, see [Cross-service confused deputy prevention](#).

4. Choose **Update Trust Policy** to update the access policy.

You have now created an IAM role that allows AWS Transfer Family to call AWS services on your behalf. You attached to the role the IAM policy that you created to give access to your user. In the [Getting started with AWS Transfer Family server endpoints](#) section, this role and policy are assigned to your user or users.

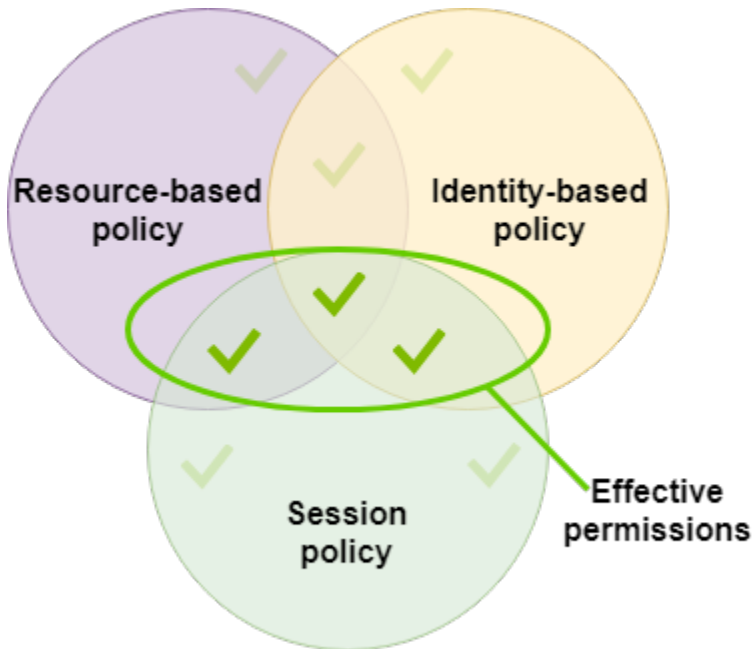
See also

- For more general information about IAM roles, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.
- To learn more about identity-based policies for Amazon S3 resources, see [Identity and access management in Amazon S3](#) in the *Amazon Simple Storage Service User Guide*.
- To learn more about identity-based policies for Amazon EFS resources, see [Using IAM to control file system data access](#) in the *Amazon Elastic File System User Guide*.

How session policies work

When an administrator creates a role, the role often includes broad permissions to cover multiple use cases or team members. If an administrator configures a [console URL](#), they can reduce permissions for the resulting session by using a *session policy*. For example, if you create a role with [read/write access](#), you can set up a URL that limits users' access to only their home directories.

Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or user. Session policies are useful for locking down users so that they have access only to portions of your bucket where object prefixes contain their username. The following diagram shows that the session policy's permissions are the intersection of the session policies and the resource-based policies plus the intersection of the session policies and identity-based policies.



For more details, see [Session policies](#) in the *IAM User Guide*.

In AWS Transfer Family, a session policy is supported only when you are transferring to or from Amazon S3. The following example policy is a session policy that limits users' access to their home directories only. Note the following:

- The `GetObjectACL` and `PutObjectACL` statements are only required if you need to enable Cross Account Access. That is, your Transfer Family server needs to access a bucket in a different account.
- The maximum length of a session policy is 2048 characters. For more details, see the [Policy request parameter](#) for the `CreateUser` action in the *API reference*.
- If your Amazon S3 bucket is encrypted using AWS Key Management Service (AWS KMS), you must specify additional permissions in your policy. For details, see [Data encryption in Amazon S3](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListingOfUserFolder",
      "Action": [
        "s3:ListBucket"
      ],
      "Effect": "Allow",
```

```

    "Resource": [
      "arn:aws:s3:::${transfer:HomeBucket}"
    ],
    "Condition": {
      "StringLike": {
        "s3:prefix": [
          "${transfer:HomeFolder}/*",
          "${transfer:HomeFolder}"
        ]
      }
    }
  },
  {
    "Sid": "HomeDirObjectAccess",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:DeleteObject",
      "s3:DeleteObjectVersion",
      "s3:GetObjectVersion",
      "s3:GetObjectACL",
      "s3:PutObjectACL"
    ],
    "Resource": "arn:aws:s3:::${transfer:HomeDirectory}/*"
  }
]
}

```

Note

The preceding policy example assumes that users have their home directories set to include a trailing slash, to signify that it is a directory. If, on the other hand, you set a user's `HomeDirectory` without the trailing slash, then you should include it as part of your policy.

In the previous example policy, note the use of the `transfer:HomeFolder`, `transfer:HomeBucket`, and `transfer:HomeDirectory` policy parameters. These parameters are set for the `HomeDirectory` that is configured for the user, as described in [HomeDirectory](#) and [Implementing your API Gateway method](#). These parameters have the following definitions:

- The `transfer:HomeBucket` parameter is replaced with the first component of `HomeDirectory`.
- The `transfer:HomeFolder` parameter is replaced with the remaining portions of the `HomeDirectory` parameter.
- The `transfer:HomeDirectory` parameter has the leading forward slash (/) removed so that it can be used as part of an S3 Amazon Resource Name (ARN) in a Resource statement.

Note

If you are using logical directories—that is, the user's `homeDirectoryType` is `LOGICAL`—these policy parameters (`HomeBucket`, `HomeDirectory`, and `HomeFolder`) are not supported.

For example, assume that the `HomeDirectory` parameter that is configured for the Transfer Family user is `/home/bob/amazon/stuff/`.

- `transfer:HomeBucket` is set to `/home`.
- `transfer:HomeFolder` is set to `/bob/amazon/stuff/`.
- `transfer:HomeDirectory` becomes `home/bob/amazon/stuff/`.

The first "Sid" allows the user to list all directories starting from `/home/bob/amazon/stuff/`.

The second "Sid" limits the user's put and get access to that same path, `/home/bob/amazon/stuff/`.

Example read/write access policy

Grant read/write access to Amazon S3 bucket

The following example policy for AWS Transfer Family grants read/write access to objects in your Amazon S3 bucket.

Note the following:

- Replace `DOC-EXAMPLE-BUCKET` with the name of your Amazon S3 bucket.

- The `GetObjectACL` and `PutObjectACL` statements are only required if you need to enable Cross Account Access. That is, your Transfer Family server needs to access a bucket in a different account.
- The `GetObjectVersion` and `DeleteObjectVersion` statements are only required if versioning is enabled on the Amazon S3 bucket that is being accessed.

Note

If you have *ever* enabled versioning for your bucket, then you need these permissions, as you can only suspend versioning in Amazon S3, and not turn it off completely. For details, see [Unversioned, versioning-enabled, and versioning-suspended buckets](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListingOfUserFolder",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
      ]
    },
    {
      "Sid": "HomeDirObjectAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:GetObjectVersion",
        "s3:GetObjectACL",
        "s3:PutObjectACL"
      ],
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
    }
  ]
}
```

```
]
}
```

Grant file system access to files in Amazon EFS file system

Note

In addition to the policy, you must also make sure your POSIX file permissions are granting the appropriate access. For more information, see [Working with users, groups, and permissions at the Network File System \(NFS\) Level](#) in the *Amazon Elastic File System User Guide*.

The following example policy grants root file system access to files in your Amazon EFS file system.

Note

In the following examples, replace *region* with your region, *account-id* with the account the file is in, and *file-system-id* with the ID of your Amazon Elastic File System (Amazon EFS).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RootFileSystemAccess",
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:ClientRootAccess",
        "elasticfilesystem:ClientMount",
        "elasticfilesystem:ClientWrite"
      ],
      "Resource": "arn:aws:elasticfilesystem:region:account-id:file-system/file-system-id"
    }
  ]
}
```

The following example policy grants user file system access to files in your Amazon EFS file system.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UserFileSystemAccess",
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:ClientMount",
        "elasticfilesystem:ClientWrite"
      ],
      "Resource": "arn:aws:elasticfilesystem:region:account-id:file-system/file-
system-id"
    }
  ]
}
```

Transfer Family tutorials

The AWS Transfer Family user guide provides detailed walkthroughs for several use cases.

- [Getting started with AWS Transfer Family server endpoints](#): this tutorial walks you through creating an SFTP Transfer Family server and service-managed user, then shows how to transfer a file using a client.
- [Setting up and using SFTP connectors](#): this tutorial illustrates how to set up an SFTP connector, and then transfer files between Amazon S3 storage and an SFTP server.
- [Setting up an Amazon API Gateway method as a custom identity provider](#) : this tutorial illustrates how to set up an Amazon API Gateway method and use it as a custom identity provider to upload files to an AWS Transfer Family server.
- [Setting up a managed workflow for decrypting a file](#): this tutorial illustrates how to set up a managed workflow that contains a decrypt step, and how to upload an encrypted file to an Amazon S3 bucket and then view the decrypted file.
- [Setting up an AS2 configuration](#): this tutorial walks through the steps needed to configure an AS2 Transfer Family server. There are instructions for importing certificates, creating profiles and agreements, optionally creating an AS2 connector, and then testing the configuration.

Topics

- [Getting started with AWS Transfer Family server endpoints](#)
- [Setting up a managed workflow for decrypting a file](#)
- [Setting up and using SFTP connectors](#)
- [Setting up an Amazon API Gateway method as a custom identity provider](#)
- [Setting up an AS2 configuration](#)

Getting started with AWS Transfer Family server endpoints

Use this tutorial to get started with AWS Transfer Family (Transfer Family). You'll learn how to create an SFTP-enabled server with publicly accessible endpoint using Amazon S3 storage, add a user with service-managed authentication, and transfer a file with Cyberduck.

Topics

- [Prerequisites](#)
- [Step 1: Sign in to the AWS Transfer Family console](#)
- [Step 2: Create an SFTP-enabled server](#)
- [Step 3: Add a service managed user](#)
- [Step 4: Transfer a file using a client](#)

Prerequisites

Before you begin, be sure to complete the requirements in [Prerequisites](#). As part of this setup, you create an Amazon Simple Storage Service (Amazon S3) bucket and an AWS Identity and Access Management (IAM) user role.

There are permissions required for using the AWS Transfer Family console, and there are permissions required for configuring other AWS services that Transfer Family uses, such as Amazon Simple Storage Service, AWS Certificate Manager, Amazon Elastic File System, and Amazon Route 53. For example, for users that are transferring files into and out of AWS using Transfer Family, **AmazonS3FullAccess** grants permissions to setup and use an Amazon S3 bucket. Some of the permissions in this policy are needed to create Amazon S3 buckets.

To use the Transfer Family console, you require the following:

- **AWSTransferConsoleFullAccess** grants permissions for your SFTP user to create Transfer Family resources.
- **IAMFullAccess** (or specifically a policy that allows creation of IAM roles) is only needed if you want Transfer Family to automatically create a logging role for your server in Amazon CloudWatch Logs or a user role for a user logging into a server.
- To create and delete VPC server types, you need to add the actions **ec2:CreateVpcEndpoint** and **ec2>DeleteVpcEndpoints** to your policy.

Note

The **AmazonS3FullAccess** and **IAMFullAccess** policies are, themselves, not needed for general usage of AWS Transfer Family. They are presented here as a simple way to make sure that all of the permissions that you need are covered. Additionally, these are AWS managed policies, which are standard policies that are available to all AWS customers. You

can view the individual permissions in these policies and determine a minimal set that you need for your purposes.

Step 1: Sign in to the AWS Transfer Family console

To sign in to Transfer Family

1. Sign in to the AWS Management Console and open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. For **Account ID or alias**, enter the ID for your AWS account.
3. For **IAM user name**, enter the name of the user role that you created for Transfer Family.
4. For **Password**, enter your AWS account password.
5. Choose **Sign in**.


Step 2: Create an SFTP-enabled server

Secure Shell (SSH) File Transfer Protocol (SFTP) is a network protocol used for secure transfer of data over the internet. The protocol supports the full security and authentication functionality of SSH. It is widely used to exchange data, including sensitive information between business partners in a variety of industries such as financial services, healthcare, retail, and advertising.

To create an SFTP-enabled server

1. Select **Servers** from the Navigation pane then choose **Create server**.
2. In **Choose protocols**, select **SFTP**, and then choose **Next**.
3. In **Choose an identity provider**, choose **Service managed** to store user identities and keys in Transfer Family, and then choose **Next**.
4. In **Choose an endpoint**, do the following:
 - a. For **Endpoint type**, choose the **Publicly accessible** endpoint type.
 - b. For **Custom hostname**, choose **None**.
 - c. Choose **Next**.
5. In **Choose a domain**, choose **Amazon S3**.

6. In **Configure additional details**, for **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server. Our latest security policy is the default: for details, see [Security policies for AWS Transfer Family servers](#).

 **Note**

Only if you are adding a managed workflow for your server, choose **Create a new role** for **CloudWatch logging**. To log server events, you do not need to create an IAM role.

7. In **Review and create**, choose **Create server**. You are taken to the **Servers** page.

It can take a couple of minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations, but you'll need to create a user first. For details on creating users, see [Managing users for server endpoints](#).

Step 3: Add a service managed user

To add a user to the SFTP-enabled server

1. On the **Servers** page, select the server that you want to add a user to.
2. Choose **Add user**.
3. In the **User configuration** section, for **Username**, enter the username. This username must be a minimum of 3 and a maximum of 100 characters. You can use the following characters in the username: a–z, A–Z, 0–9, underscore '_', hyphen '-', period '.', and at sign (@). The username can't start with a hyphen, period, or at sign.
4. For **Access**, choose the IAM role that you created in [Create an IAM role and policy](#). This IAM role includes an IAM policy that contains permissions to access your Amazon S3 bucket, as well as a trust relationship with the AWS Transfer Family service. The procedure outlined in [To establish a trust relationship](#) shows how to establish the proper trust relationship.
5. For **Policy**, choose **None**.
6. For **Home directory**, choose the Amazon S3 bucket where you want to store the data that you transfer using AWS Transfer Family. Enter the path to the home directory. This is the directory that your users see when they log in using their client.

We recommend using a directory path that contains the username so that you have the option to use a session policy. A session policy limits a user's access in the Amazon S3 bucket to that

user's home directory. For more information about using session policies, see [How session policies work](#).

If you prefer, you can keep this parameter blank to use your Amazon S3 bucket's root directory. If you choose this option, make sure that your IAM role provides access to the root directory.

7. Select the **Restricted** check box to prevent your users from accessing anything outside of their home directory. This also prevents users from seeing the Amazon S3 bucket name or folder name.
8. For **SSH public key**, enter the public SSH key portion of the SSH key pair in `ssh-rsa <string>` format.

Your key must be validated by the service before you can add your new user. For more information about how to generate an SSH key pair, see [Generate SSH keys for service-managed users](#).

9. (Optional) For **Key** and **Value**, enter one or more tags as key-value pairs, and choose **Add tag**.
10. Choose **Add** to add your new user to the server that you chose.

The new user appears in the **Users** section of the **Server details** page.

Step 4: Transfer a file using a client

You transfer files over the AWS Transfer Family service by specifying the transfer operation in a client. AWS Transfer Family supports several clients. For details, see [Transferring files over a server endpoint using a client](#)

This section contains procedures for using Cyberduck and OpenSSH.

Topics

- [Use Cyberduck](#)
- [Use OpenSSH](#)

Use Cyberduck

To transfer files over AWS Transfer Family using Cyberduck

1. Open the [Cyberduck](#) client.

2. Choose **Open Connection**.
3. In the **Open Connection** dialog box, choose **SFTP (SSH File Transfer Protocol)**.
4. For **Server**, enter your server endpoint. The server endpoint is located on the **Server details** page, see [View SFTP, FTPS, and FTP server details](#).
5. For **Port number**, enter **22** for SFTP.
6. For **Username**, enter the name for the user that you created in [Managing users for server endpoints](#).
7. For **SSH Private Key**, choose or enter the SSH private key.
8. Choose **Connect**.
9. Perform your file transfer.

Depending on where your files are, do one of the following:

- In your local directory (the source), choose the files that you want to transfer, and drag and drop them into the Amazon S3 directory (the target).
- In the Amazon S3 directory (the source), choose the files that you want to transfer, and drag and drop them into your local directory (the target).

Use OpenSSH

Use the instructions that follow to transfer files from the command line using OpenSSH.

Note

This client works only with an SFTP-enabled server.

To transfer files over AWS Transfer Family using the OpenSSH command line utility

1. On Linux or Macintosh, open a command terminal.
2. At the prompt, enter the following command: `% sftp -i transfer-key sftp_user@service_endpoint`

In the preceding command, `sftp_user` is the username and `transfer-key` is the SSH private key. Here, `service_endpoint` is the server's endpoint as shown in the AWS Transfer Family console for the selected server.

An sftp prompt should appear.

3. (Optional) To view the user's home directory, enter the following command at the sftp prompt: `sftp> pwd`
4. On the next line, enter the following text: `sftp> cd /mybucket/home/sftp_user`

In this getting-started exercise, this Amazon S3 bucket is the target of the file transfer.

5. On the next line, enter the following command: `sftp> put filename.txt`

The put command transfers the file into the Amazon S3 bucket.

A message like the following appears, indicating that the file transfer is in progress, or complete.

```
Uploading filename.txt to /my-bucket/home/sftp_user/filename.txt
some-file.txt 100% 127 0.1KB/s 00:00
```

Setting up a managed workflow for decrypting a file

This tutorial illustrates how to set up a managed workflow that contains a decrypt step. The tutorial also shows how to upload an encrypted file to an Amazon S3 bucket and then view the decrypted file in that same bucket.

Note

The AWS storage blog has a post that describes how to simply decrypt files without writing any code using Transfer Family Managed workflows, [Encrypt and decrypt files with PGP and AWS Transfer Family](#).

Topics

- [Step 1: Configure an execution role](#)
- [Step 2: Create a managed workflow](#)
- [Step 3: Add the workflow to a server and create a user](#)
- [Step 4: Create a PGP key pair](#)
- [Step 5: Store the PGP private key in AWS Secrets Manager](#)

- [Step 6: Encrypt a file](#)
- [Step 7: Run the workflow and view the results](#)

Step 1: Configure an execution role

Create an AWS Identity and Access Management (IAM) execution role that Transfer Family can use to launch a workflow. The process of creating an execution role is described in [IAM policies for workflows](#).

Note

As part of creating an execution role, make sure to establish a trust relationship between the execution role and Transfer Family, as described in [To establish a trust relationship](#).

The following execution role policy contains all the required permissions to start the workflow that you create in this tutorial. To use this example policy, replace the *user input placeholders* with your own information. Replace DOC-EXAMPLE-BUCKET with the name of the Amazon S3 bucket where you upload your encrypted files.

Note

Not every workflow requires every permission that's listed in this example. You can restrict permissions based on the types of steps in your specific workflow. The permissions needed for each predefined step type are described in [Use predefined steps](#). The permissions needed for a custom step are described in [IAM permissions for a custom step](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WorkflowsS3Permissions",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectTagging",
        "s3:GetObjectVersion",
        "s3:PutObject",
```

```

        "s3:PutObjectTagging",
        "s3:ListBucket",
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject"
    ],
    "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET"]
    "Condition": {
        "StringEquals": {
            "s3:RequestObjectTag/Archive": "yes"
        }
    }
},
{
    "Sid": "DecryptSecret",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue"
    ],
    "Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/transfer/
*"
    }
]
}

```


Step 2: Create a managed workflow

Now you need to create a workflow that contains a decrypt step.

To create a workflow that contains a decrypt step

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Workflows**, and then choose **Create workflow**.
3. Enter the following details:
 - Enter a description, for example **Decrypt workflow example**.
 - In the **Nominal steps** section, choose **Add step**.
4. For **Choose step type**, choose **Decrypt file**, and then choose **Next**.
5. In the **Configure parameters** dialog box, specify the following:

- Enter a descriptive step name, for example, **decrypt-step**. Spaces are not allowed in step names.
- For the **Destination for decrypted files**, choose Amazon S3.
- For the **Destination bucket name**, choose the same Amazon S3 bucket that you specified as the DOC-EXAMPLE-BUCKET in the IAM policy that you created in Step 1.
- For the **Destination key prefix**, enter the name of the prefix (folder) where you want to store your decrypted files in your destination bucket, for example, **decrypted-files/**.

 **Note**

Make sure to add a trailing slash (/) to your prefix.

- For this tutorial, leave **Overwrite existing** cleared. When this setting is cleared, if you try to decrypt a file with the identical name of an existing file, the workflow processing stops, and the new file is not processed.

Choose **Next** to move to the review screen.

6. Review the details for the step. If everything is correct, choose **Create step**.
7. Your workflow needs only the single decrypt step, so there are no additional steps to configure. Choose **Create workflow** to create the new workflow.

Note the workflow ID for your new workflow. You will need this ID for the next step. This tutorial uses *w-1234abcd5678efghi* as the example workflow ID.

Step 3: Add the workflow to a server and create a user

Now that you have a workflow with a decrypt step, you must associate it with a Transfer Family server. This tutorial shows how to attach the workflow to an existing Transfer Family server. Alternatively, you can create a new server to use with your workflow.

After you attach the workflow to a server, you must create a user that can SFTP into the server and trigger the workflow to run.

To configure a Transfer Family server to run a workflow

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.

2. In the left navigation pane, choose **Servers**, and then choose a server from the list. Make sure that this server supports the SFTP protocol.
3. On the details page for the server, scroll down to the **Additional details** section, and then choose **Edit**.
4. On the **Edit additional details** page, in the **Managed workflows** section, choose your workflow, and choose a corresponding execution role.
 - For **Workflow for complete file uploads**, choose the workflow that you created in [Step 2: Create a managed workflow](#), for example, **w-1234abcd5678efghi**.
 - For **Managed workflows execution role**, choose the IAM role that you created in [Step 1: Configure an execution role](#).
5. Scroll to the bottom of the page, and choose **Save** to save your changes.

Note the ID for the server that you are using. The name of the AWS Secrets Manager secret that you use to store your PGP keys is based in part on the server ID.

To add a user that can trigger the workflow

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Servers**, and then choose the server that you're using for the decrypt workflow.
3. On the server details page, scroll down to the **Users** section, and choose **Add user**.
4. For your new user, enter the following details:
 - For **Username**, enter **decrypt-user**.
 - For **Role**, choose a user role that can access your server.
 - For **Home directory**, choose the Amazon S3 bucket that you used earlier, for example, **DOC-EXAMPLE-BUCKET**.
 - For **SSH public keys**, paste in a public key that corresponds to a private key that you have. For details, see [Generate SSH keys for service-managed users](#).
5. Choose **Add** to save your new user.

Note the name of your Transfer Family user for this server. The secret is partially based on the name of the user. For simplicity, this tutorial uses a default secret that can be used by any user of the server.

Step 4: Create a PGP key pair

Use one of the [supported PGP clients](#) to generate a PGP key pair. This process is described in detail in [Generate PGP keys](#).

To generate a PGP key pair

1. For this tutorial, you can use gpg (GnuPG) version 2.0.22 client to generate a PGP key pair that uses RSA as the encryption algorithm. For this client, run the following command, and provide an email address and a passphrase. You can use any name or email address that you like. Make sure that you remember the values that you use, because you will need to enter them later in the tutorial.

```
gpg --gen-key
```

Note

If you're using GnuPG version 2.3.0 or newer, you must run `gpg --full-gen-key`. When prompted for the type of key to create, choose RSA or ECC. However, if you choose ECC, make sure to choose either NIST or BrainPool for the elliptic curve. **Do not** choose Curve 25519.

2. Export the private key by running the following command. Replace *user@example.com* with the email address that you used when you generated the key.

```
gpg --output workflow-tutorial-key.pgp --armor --export-secret-key user@example.com
```

This command exports the private key to the **workflow-tutorial-key.pgp** file. You can name the output file anything that you like. You can also delete the private key file after you have added it to AWS Secrets Manager.

Step 5: Store the PGP private key in AWS Secrets Manager

You need to store the private key in Secrets Manager, in a very specific way, so that the workflow can find the private key when the workflow runs a decrypt step on an uploaded file.

Note

When you store secrets in Secrets Manager, your AWS account incurs charges. For information about pricing, see [AWS Secrets Manager Pricing](#).

To store a PGP private key in Secrets Manager

1. Sign in to the AWS Management Console and open the AWS Secrets Manager console at <https://console.aws.amazon.com/secretsmanager/>.
2. In the left navigation pane, choose **Secrets**.
3. On the **Secrets** page, choose **Store a new secret**.
4. On the **Choose secret type** page, for **Secret type**, choose **Other type of secret**.
5. In the **Key/value pairs** section, choose the **Key/value** tab.
 - **Key** – Enter **PGPprivateKey**.
 - **value** – Paste the text of your private key into the value field.
6. Choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.
 - **Key** – Enter **PGPPassphrase**.
 - **value** – Enter the passphrase that you used when you generated your PGP key pair in [Step 4: Create a PGP key pair](#).
7. Choose **Next**.
8. On the **Configure secret** page, enter a name and description for your secret. You can create a secret for a specific user or one that can be used by all users. If your server ID is *s-11112222333344445*, you name the secret as follows.
 - To create a default secret for all users, name the secret **aws/transfer/s-11112222333344445/@pgp-default**.
 - To create a secret only for the user that you created earlier, name the secret **aws/transfer/s-11112222333344445/decrypt-user**.
9. Choose **Next**, and then accept the defaults on the **Configure rotation** page. Then choose **Next**.
10. On the **Review** page, choose **Store** to create and store the secret.

For more information about adding your PGP private key to Secrets Manager, see [Use AWS Secrets Manager to store your PGP key](#).

Step 6: Encrypt a file

Use the `gpg` program to encrypt a file for use in your workflow. Run the following command to encrypt a file:

```
gpg -e -r marymajor@example.com --openpgp testfile.txt
```

Before running this command, note the following:

- For the `-r` argument, replace *marymajor@example.com* with the email address that you used when you created the PGP key pair.
- The `--openpgp` flag is optional. This flag makes the encrypted file conform to the [OpenPGP RFC4880](#) standard.
- This command creates a file named **testfile.txt.gpg** in the same location as **testfile.txt**.

Step 7: Run the workflow and view the results

To run the workflow, you connect to the Transfer Family server with the user that you created in Step 3. Then you can look in the Amazon S3 bucket that you specified in [Step 2.5, configure destination parameters](#) to see the decrypted file.

To run the decrypt workflow

1. Open a command terminal.
2. Run the following command, replacing *your-endpoint* with your actual endpoint, and *transfer-key* with your user's SSH private key:

```
sftp -i transfer-key decrypt-user@your-endpoint
```

For example, if the private key is stored in `~/ .ssh/decrypt-user`, and your endpoint is `s-11112222333344445.server.transfer.us-east-2.amazonaws.com`, the command is as follows:

```
sftp -i ~/.ssh/decrypt-user decrypt-user@s-11112222333344445.server.transfer.us-east-2.amazonaws.com
```

3. Run the `pwd` command. If successful, this command will return the following:

```
Remote working directory: /DOC-EXAMPLE-BUCKET/decrypt-user
```

Your directory reflects the name of your Amazon S3 bucket.

4. Run the following command to upload the file and trigger the workflow to run:

```
put testfile.txt.gpg
```

5. For the destination of the decrypted files, you specified the `decrypted-files/` folder when you created the workflow. Now, you can navigate to that folder and list the contents.

```
cd ../decrypted-files/  
ls
```

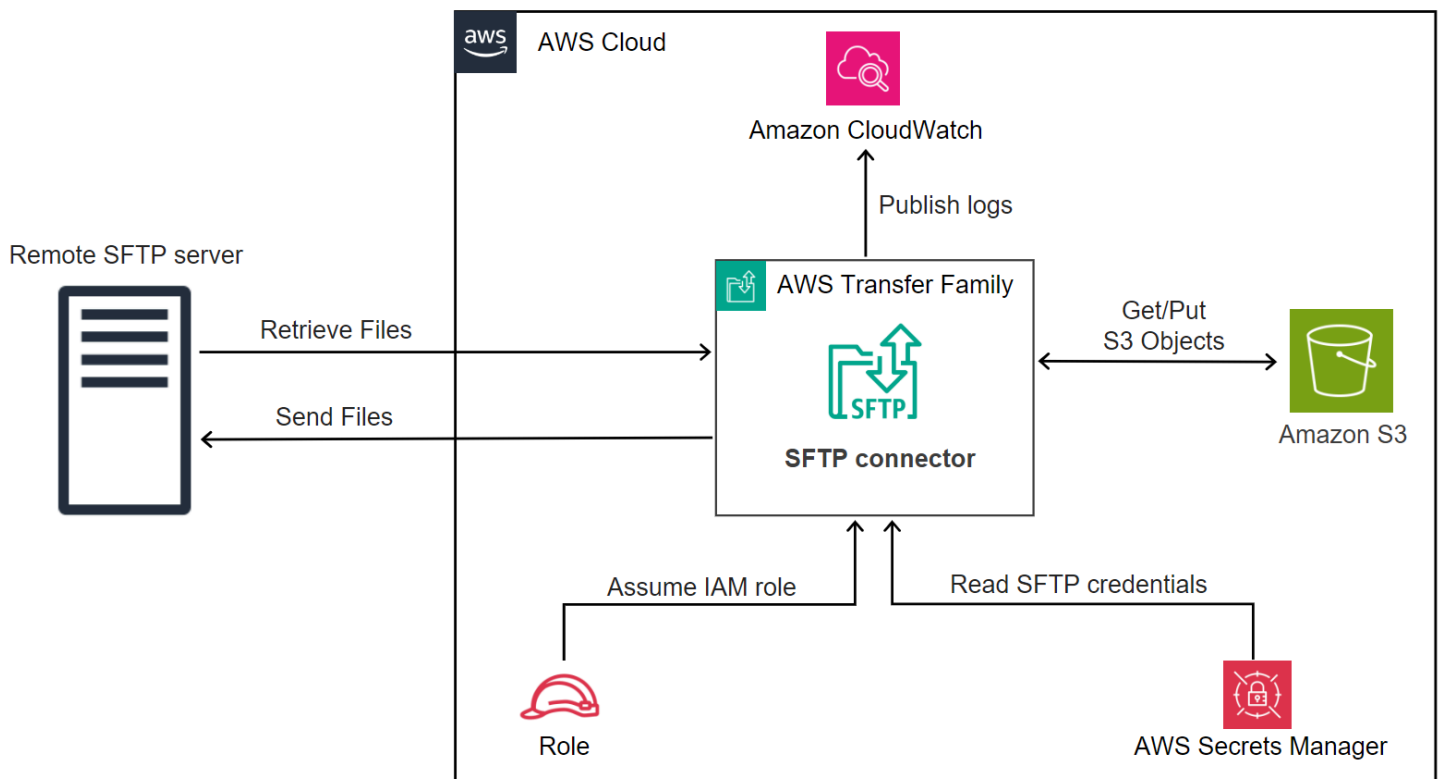
If successful, the `ls` command lists the `testfile.txt` file. You can download this file and verify that it is the same as the original file that you encrypted earlier.

Setting up and using SFTP connectors

The purpose of a connector is to establish a relationship between your AWS storage and a partner's SFTP server. You can send files from Amazon S3 to an external, partner-owned destination. You can also use an SFTP connector to retrieve files from a partner's SFTP server.

This tutorial illustrates how to set up an SFTP connector, and then transfer files between Amazon S3 storage and an SFTP server.

An SFTP connector retrieves SFTP credentials from AWS Secrets Manager to authenticate into a remote SFTP server and establish a connection. The connector sends files to or retrieves files from the remote server, and stores the files in Amazon S3. An IAM role is used to allow access to the Amazon S3 bucket and to the credentials stored in Secrets Manager. And you can log to Amazon CloudWatch.



The following blog posts provides a reference architecture to build an MFT workflow using SFTP connectors, including encryption of files using PGP before sending them to a remote SFTP server using SFTP connectors: [Architecting secure and compliant managed file transfers with AWS Transfer Family SFTP connectors and PGP encryption.](#)

Topics

- [Step 1: Create the necessary supporting resources](#)
- [Step 2: Create and test an SFTP connector](#)
- [Step 3: Send and retrieve files using the SFTP connector](#)
- [Procedures to create a Transfer Family server to use as your remote SFTP server](#)

Step 1: Create the necessary supporting resources

You can use SFTP connectors to copy files between Amazon S3 and any remote SFTP server. For this tutorial, we are using an AWS Transfer Family server as our remote SFTP server. We need to create and configure the following resources:

- Create Amazon S3 buckets to store files in your AWS environment, and to send and retrieve files from the remote SFTP server: [Create Amazon S3 buckets.](#)

- Create an AWS Identity and Access Management role for accessing Amazon S3 storage and our secret in Secrets Manager: [Create an IAM role with the necessary permissions](#).
- Create a Transfer Family server that uses the SFTP protocol, and a service-managed user that uses the SFTP connector to transfer files to or from the SFTP server: [Create a Transfer Family SFTP server and a user](#).
- Create an AWS Secrets Manager secret that stores the credentials used by the SFTP connector to log in to the remote SFTP server: [Create and store a secret in AWS Secrets Manager](#).

Create Amazon S3 buckets

To create an Amazon S3 bucket

1. Sign in to the AWS Transfer Family console at <https://console.aws.amazon.com/s3/>.
2. Choose a Region and enter a name.

For this tutorial, our bucket is in **US East (N. Virginia) us-east-1**, and the name is **sftp-server-storage-east**.

3. Accept the defaults and choose **Create bucket**.

For complete details about creating Amazon S3 buckets, see [How do I create an S3 bucket?](#) in the *Amazon Simple Storage Service User Guide*.

Create an IAM role with the necessary permissions

For the access role, create a policy with the following permissions.

The following example grants the necessary permissions to access the **DOC-EXAMPLE-BUCKET** in Amazon S3, and the specified secret stored in Secrets Manager.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListingOfUserFolder",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
```

```

    "Resource": [
      "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
    ]
  },
  {
    "Sid": "HomeDirObjectAccess",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:DeleteObject",
      "s3:DeleteObjectVersion",
      "s3:GetObjectVersion",
      "s3:GetObjectACL",
      "s3:PutObjectACL"
    ],
    "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
  },
  {
    "Sid": "GetConnectorSecretValue",
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/transfer/SecretName-6RandomCharacters"
  }
]
}

```

Replace items as follows:

- For DOC-EXAMPLE-BUCKET, the tutorial uses **sftp-server-storage-east**.
- For *region*, the tutorial uses **us-east-1**.
- For *account-id*, use your AWS account ID.
- For *SecretName-6RandomCharacters*, we are **using sftp-connector1** for the name (you will have your own six random characters for your secret).

You must also make sure that this role contains a trust relationship that allows the connector to access your resources when servicing your users' transfer requests. For details on establishing a trust relationship, see [To establish a trust relationship](#).

Note

To see the details for the role that we are using for the tutorial, see [Combined user and access role](#).

Create and store a secret in AWS Secrets Manager

We need to store a secret in Secrets Manager to store user credentials for your SFTP connector. You can use a password, SSH private key, or both. For the tutorial, we are using a private key.

Note

When you store secrets in Secrets Manager, your AWS account incurs charges. For information about pricing, see [AWS Secrets Manager Pricing](#).

Before you begin the procedure to store the secret, retrieve and format your private key. The private key must correspond to the public key that is configured for the user on the remote SFTP server. For our tutorial, the private key must correspond to the public key that is stored for our test user on the Transfer Family SFTP server that we are using as remote server.

To do this, run the following command:

```
jq -sR . path-to-private-key-file
```

For example, if your private key file is located in `~/.ssh/sftp-testuser-privatekey`, the command is as follows.

```
jq -sR . ~/.ssh/sftp-testuser-privatekey
```

This outputs the key in the correct format (with embedded newline characters) to standard output. Copy this text somewhere, as you need to paste it in the following procedure (in step 6).

To store user credentials in Secrets Manager for an SFTP connector

1. Sign in to the AWS Management Console and open the AWS Secrets Manager console at <https://console.aws.amazon.com/secretsmanager/>.
2. In the left navigation pane, choose **Secrets**.

3. On the **Secrets** page, choose **Store a new secret**.
 4. On the **Choose secret type** page, for **Secret type**, choose **Other type of secret**.
 5. In the **Key/value pairs** section, choose the **Key/value** tab.
 - **Key** – Enter **Username**.
 - **value** – Enter the name of our user, **sftp-testuser**.
 6. To enter the key, we recommend that you use the **Plaintext** tab.
 - a. Choose **Add row**, then enter **PrivateKey**.
 - b. Choose the **Plaintext** tab. The field now contains the following text:


```
{"Username":"sftp-testuser","PrivateKey":""}
```
 - c. Paste in the text for your private key (saved earlier) between the empty double quotes ("").
- Your screen should look as follows (key data is grayed out).



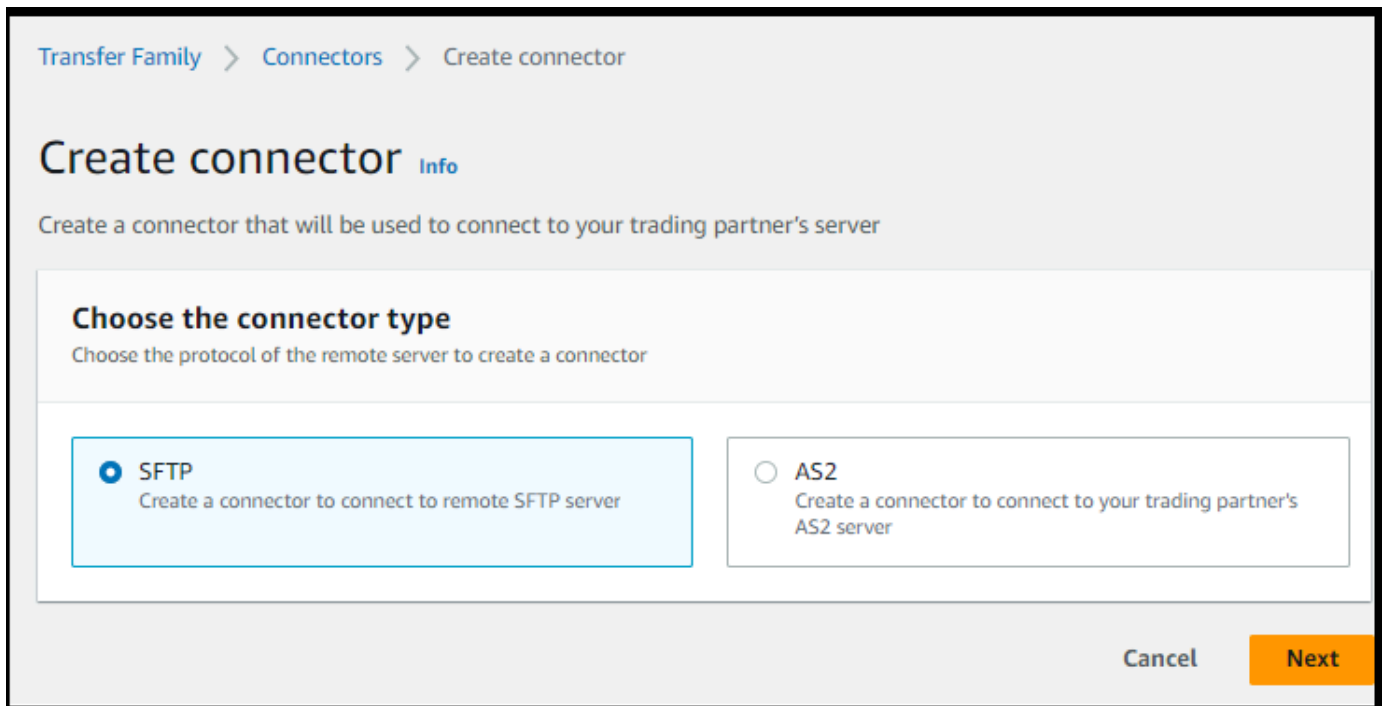
7. Choose **Next**.
8. On the **Configure secret** page, enter a name for your secret. For this tutorial, we name the secret **aws/transfer/sftp-connector1**.
9. Choose **Next**, and then accept the defaults on the **Configure rotation** page. Then choose **Next**.
10. On the **Review** page, choose **Store** to create and store the secret.

Step 2: Create and test an SFTP connector

In this section, we create an SFTP connector that uses all of the resources that we created earlier. For more details, see [Configure SFTP connectors](#).

To create an SFTP connector

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Connectors**, then choose **Create connector**.
3. Choose **SFTP** for the connector type to create an SFTP connector, and then choose **Next**.



Transfer Family > Connectors > Create connector

Create connector Info

Create a connector that will be used to connect to your trading partner's server

Choose the connector type

Choose the protocol of the remote server to create a connector

SFTP
Create a connector to connect to remote SFTP server

AS2
Create a connector to connect to your trading partner's AS2 server

Cancel **Next**

4. In the **Connector configuration** section, provide the following information:
 - For the **URL**, enter the URL of the remote SFTP server. For the tutorial, we enter the URL of the Transfer Family server that we are using as the remote SFTP server.

```
sftp://s-1111aaaa2222bbbb3.server.transfer.us-east-1.amazonaws.com
```

Replace *1111aaaa2222bbbb3* with your Transfer Family server ID.

- For the **Access role**, enter the role we created earlier, **sftp-connector-role**.
- For the **Logging role**, choose **AWSTransferLoggingAccess**.

Note

AWSTransferLoggingAccess is an AWS managed policy. This policy is described in detail in [AWS managed policy: AWSTransferLoggingAccess](#).

Connector configuration

URL

Specify the URL of remote server

Access role

IAM Role for Amazon S3 access and AWS Secrets Manager access

Logging role - optional [Info](#)

IAM role for the connector to push events to your CloudWatch logs



5. In the **SFTP Configuration** section, provide the following information:
- For **Connector credentials**, choose the name of your Secrets Manager resource that contains SFTP credentials. For the tutorial, choose **aws/transfer/sftp-connector1**.
 - For **Trusted host keys**, paste in the public portion of the host key. You can retrieve this key by running `ssh-keyscan` for your SFTP server. For details on how to format and store the trusted host key, see the [SftpConnectorConfig](#) data type documentation.

SFTP configuration Info

Connector credentials
Select the username and password / SSH private key that will be used to connect to the remote server from AWS Secret Manager

aws/transfer/sftp-connector1 ↕ ↻ Store a new secret [↗](#)

Trusted host keys
Connector connects to the remote server only if the SSH public key matches one of the below

ssh-rsa AAA... Remove

Add trusted host key

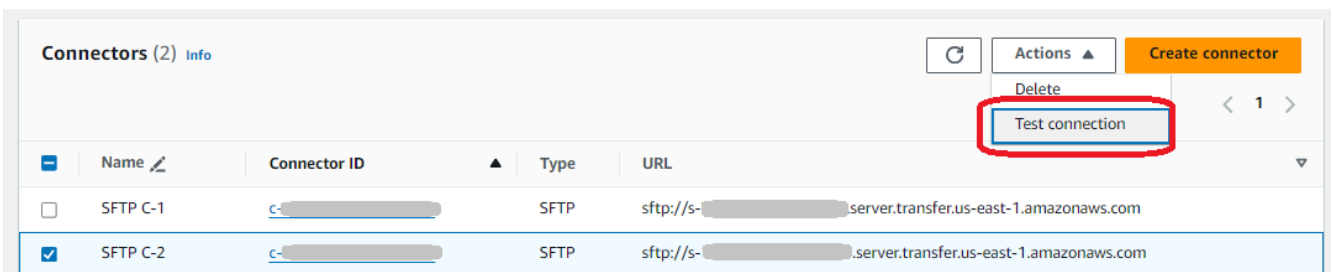
- After you have confirmed all of your settings, choose **Create connector** to create the SFTP connector.

After you create an SFTP connector, we recommend that you test it before you attempt to transfer any files using your new connector.

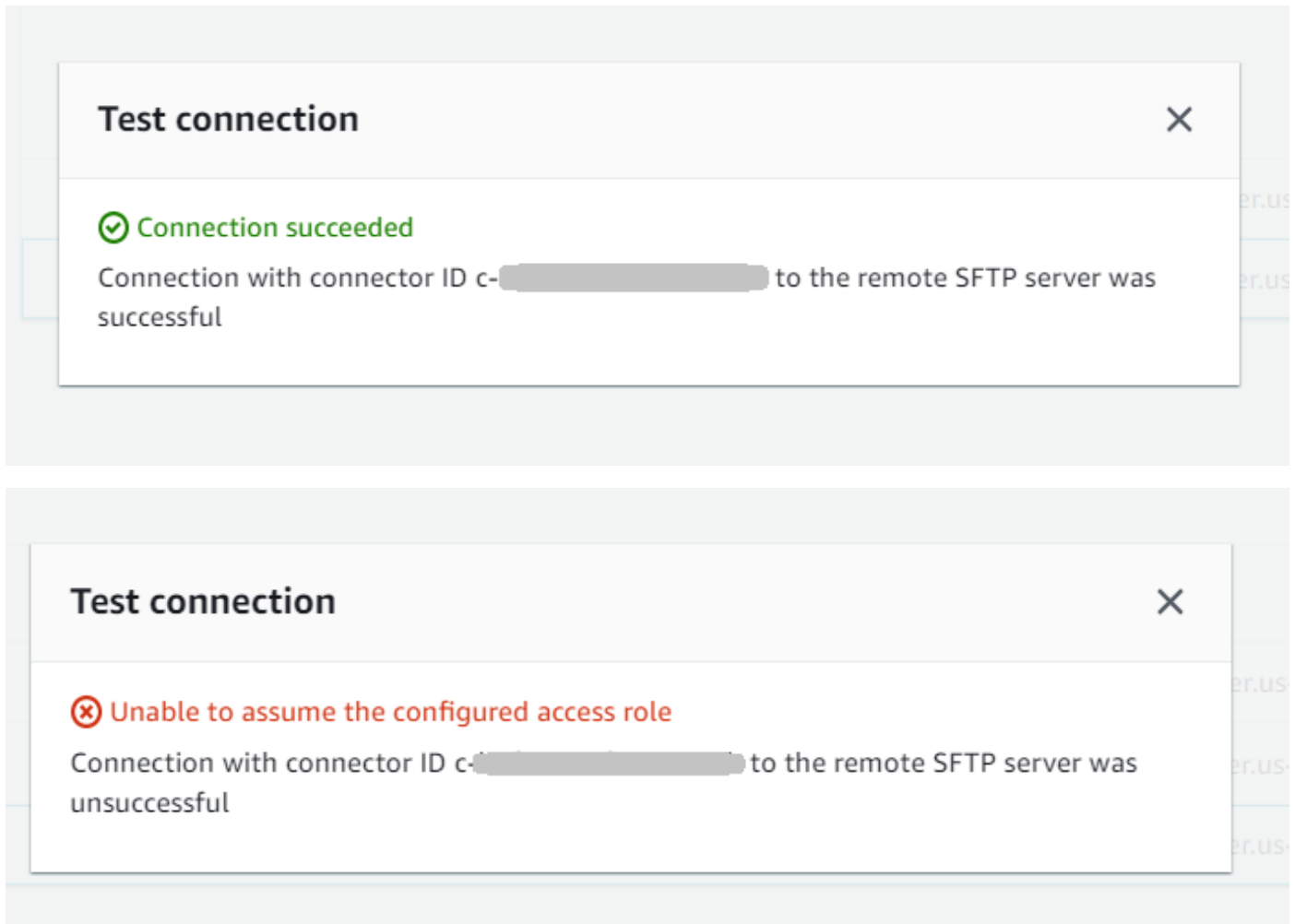
Test a connector using the console

To test an SFTP connector

- Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
- In the left navigation pane, choose **Connectors**, and select a connector.
- From the **Actions** menu, choose **Test connection**.



The system returns a message, indicating whether the test passes or fails. If the test fails, the system provides an error message based on the reason the test failed.



Test a connector using the CLI

To test a connector using the AWS Command Line Interface, run the following command at a command prompt (replace *connector-id* with your actual connector ID):

```
aws transfer test-connection --connector-id c-connector-id
```

If the test is successful, the following lines are returned:

```
{
  "Status": "OK",
  "StatusMessage": "Connection succeeded"
}
```

If the test is unsuccessful, you receive a descriptive error message, for example:

```
{
```

```
"Status": "ERROR",  
"StatusMessage": "Unable to assume the configured access role"  
}
```

Step 3: Send and retrieve files using the SFTP connector

For simplicity, we assume that you already have files in your Amazon S3 bucket.

Note

The tutorial is using Amazon S3 buckets for both source and destination storage locations. If your SFTP server doesn't use Amazon S3 storage, then wherever you see `sftp-server-storage-east` in the following commands, you can replace the path with a path to file locations accessible from your SFTP server.

- We send a file named `SEND-to-SERVER.txt` from Amazon S3 storage to the SFTP server.
- We retrieve a file named `RETRIEVE-to-S3.txt` from the SFTP server to Amazon S3 storage.

Note

In the following commands, replace *connector-id* with your connector ID.

First, we send a file from our Amazon S3 bucket to the remote SFTP server. From a command prompt, run the following command:

```
aws transfer start-file-transfer --connector-id c-connector-id --send-file-paths "/  
sftp-server-storage-east/SEND-to-SERVER.txt" /  
--remote-directory-path "/sftp-server-storage-east/incoming"
```

Your `sftp-server-storage-east` bucket should now look like this.

Amazon S3 > Buckets > sftp-server-storage-east > incoming/

incoming/


Copy S3 URI

Objects | Properties

Objects (1) Info

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 SEND-to-SERVER.txt	txt	December 18, 2023, 10:36:40 (UTC-05:00)	4.1 KB	Standard

If you don't see the file as expected, check your CloudWatch logs.

To check your CloudWatch logs

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>
2. Select **Log groups** from the left navigation menu.
3. Enter your connector ID in the search bar to find your logs.
4. Select the Log stream that is returned from the search.
5. Expand the most recent log entry.

If successful, the log entry looks like the following:

```
{
  "operation": "SEND",
  "timestamp": "2023-12-18T15:26:57.346283Z",
  "connector-id": "connector-id",
  "transfer-id": "transfer-id",
  "file-transfer-id": "transfer-id/file-transfer-id",
  "url": "sftp://server-id.server.transfer.us-east-1.amazonaws.com",
  "file-path": "/sftp-server-storage-east/SEND-to-SERVER.txt",
```



```

    "status-code": "COMPLETED",
    "start-time": "2023-12-18T15:26:56.915864Z",
    "end-time": "2023-12-18T15:26:57.298122Z",
    "account-id": "500655546075",
    "connector-arn": "arn:aws:transfer:us-east-1:500655546075:connector/connector-id",
    "remote-directory-path": "/sftp-server-storage-east/incoming"
  }

```

If the file transfer failed, the log entry contains an error message that specifies the issue. Common causes for errors are problems with the IAM permissions and incorrect file paths.

Next, we retrieve a file from the SFTP server into an Amazon S3 bucket. From a command prompt, run the following command:

```

aws transfer start-file-transfer --connector-id c-connector-id --retrieve-file-paths
"/sftp-server-storage-east/RETRIEVE-to-S3.txt" --local-directory-path "/sftp-server-
storage-east/incoming"

```

If the transfer succeeds, your Amazon S3 bucket contains the transferred file, as shown here.

The screenshot shows the Amazon S3 console interface. The breadcrumb navigation is: Amazon S3 > Buckets > sftp-server-storage-east > incoming/. The page title is 'incoming/'. There are tabs for 'Objects' and 'Properties'. Below the tabs, there is a section for 'Objects (1) Info' with a description and a 'Learn more' link. A toolbar contains buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', and 'Create folder'. There is also an 'Upload' button. A search bar is present with the placeholder text 'Find objects by prefix'. Below the search bar is a table with the following columns: Name, Type, Last modified, Size, and Storage class. The table contains one row for the file 'RETRIEVE-to-S3.txt' with a type of 'txt', last modified on December 18, 2023, at 10:26:58 (UTC-05:00), a size of 4.1 KB, and a storage class of 'Standard'.

Name	Type	Last modified	Size	Storage class
RETRIEVE-to-S3.txt	txt	December 18, 2023, 10:26:58 (UTC-05:00)	4.1 KB	Standard

If successful, the log entry looks like the following:

```
{
```

```
"operation": "RETRIEVE",
"timestamp": "2023-12-18T15:36:40.017800Z",
"connector-id": "c-connector-id",
"transfer-id": "transfer-id",
"file-transfer-id": "transfer-id/file-transfer-id",
"url": "sftp://s-server-id.server.transfer.us-east-1.amazonaws.com",
"file-path": "/sftp-server-storage-east/RETRIEVE-to-S3.txt",
"status-code": "COMPLETED",
"start-time": "2023-12-18T15:36:39.727626Z",
"end-time": "2023-12-18T15:36:39.895726Z",
"account-id": "500655546075",
"connector-arn": "arn:aws:transfer:us-east-1:500655546075:connector/c-connector-id",
"local-directory-path": "/sftp-server-storage-east/incoming"
}
```

Procedures to create a Transfer Family server to use as your remote SFTP server

Following, we outline the steps to create a Transfer Family server that serves as your remote SFTP server for this tutorial. Note the following:

- We use a Transfer Family server to represent a remote SFTP server. Typical SFTP connector users have their own remote SFTP server. See [Create a Transfer Family SFTP server and a user](#).
- Because we're using a Transfer Family server, we're also using a service-managed SFTP user. And, for simplicity, we combined the permissions that this user needs to access the Transfer Family server with permissions they need to use our connector. Again, most SFTP connector use cases have a separate SFTP user that is not associated with a Transfer Family server. See [Create a Transfer Family SFTP server and a user](#).
- For the tutorial, because we are using Amazon S3 storage for our remote SFTP server, we need to create a second bucket, **sftp-server-storage-east**, so that we can transfer files from one bucket to another.

Create a Transfer Family SFTP server and a user

Most users won't need to create a Transfer Family SFTP server and a user, as you already have an SFTP server with users, and you can use this server to transfer files to and from. However, for this tutorial, for simplicity, we are using a Transfer Family server to function as the remote SFTP server.

Follow the procedure described in [Create an SFTP-enabled server](#) to create a server, and [Step 3: Add a service managed user](#) to add a user. These are the user details that we are using for the tutorial:

- Create your service-managed user, `sftp-testuser`.
 - Set the home directory to `/sftp-server-storage-east/sftp-testuser`
 - When you create the user, you store a public key. Later, when you create the secret in Secrets Manager, you need to provide the corresponding private key.
- Role: `sftp-connector-role`. For the tutorial, we are using the same IAM role for both our SFTP user and for accessing the SFTP connector. When you create connectors for your organization, you might have separate user and access roles.
- Server host key: You need to use the server host key when you create the connector. You can retrieve this key by running `ssh-keyscan` for your server. For example, if your server ID is `s-1111aaaa2222bbbb3`, and its endpoint is in `us-east-1`, the following command retrieves the server host key:

```
ssh-keyscan s-1111aaaa2222bbbb3.server.transfer.us-east-1.amazonaws.com
```

Copy this text somewhere, as you need to paste it in the [Step 2: Create and test an SFTP connector](#) procedure.

Combined user and access role

For the tutorial, we are using a single, combined role. We use this role both for our SFTP user, as well as for access to the connector. The following example contains the details for this role, in case you want to perform the tasks in the tutorial.

The following example grants the necessary permissions to access our two buckets in Amazon S3, and the secret named `aws/transfer/sftp-connector1` stored in Secrets Manager. For the tutorial, this role is named `sftp-connector-role`.

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

```

        "s3:ListBucket",
        "s3:GetBucketLocation"
    ],
    "Effect": "Allow",
    "Resource": [
        "arn:aws:s3:::sftp-server-storage-east",
        "arn:aws:s3:::sftp-server-storage-east"
    ]
},
{
    "Sid": "HomeDirObjectAccess",
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:GetObjectVersion",
        "s3:GetObjectACL",
        "s3:PutObjectACL"
    ],
    "Resource": [
        "arn:aws:s3:::sftp-server-storage-east/*",
        "arn:aws:s3:::sftp-server-storage-east/*"
    ]
},
{
    "Sid": "GetConnectorSecretValue",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue"
    ],
    "Resource": "arn:aws:secretsmanager:us-east-1:500655546075:secret:aws/transfer/sftp-connector1-6RandomCharacters"
}
]
}

```

For complete details about creating roles for Transfer Family, follow the procedure described in [Create a user role](#) to create a role.

Setting up an Amazon API Gateway method as a custom identity provider

This tutorial illustrates how to set up an Amazon API Gateway method and use it as a custom identity provider to upload files to an AWS Transfer Family server. This tutorial uses the [Basic stack template](#), and other basic functionality as an example only.

Topics

- [Prerequisites](#)
- [Step 1: Create a CloudFormation stack](#)
- [Step 2: Check the API Gateway method configuration for your server](#)
- [Step 3: View the Transfer Family server details](#)
- [Step 4: Test that your user can connect to the server](#)
- [Step 5: Test the SFTP connection and file transfer](#)
- [Step 6: Limit access to the bucket](#)
- [Update Lambda if using Amazon EFS](#)

Prerequisites

Before you create the Transfer Family resources in AWS CloudFormation, create your storage and your user role.

To specify storage and create a user role

1. Depending on which storage you are using, see the following documentation:
 - To create an Amazon S3 bucket, see [How do I create an S3 bucket?](#) in the *Amazon Simple Storage Service User Guide*.
 - To create an Amazon EFS file system, see [Configure an Amazon EFS file system](#).
2. To create a user role, see [Create an IAM role and policy](#)

You enter the details for your storage and your user role when you create your AWS CloudFormation stack in the next section.

Step 1: Create a CloudFormation stack

To create an AWS CloudFormation stack from the provided template

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. Select **Create stack**, and choose **With new resources (standard)**.
3. In the **Prerequisite - Prepare template** pane, choose **Template is ready**.
4. Copy this link, [Basic stack template](#), and paste it into the **Amazon S3 URL** field.
5. Click **Next**.
6. Specify parameters, including a name for your stack. Be sure to do the following:
 - Replace the default values for **UserName** and **UserPassword**.
 - For **UserHomeDirectory**, enter the details for the storage (either an Amazon S3 bucket or an Amazon EFS filesystem) that you created earlier.
 - Replace the default **UserRoleArn** with the user role that you created earlier. The AWS Identity and Access Management (IAM) role must have the appropriate permissions. For an example IAM role and bucket policy, see [Step 6: Limit access to the bucket](#).
 - If you want to authenticate using a public key instead of a password, enter your public key in the **UserPublicKey1** field. The first time that you connect to the server using SFTP, you then provide the private key instead of a password.
7. Choose **Next**, and then choose **Next** again on the **Configure stack options** page.
8. Review the details for the stack that you are creating, and then choose **Create stack**.

Note

At the bottom of the page, under **Capabilities**, you must acknowledge that AWS CloudFormation might create IAM resources.

Step 2: Check the API Gateway method configuration for your server

Note

To improve security, you can configure a web application firewall. AWS WAF is a web application firewall that lets you monitor the HTTP and HTTPS requests that are forwarded to an Amazon API Gateway. For details, see [Add a web application firewall](#).

To check the API Gateway method configuration for your server and deploy it

1. Open the API Gateway console at <https://console.aws.amazon.com/apigateway/>.
2. Choose the **Transfer Custom Identity Provider basic template API** that the AWS CloudFormation template generated.
3. In the **Resources** pane, choose **GET**, and then choose **Method Request**.
4. For **Actions**, choose **Deploy API**. For **Deployment stage**, choose **prod**, and then choose **Deploy**.

After the API Gateway method is successfully deployed, view its performance in the **Stage Editor** section.

Note

Copy the **Invoke URL** address that appears at the top of the page. You will need it for the next step.

Step 3: View the Transfer Family server details

When you use the template to create an AWS CloudFormation stack, a Transfer Family server is automatically created.

To view your Transfer Family server details

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. Choose the stack that you created.
3. Choose the **Resources** tab.

Resources (18)			
<input type="text" value="Search resources"/>			
Logical ID	Physical ID	Type	
ApiCloudWatchLogsRole	-ApiCloudWatchLogsRole-	AWS::IAM::Role	
ApiDeployment202008		AWS::ApiGateway::Deployment	
ApiLoggingAccount		AWS::ApiGateway::Account	
ApiStage	prod	AWS::ApiGateway::Stage	
CloudWatchLoggingRole	-CloudWatchLoggingRole-	AWS::IAM::Role	
CustomIdentityProviderApi		AWS::ApiGateway::RestApi	
GetUserConfigLambda	-GetUserConfigLambda-	AWS::Lambda::Function	
GetUserConfigLambdaPermission	-GetUserConfigLambdaPermission-	AWS::Lambda::Permission	
GetUserConfigRequest		AWS::ApiGateway::Method	
GetUserConfigResource		AWS::ApiGateway::Resource	
GetUserConfigResponseModel	UserConfigResponseModel	AWS::ApiGateway::Model	
LambdaExecutionRole	-LambdaExecutionRole-	AWS::IAM::Role	
ServerIdResource		AWS::ApiGateway::Resource	
ServersResource		AWS::ApiGateway::Resource	
TransferIdentityProviderRole	-TransferIdentityProviderRole-	AWS::IAM::Role	
TransferServer	arn:aws:transfer:us-east-2:::server/s-	AWS::Transfer::Server	
UserNameResource		AWS::ApiGateway::Resource	
UsersResource		AWS::ApiGateway::Resource	

The server ARN is shown in the **Physical ID** column for the **TransferServer** row. The server ID is contained in the ARN, for example **s-11112222333344445**.

- Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>, and on the **Servers** page, choose the new server.

The server ID matches the ID displayed for the **TransferServer** resource in AWS CloudFormation.

Step 4: Test that your user can connect to the server

To test that your user can connect to the server, using the Transfer Family console

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. On the **Servers** page, choose your new server, choose **Actions**, and then choose **Test**.
3. Enter the text for your sign-in credentials into the **Username** field, and into the **Password** field. These are the values that you set when you deployed the AWS CloudFormation stack.
4. For **Server Protocol**, select **SFTP**, and for **Source IP**, enter **127.0.0.1**.
5. Choose **Test**.

If user authentication succeeds, the test returns a `StatusCode: 200` HTML response and a JSON object containing the details of the user's roles and permissions. For example:

```
{
  "Response": "{\"Role\": \"arn:aws:iam::123456789012:role/my-user-role\",
  \"HomeDirectory\": \"/${transfer:HomeBucket}/\"}\",
  "StatusCode": 200,
  "Message": "",
  "Url": "https://1a2b3c4d5e.execute-api.us-east-2.amazonaws.com/prod/servers/s-1234abcd5678efgh0/users/myuser/config"
}
```

If the test fails, add one of the API Gateway AWS managed policies to the role that you are using for your API.

Step 5: Test the SFTP connection and file transfer

To test the SFTP connection

1. On a Linux or macOS device, open a command terminal.
2. Enter one of the following commands, depending on whether you are using a password or a key pair for authentication.
 - If you are using a password, enter this command:

```
sftp -o PubkeyAuthentication=no myuser@server-ID.server.transfer.region-code.amazonaws.com
```

When prompted, enter your password.

- If you are using a key pair, enter this command:

```
sftp -i private-key-file myuser@server-ID.server.transfer.region-code.amazonaws.com
```

Note

For these sftp commands, insert the code for the AWS Region where your Transfer Family server is located. For example, if your server is in US East (Ohio), enter **us-east-2**.

3. At the sftp> prompt, make sure that you can upload (put), download (get), and view directories and files (pwd and ls).

Step 6: Limit access to the bucket

You can limit who can access a specific Amazon S3 bucket. The following example shows the settings to use in your CloudFormation stack and in the policy that you select for your user.

In this example, we set the following parameters for the AWS CloudFormation stack:

- **CreateServer:** true
- **UserHomeDirectory:** /DOC-EXAMPLE-BUCKET1
- **UserName:** myuser
- **UserPassword:** MySuperSecretPassword

Important

This is an example password. When you configure your API Gateway method, make sure that you enter a strong password.

- **UserPublicKey1:** *your-public-key*
- **UserRoleArn:** arn:aws:iam::*role-id*:role/myuser-api-gateway-role

The **UserPublicKey1** is a public key that you have generated as part of a public/private key pair.

The *role-id* is unique to the user role that you create. The policy attached to the `myuser-api-gateway-role` is the following:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObjectAcl",
        "s3:GetObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:PutObjectAcl",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
    }
  ]
}
```

To connect to the server using SFTP, enter one of the following commands at the prompt.

- If you are using a password to authenticate, run the following command:

```
sftp -o PubkeyAuthentication=no myuser@transfer-server-ID.server.transfer.region-id.amazonaws.com
```

When prompted, enter your password.

- If you are using a key pair to authenticate, run the following command:

```
sftp -i private-key-file myuser@transfer-server-ID.server.transfer.region-id.amazonaws.com
```

Note

For these `sftp` commands, use the ID for the AWS Region where your Transfer Family server is located. For example, if your server is in US East (Ohio), use `us-east-2`.

At the `sftp` prompt, you are directed to your home directory, which you can view by running the `pwd` command. For example:

```
sftp> pwd
Remote working directory: /DOC-EXAMPLE-BUCKET1
```

The user cannot view any directories above the home directory. For example:

```
sftp> pwd
Remote working directory: /DOC-EXAMPLE-BUCKET1
sftp> cd ..
sftp> ls
Couldn't read directory: Permission denied
```

Update Lambda if using Amazon EFS

If you selected Amazon EFS as the storage option for your Transfer Family server, you need to edit the lambda function for your stack.

To add a Posix profile to your Lambda function

1. Open the Lambda console at <https://console.aws.amazon.com/lambda/>.
2. Select the Lambda function that you created earlier. The Lambda function has the format of ***stack-name-GetUserConfigLambda-lambda-identifier***, where *stack-name* is the CloudFormation stack name and *lambda-identifier* is the identifier for the function.
3. In the **Code** tab, select **index.js** to display the code for the function.
4. In the response, add the following line between Policy and HomeDirectory:

```
PosixProfile: {"Uid": uid-value, "Gid": gid-value},
```

Where the *uid-value* and *gid-value* are integers, 0 or greater, that represent the User ID and Group ID respectively.

For example, after you add the Posix profile, the response field might look like the following:

```
response = {
  Role: 'arn:aws:iam::123456789012:role/api-gateway-transfer-efs-role', // The
  user will be authenticated if and only if the Role field is not blank
  Policy: '', // Optional JSON blob to further restrict this user's permissions
  PosixProfile: {"Gid": 65534, "Uid": 65534},
  HomeDirectory: '/fs-fab2c234' // Not required, defaults to '/'
};
```

Setting up an AS2 configuration

This tutorial walks through how to set up an Applicability Statement 2 (AS2) configuration with AWS Transfer Family. After you complete the steps described here, you will have an AS2-enabled server that's ready for accepting AS2 messages from a sample trading partner. You will also have a connector that can be used to send AS2 messages to the sample trading partner.

Note

Some portions of the example setup use the AWS Command Line Interface (AWS CLI). If you haven't already installed the AWS CLI, see [Installing or updating the latest version of the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

1. Create certificates for yourself and your trading partner. If you have existing certificates that you can use, you can skip this section.

This process is described in [Step 1: Create certificates for AS2](#).

2. Create an AWS Transfer Family server that uses the AS2 protocol. Optionally, you can add an Elastic IP address to the server to make it internet-facing.

This process is described in [Step 2: Create a Transfer Family server that uses the AS2 protocol](#).

Note

You must create a Transfer Family server for inbound transfers only. If you're performing only outbound transfers, you don't need a Transfer Family server.

3. Import the certificates that you created in step 1.

This process is described in [Step 3: Import certificates as Transfer Family certificate resources](#).

4. To set up your trading partners, create a local profile and a partner profile.

This process is described in [Step 4: Create profiles for you and your trading partner](#).

5. Create an agreement between you and your trading partner.

This process is described in [Step 5: Create an agreement between you and your partner](#).

Note

You must create an agreement for inbound transfers only. If you're performing only outbound transfers, you don't need an agreement.

6. Create a connector between you and your trading partner.

This process is described in [Step 6: Create a connector between you and your partner](#).

Note

You must create a connector for outbound transfers only. If you're performing only inbound transfers, you don't need a connector.

7. Test an AS2 file exchange.

This process is described in [Step 7: Test exchanging files over AS2 by using Transfer Family](#).

After you complete these steps, you can do the following:

- Send files to a remote AS2-enabled partner server with the Transfer Family `start-file-transfer` AWS Command Line Interface (AWS CLI) command.

- Receive files from a remote AS2-enabled partner server on port 5080 through your virtual private cloud (VPC) endpoint.

Step 1: Create certificates for AS2

Both parties in an AS2 exchange need X.509 certificates. You can create these certificates in any way that you like. This topic describes how to use [OpenSSL](#) from the command line to create a root certificate, and then sign subordinate certificates. Both parties must generate their own certificates.

Note

The key length for AS2 certificates must be at least 2048 bits, and at most 4096.

To transfer files with a partner, take note of the following:

- You can attach certificates to profiles. The certificates contain public or private keys.
- Your trading partner sends you their public keys, and you send them yours.
- Your trading partner encrypts messages with your public key and signs them with their private key. Conversely, you encrypt messages with your partner's public key and sign them with your private key.

Note

If you prefer to manage keys with a GUI, [Portecle](#) is one option that you can use.

To generate example certificates

Important

Do not send your partner your private keys. In this example, you generate a set of self-signed public and private keys for one party. If you are going to act as both trading partners for testing purposes, you can repeat these instructions to generate two sets of

keys: one for each trading partner. In this case, you do not need to generate two root certificate authorities (CAs).

1. Run the following command to generate an RSA private key with a 2048-bit-long modulus.

```
/usr/bin/openssl genrsa -out root-ca-key.pem 2048
```

2. Run the following command to create a self-signed certificate with your `root-ca-key.pem` file.

```
/usr/bin/openssl req \
-x509 -new -nodes -sha256 \
-days 1825 \
-subj "/C=US/ST=MA/L=Boston/O=TransferFamilyCustomer/OU=IT-dept/CN=ROOTCA" \
-key root-ca-key.pem \
-out root-ca.pem
```

The `-subj` argument consists of the following values.

	Name	Description
C	Country code	A two-letter code for the country in which your organization is located.
ST	State, region, or province	The state, region, or province in which your organization is located. (In this case, <i>region</i> does not refer to your AWS Region.)
L	Locality name	The city in which your organization is located.
O	Organization name	The full legal name of your organization, including

	Name	Description
		suffixes, such as LLC, Corp, and so on.
OU	Organizational unit name	The division in your organization that deals with this certificate.
CN	Common name or fully qualified domain name (FQDN)	In this case, we're creating a root certificate, so the value is ROOTCA. In these examples, we are using CN to describe the purpose of the certificate.

3. Create a signing key and an encryption key for your local profile.

```
/usr/bin/openssl genrsa -out signing-key.pem 2048
/usr/bin/openssl genrsa -out encryption-key.pem 2048
```

Note

Some AS2-enabled servers, such as OpenAS2, require that you use the same certificate for both signing and encryption. In this case, you can import the same private key and certificate for both purposes. To do so, run this command instead of the two previous commands:

```
/usr/bin/openssl genrsa -out signing-and-encryption-key.pem 2048
```

4. Run the following commands to create Certificate Signing Requests (CSRs) for the root key to sign.

```
/usr/bin/openssl req -new -key signing-key.pem -subj \
"/C=US/ST=MA/L=Boston/O=TransferFamilyCustomer/OU=IT-dept/CN=Signer" -out signing-
key-csr.pem
```

```
/usr/bin/openssl req -new -key encryption-key.pem -subj \
```

```
"/C=US/ST=MA/L=Boston/O=TransferFamilyCustomer/OU=IT-dept/CN=Encrypter" -out
encryption-key-csr.pem
```

5. Next, you must create a `signing-cert.conf` file and an `encryption-cert.conf` file.

- Use a text editor to create the `signing-cert.conf` file with the following contents:

```
authorityKeyIdentifier=keyid,issuer
keyUsage = digitalSignature, nonRepudiation
```

- Use a text editor to create the `encryption-cert.conf` file with the following contents:

```
authorityKeyIdentifier=keyid,issuer
keyUsage = dataEncipherment
```

6. Finally, you create the signed certificates by running the following commands.

```
/usr/bin/openssl x509 -req -sha256 -CAcreateserial -days 1825 -in signing-key-
csr.pem -out signing-cert.pem -CA \
root-ca.pem -CAkey root-ca-key.pem -extfile signing-cert.conf
```

```
/usr/bin/openssl x509 -req -sha256 -CAcreateserial -days 1825 -in encryption-key-
csr.pem -out encryption-cert.pem \
-CA root-ca.pem -CAkey root-ca-key.pem -extfile encryption-cert.conf
```

Step 2: Create a Transfer Family server that uses the AS2 protocol

This procedure explains how to create an AS2-enabled server by using the Transfer Family AWS CLI.

Note

Many of the example steps use commands that load parameters from a file. For more details about using files to load parameters, see [How to load parameters from a file](#).

If you want to use the console instead, see [Create an AS2 server using the Transfer Family console](#).

Similar to how you create an SFTP or FTPS AWS Transfer Family server, you create an AS2-enabled server by using the `--protocols AS2` parameter of the `create-server` AWS CLI command.

Currently, Transfer Family supports only VPC endpoint types and Amazon S3 storage with the AS2 protocol.

When you create your AS2-enabled server for Transfer Family by using the `create-server` command, a VPC endpoint is automatically created for you. This endpoint exposes TCP port 5080 so that it can accept AS2 messages.

If you want to expose your VPC endpoint publicly to the internet, you can associate Elastic IP addresses with your VPC endpoint.

To use these instructions, you need the following:

- The ID of your VPC (for example, **vpc-abcdef01**).
- The IDs of your VPC subnets (for example, **subnet-abcdef01**, **subnet-subnet-abcdef01**, **subnet-021345ab**).
- One or more IDs of the security groups that allow incoming traffic on TCP port 5080 from your trading partners (for example, **sg-1234567890abcdef0** and **sg-abcdef01234567890**).
- (Optional) The Elastic IP addresses that you want to associate with your VPC endpoint.
- If your trading partner is not connected to your VPC through a VPN, you need an internet gateway. For more information, see [Connect to the internet using an internet gateway](#) in the *Amazon VPC User Guide*.

To create an AS2-enabled server

1. Run the following command. Replace each *user input placeholder* with your own information.

```
aws transfer create-server --endpoint-type VPC \  
--endpoint-details VpcId=vpc-abcdef01,SubnetIds=subnet-abcdef01,subnet-  
abcdef01,subnet-  
021345ab,SecurityGroupIds=sg-abcdef01234567890,sg-1234567890abcdef0 --protocols AS2 \  
\   
--protocol-details As2Transports=HTTP
```

2. (Optional) You can make the VPC endpoint public. You can attach Elastic IP addresses to a Transfer Family server only through an `update-server` operation. The following commands stop the server, update it with Elastic IP addresses, and then start it again.

```
aws transfer stop-server --server-id your-server-id
```

```
aws transfer update-server --server-id your-server-id --endpoint-details \  
AddressAllocationIds=eipalloc-abcdef01234567890,eipalloc-  
1234567890abcdef0,eipalloc-abcd012345ccccccc
```

```
aws transfer start-server --server-id your-server-id
```

This `start-server` command automatically creates a DNS record for you that contains the public IP address for your server. To give your trading partner access to the server, you provide them with the following information. In this case, *your-region* refers to your AWS Region.

s-your-server-id.server.transfer.*your-region*.amazonaws.com

The full URL that you provide to your trading partner is as follows:

`http://s-your-server-id.server.transfer.your-region.amazonaws.com:5080`

3. To test whether your AS2-enabled server is accessible, use the following commands. Make sure that your server can be accessed either through your VPC endpoint's private DNS address, or through your public endpoint (if you associated an Elastic IP address with your endpoint).

If your server is configured correctly, the connection will succeed. However, you will receive an HTTP status code 400 (Bad Request) response because you aren't sending a valid AS2 message.

- For a public endpoint (if you associated an Elastic IP address in the previous step), run the following command, substituting your server ID and Region.

```
curl -vv -X POST http://s-your-server-id.transfer.your-region.amazonaws.com:5080
```

- If you are connecting within your VPC, look up your VPC endpoint's private DNS name by running the following commands.

```
aws transfer describe-server --server-id s-your-server-id
```

This `describe-server` command returns your VPC endpoint ID in the `VpcEndpointId` parameter. Use this value to run the following command.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-your-vpc-endpoint-id
```

This `describe-vpc-endpoints` command returns a `DNSEntries` array, with several `DnsName` parameters. Use the Regional DNS name (the one that does not include the Availability Zone) in the following command.

```
curl -vv -X POST http://vpce-your-vpce.vpce-svc-your-vpce-svc.your-region.vpce.amazonaws.com:5080
```

For example, the following command shows sample values for the placeholders in the previous command.

```
curl -vv -X POST http://vpce-0123456789abcdefg-fghij123.vpce-svc-11111aaaa2222bbbb.us-east-1.vpce.amazonaws.com:5080
```

4. (Optional) Configure a logging role. Transfer Family logs the status of messages sent and received in a structured JSON format to Amazon CloudWatch logs. To provide Transfer Family with access to the CloudWatch logs in your account, you must configure a logging role on your server.

Create an AWS Identity and Access Management (IAM) role that trusts `transfer.amazonaws.com`, and attach the `AWSTransferLoggingAccess` managed policy. For details, see [Create an IAM role and policy](#). Note the Amazon Resource Name (ARN) of the IAM role that you just created, and associate it with the server by running the following `update-server` command:

```
aws transfer update-server --server-id your-server-id --logging-role arn:aws:iam::your-account-id:role/logging-role-name
```

Note

Even though the logging role is optional, we highly recommend setting it up so that you can see the status of your messages and troubleshoot configuration issues.

Step 3: Import certificates as Transfer Family certificate resources

This procedure explains how to import certificates by using the AWS CLI. If you want to use the Transfer Family console instead, see [the section called "Import AS2 certificates"](#).

To import the signing and encryption certificates that you created in step 1, run the following `import-certificate` commands. If you're using the same certificate for encryption and signing, import the same certificate twice (once with the `SIGNING` usage and again with the `ENCRYPTION` usage).

```
aws transfer import-certificate --usage SIGNING --certificate file://signing-cert.pem \  
    --private-key file://signing-key.pem --certificate-chain file://root-ca.pem
```

This command returns your signing `CertificateId`. In the next section, this certificate ID is referred to as *my-signing-cert-id*.

```
aws transfer import-certificate --usage ENCRYPTION --certificate file://encryption-  
cert.pem \  
    --private-key file://encryption-key.pem --certificate-chain file://root-  
ca.pem
```

This command returns your encryption `CertificateId`. In the next section, this certificate ID is referred to as *my-encrypt-cert-id*.

Next, import your partner's encryption and signing certificates by running the following commands.

```
aws transfer import-certificate --usage ENCRYPTION --certificate file://partner-  
encryption-cert.pem \  
    --certificate-chain file://partner-root-ca.pem
```

This command returns your partner's encryption `CertificateId`. In the next section, this certificate ID is referred to as *partner-encrypt-cert-id*.

```
aws transfer import-certificate --usage SIGNING --certificate file://partner-signing-  
cert.pem \  
    --certificate-chain file://partner-root-ca.pem
```

This command returns your partner's signing `CertificateId`. In the next section, this certificate ID is referred to as *partner-signing-cert-id*.

Step 4: Create profiles for you and your trading partner

This procedure explains how to create AS2 profiles by using AWS CLI. If you want to use the Transfer Family console instead, see [the section called “Create AS2 profiles”](#).

Create your local AS2 profile by running the following command. This command references the certificates that contain your public and private keys.

```
aws transfer create-profile --as2-id MYCORP --profile-type LOCAL --certificate-ids \  
my-signing-cert-id my-encrypt-cert-id
```

This command returns your profile ID. In the next section, this ID is referred to as *my-profile-id*.

Now create the partner profile by running the following command. This command uses only your partner's public key certificates. To use this command, replace the *user input placeholders* with your own information; for example, your partner's AS2 name and certificate IDs.

```
aws transfer create-profile --as2-id PARTNER-COMPANY --profile-type PARTNER --  
certificate-ids \  
partner-signing-cert-id partner-encrypt-cert-id
```

This command returns your partner's profile ID. In the next section, this ID is referred to as *partner-profile-id*.

Note

In the previous commands, replace *MYCORP* with the name of your organization, and *PARTNER-COMPANY* with the name of your trading partner's organization.

Step 5: Create an agreement between you and your partner

This procedure explains how to create AS2 agreements by using the AWS CLI. If you want to use the Transfer Family console instead, see [the section called “Create AS2 agreements”](#).

Agreements bring together the two profiles (local and partner), their certificates, and a server configuration that allows inbound AS2 transfers between two parties. You can list your items by running the following commands.

```
aws transfer list-profiles --profile-type LOCAL
aws transfer list-profiles --profile-type PARTNER
aws transfer list-servers
```

This step requires an Amazon S3 bucket and IAM role with read/write access to and from the bucket. The instructions for creating this role are the same as for the Transfer Family SFTP, FTP, and FTPS protocols and are available in [Create an IAM role and policy](#).

To create an agreement, you need the following items:

- The Amazon S3 bucket name (and object prefix, if specified)
- The ARN of the IAM role with access to the bucket
- Your Transfer Family server ID
- Your profile ID and your partner's profile ID

Create the agreement by running the following command.

```
aws transfer create-agreement --description "ExampleAgreementName" --server-id your-server-id \
--local-profile-id your-profile-id --partner-profile-id your-partner-profile-id --base-
directory /DOC-EXAMPLE-DESTINATION-BUCKET/AS2-inbox \
--access-role arn:aws:iam::111111111111:role/TransferAS2AccessRole
```

If successful, this command returns the ID for the agreement. You can then view the details of the agreement with the following command.

```
aws transfer describe-agreement --agreement-id agreement-id --server-id your-server-id
```

Step 6: Create a connector between you and your partner

This procedure explains how to create AS2 connectors by using the AWS CLI. If you want to use the Transfer Family console instead, see [the section called "Configure AS2 connectors"](#).

You can use the `StartFileTransfer` API operation to send files that are stored in Amazon S3 to your trading partner's AS2 endpoint by using a connector. You can find the profiles that you created earlier by running the following command.

```
aws transfer list-profiles
```


When you create the connector, you must provide your partner's AS2 server URL. Copy the following text to a file named `testAS2Config.json`.

```
{
  "Compression": "ZLIB",
  "EncryptionAlgorithm": "AES256_CBC",
  "LocalProfileId": "your-profile-id",
  "MdnResponse": "SYNC",
  "MdnSigningAlgorithm": "DEFAULT",
  "MessageSubject": "Your Message Subject",
  "PartnerProfileId": "partner-profile-id",
  "SigningAlgorithm": "SHA256"
}
```

Note

For `EncryptionAlgorithm`, do not specify the `DES_EDE3_CBC` algorithm unless you must support a legacy client that requires it, as it is a weak encryption algorithm.

Then run the following command to create the connector.

```
aws transfer create-connector --url "http://partner-as2-server-url" \
--access-role your-IAM-role-for-bucket-access \
--logging-role arn:aws:iam::your-account-id:role/service-role/AWSTransferLoggingAccess
\
--as2-config file:///path/to/testAS2Config.json
```

Step 7: Test exchanging files over AS2 by using Transfer Family

Receive a file from your trading partner

If you associated a public Elastic IP address with your VPC endpoint, Transfer Family automatically created a DNS name that contains your public IP address. The subdomain is your AWS Transfer Family server ID (of the format `s-1234567890abcdef0`). Provide your server URL to your trading partner in the following format.

```
http://s-1234567890abcdef0.server.transfer.us-east-1.amazonaws.com:5080
```

If you didn't associate a public Elastic IP address with your VPC endpoint, look up the hostname of the VPC endpoint that can accept AS2 messages over HTTP POST from your trading partners on port 5080. To retrieve the VPC endpoint details, use the following command.

```
aws transfer describe-server --server-id s-1234567890abcdef0
```

For example, assume the preceding command returns a VPC endpoint ID of `vpce-1234abcd5678efghi`. Then, you would use the following command to retrieve the DNS names.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-1234abcd5678efghi
```

This command returns all the details for the VPC endpoint that you need to run the following command.

The DNS name is listed in the `DnsEntries` array. Your trading partner must be within your VPC to access your VPC endpoint (for example through AWS PrivateLink or a VPN). Provide your VPC endpoint URL to your partner in the following format.

```
http://vpce-your-vpce-id.vpce-svc-your-vpce-svc-id.your-region.vpce.amazonaws.com:5080
```

For example, the following URL shows sample values for the placeholders in the previous commands.

```
http://vpce-0123456789abcdefg-fghij123.vpce-svc-11111aaaa2222bbbb.us-east-1.vpce.amazonaws.com:5080
```

In this example, successful transfers are stored at the location that's specified in the `base-directory` parameter that you specified in [Step 5: Create an agreement between you and your partner](#). If we successfully receive files named `myfile1.txt` and `myfile2.txt`, the files are stored as `/path-defined-in-the-agreement/processed/original_filename.messageId.original_extension`. Here, the files are stored as `/DOC-EXAMPLE-DESTINATION-BUCKET/AS2-inbox/processed/myfile1.messageId.txt` and `/DOC-EXAMPLE-DESTINATION-BUCKET/AS2-inbox/processed/myfile2.messageId.txt`.

If you configured a logging role when you created your Transfer Family server, you can also check your CloudWatch logs for the status of AS2 messages.

Send a file to your trading partner

You can use Transfer Family to send AS2 messages by referencing the connector ID and the paths to the files, as illustrated in the following `start-file-transfer` AWS Command Line Interface (AWS CLI) command:

```
aws transfer start-file-transfer --connector-id c-1234567890abcdef0 \  
--send-file-paths "/DOC-EXAMPLE-SOURCE-BUCKET/myfile1.txt" "/DOC-EXAMPLE-SOURCE-BUCKET/  
myfile2.txt"
```

To get the details for your connectors, run the following command:

```
aws transfer list-connectors
```

The `list-connectors` command returns the connector IDs, URLs, and Amazon Resource Names (ARNs) for your connectors.

To return the properties of a particular connector, run the following command with the ID that you want to use:

```
aws transfer describe-connector --connector-id your-connector-id
```

The `describe-connector` command returns all of the properties for the connector, including its URL, roles, profiles, Message Disposition Notices (MDNs), tags, and monitoring metrics.

You can confirm that the partner successfully received the files by viewing the JSON and MDN files. These files are named according to the conventions described in [File names and locations](#). If you configured a logging role when you created the connector, you can also check your CloudWatch logs for the status of AS2 messages.

Configuring an SFTP, FTPS, or FTP server endpoint

This topic provides details for creating and using AWS Transfer Family server endpoints that use one or more of the SFTP, FTPS, and FTP protocols.

Topics

- [Identity provider options](#)
- [AWS Transfer Family endpoint type matrix](#)
- [Configuring an SFTP, FTPS, or FTP server endpoint](#)
- [Transferring files over a server endpoint using a client](#)
- [Managing users for server endpoints](#)
- [Using logical directories to simplify your Transfer Family directory structures](#)

Identity provider options

AWS Transfer Family provides several methods for authenticating and managing users. The following table compares the available identity providers that you can use with Transfer Family.

Action	AWS Transfer Family service managed	AWS Managed Microsoft AD	Amazon API Gateway	AWS Lambda
Supported protocols	SFTP	SFTP, FTPS, FTP	SFTP, FTPS, FTP	SFTP, FTPS, FTP
Key-based authentication	Yes	No	Yes	Yes
Password authentication	No	Yes	Yes	Yes
AWS Identity and Access Management (IAM) and POSIX	Yes	Yes	Yes	Yes

Action	AWS Transfer Family service managed	AWS Managed Microsoft AD	Amazon API Gateway	AWS Lambda
Logical home directory	Yes	Yes	Yes	Yes
Parameterized access (username-based)	Yes	Yes	Yes	Yes
Ad hoc access structure	Yes	No	Yes	Yes
AWS WAF	No	No	Yes	No

Notes:

- IAM is used to control access for Amazon S3 backing storage, and POSIX is used for Amazon EFS.
- *Ad hoc* refers to the ability to send the user profile at runtime. For example, you can land users in their home directories by passing the username as a variable.
- For details about AWS WAF, see [Add a web application firewall](#).
- There is a blog post that describes using a Lambda function integrated with Microsoft Azure AD as your Transfer Family identity provider. For details, see [Authenticating to AWS Transfer Family with Azure Active Directory and AWS Lambda](#).
- We provide several AWS CloudFormation templates to help you quickly deploy a Transfer Family server that uses a custom identity provider. For details, see [Lambda function templates](#).

In the following procedures, you can create an SFTP-enabled server, FTPS-enabled server, FTP-enabled server, or AS2-enabled server.

Next step

- [Create an SFTP-enabled server](#)
- [Create an FTPS-enabled server](#)

- [Create an FTP-enabled server](#)
- [Configuring AS2](#)

AWS Transfer Family endpoint type matrix

When you create a Transfer Family server, you choose the type of endpoint to use. The following table describes characteristics for each type of endpoint.

Endpoint type matrix

Characteristic	Public	VPC - Internet	VPC - Internal	VPC_Endpoint (deprecated)
Supported protocols	SFTP	SFTP, FTPS, AS2	SFTP, FTP, FTPS, AS2	SFTP
Access	From over the internet. This endpoint type doesn't require any special configuration in your VPC.	Over the internet and from within VPC and VPC-connected environments, such as an on-premises data center over AWS Direct Connect or VPN.	From within VPC and VPC-connected environments, such as an on-premises data center over AWS Direct Connect or VPN.	From within VPC and VPC-connected environments, such as an on-premises data center over AWS Direct Connect or VPN.
Static IP address	You can't attach a static IP address. AWS provides IP addresses that are subject to change.	You can attach Elastic IP addresses to the endpoint. These can be AWS-owned IP addresses or your own IP addresses (Bring your own IP addresses)	Private IP addresses attached to the endpoint don't change.	Private IP addresses attached to the endpoint don't change.

Characteristic	Public	VPC - Internet	VPC - Internal	VPC_Endpoint (deprecated)
		<p>). Elastic IP addresses attached to the endpoint don't change.</p> <p>Private IP addresses attached to the server also don't change.</p>		

Characteristic	Public	VPC - Internet	VPC - Internal	VPC_Endpoint (deprecated)
Source IP allow list	<p>This endpoint type does not support allow lists by source IP addresses.</p> <p>The endpoint is publicly accessible and listens for traffic over port 22.</p> <div data-bbox="402 814 649 1696" style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>For VPC-hosted endpoints, SFTP Transfer Family servers can operate over port 22 (the default), port 2222, or port 22000.</p> </div>	<p>To allow access by source IP address, you can use security groups attached to the server endpoints and network ACLs attached to the subnet that the endpoint is in.</p>	<p>To allow access by source IP address, you can use security groups attached to the server endpoints and network access control lists (network ACLs) attached to the subnet that the endpoint is in.</p>	<p>To allow access by source IP address, you can use security groups attached to the server endpoints and network ACLs attached to the subnet that the endpoint is in.</p>

Characteristic	Public	VPC - Internet	VPC - Internal	VPC_Endpoint (deprecated)
Client firewall allow list	<p>You must allow the DNS name of the server.</p> <p>Because IP addresses are subject to change, avoid using IP addresses for your client firewall allow list.</p>	<p>You can allow the DNS name of the server or the Elastic IP addresses attached to the server.</p>	<p>You can allow the private IP addresses or the DNS name of the endpoints.</p>	<p>You can allow the private IP addresses or the DNS name of the endpoints.</p>

Note

The VPC_ENDPOINT endpoint type is now deprecated and cannot be used to create new servers. Instead of using `EndpointType=VPC_ENDPOINT`, use the new VPC endpoint type (`EndpointType=VPC`), which you can use as either **Internal** or **Internet Facing**, as described in the preceding table. For details, see [Discontinuing the use of VPC_ENDPOINT](#).

Consider the following options to increase the security posture of your AWS Transfer Family server:

- Use a VPC endpoint with internal access, so that the server is accessible only to clients within your VPC or VPC-connected environments such as an on-premises data center over AWS Direct Connect or VPN.
- To allow clients to access the endpoint over the internet and protect your server, use a VPC endpoint with internet-facing access. Then, modify the VPC's security groups to allow traffic only from certain IP addresses that host your users' clients.
- If you require password-based authentication and you use a custom identity provider with your server, it's a best practice that your password policy prevents users from creating weak passwords and limits the number of failed login attempts.

- AWS Transfer Family is a managed service, and so it doesn't provide shell access. You cannot directly access the underlying SFTP server to run OS native commands on Transfer Family servers.
- Use a Network Load Balancer in front of a VPC endpoint with internal access. Change the listener port on the load balancer from port 22 to a different port. This can reduce, but not eliminate, the risk of port scanners and bots probing your server, because port 22 is most commonly used for scanning. For details, see the blog post [Network Load Balancers now support Security groups](#).

Note

If you use a Network Load Balancer, the AWS Transfer Family CloudWatch logs show the IP address for the NLB, rather than the actual client IP address.

Configuring an SFTP, FTPS, or FTP server endpoint

You can create a file transfer server by using the AWS Transfer Family service. The following file transfer protocols are available:

- Secure Shell (SSH) File Transfer Protocol (SFTP) – File transfer over SSH. For details, see [the section called “Create an SFTP-enabled server”](#).

Note

We provide an AWS CDK example for creating an SFTP Transfer Family server. The example uses TypeScript, and is available on GitHub [here](#).

- File Transfer Protocol Secure (FTPS) – File transfer with TLS encryption. For details, see [the section called “Create an FTPS-enabled server”](#).
- File Transfer Protocol (FTP) – Unencrypted file transfer. For details, see [the section called “Create an FTP-enabled server”](#).
- Applicability Statement 2 (AS2) – File transfer for transporting structured business-to-business data. For details, see [the section called “Configure AS2”](#). For AS2, you can quickly create an AWS CloudFormation stack for demonstration purposes. This procedure is described in [Use a template to create a demo Transfer Family AS2 stack](#).

You can create a server with multiple protocols.

Note

If you have multiple protocols enabled for the same server endpoint and you want to provide access by using the same username over multiple protocols, you can do so as long as the credentials specific to the protocol have been set up in your identity provider. For FTP, we recommend maintaining separate credentials from SFTP and FTPS. This is because, unlike SFTP and FTPS, FTP transmits credentials in clear text. By isolating FTP credentials from SFTP or FTPS, if FTP credentials are shared or exposed, your workloads using SFTP or FTPS remain secure.

When you create a server, you choose a specific AWS Region to perform the file operation requests of users who are assigned to that server. Along with assigning the server one or more protocols, you also assign one of the following identity provider types:

- **Service managed by using SSH keys.** For details, see [Working with service-managed users](#).
- **AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD).** This method allows you integrate your Microsoft Active Directory groups to provide access to your Transfer Family servers. For details, see [Using AWS Directory Service for Microsoft Active Directory](#).
- **A custom method.** The custom identity provider method uses AWS Lambda or Amazon API Gateway and enables you to integrate your directory service to authenticate and authorize your users. The service automatically assigns an identifier that uniquely identifies your server. For details, see [Working with custom identity providers](#). Transfer Family provides AWS CloudFormation templates that you can use to quickly deploy servers that use a custom identity provider.
 - [Lambda functions for authentication](#) describes CloudFormation templates that use a Lambda function for authentication.
 - [Authenticating using an API Gateway method](#) describes CloudFormation templates that use an Amazon API Gateway method for authentication.

You also assign the server an endpoint type (publicly accessible or VPC hosted) and a hostname by using the default server endpoint, or a custom hostname by using the Amazon Route 53 service or by using a Domain Name System (DNS) service of your choice. A server hostname must be unique in the AWS Region where it's created.

Additionally, you can assign an Amazon CloudWatch logging role to push events to your CloudWatch logs, choose a security policy that contains the cryptographic algorithms that are enabled for use by your server, and add metadata to the server in the form of tags that are key-value pairs.

Important

You incur costs for instantiated servers and for data transfer. For information about pricing and to use AWS Pricing Calculator to get an estimate of the cost to use Transfer Family, see [AWS Transfer Family pricing](#).

Create an SFTP-enabled server

Secure Shell (SSH) File Transfer Protocol (SFTP) is a network protocol used for secure transfer of data over the internet. The protocol supports the full security and authentication functionality of SSH. It's widely used to exchange data, including sensitive information between business partners in a variety of industries such as financial services, healthcare, retail, and advertising.

Note

SFTP servers for Transfer Family operate over port 22. For VPC-hosted endpoints, SFTP Transfer Family servers can also operate over port 2222 or port 22000. For details, see [Create a server in a virtual private cloud](#).


See also

- We provide an AWS CDK example for creating an SFTP Transfer Family server. The example uses TypeScript, and is available on GitHub [here](#).
- For a walkthrough of how to deploy a Transfer Family server inside of a VPC, see [Use IP allow list to secure your AWS Transfer Family servers](#).

To create an SFTP-enabled server

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/> and select **Servers** from the navigation pane, then choose **Create server**.

2. In **Choose protocols**, select **SFTP**, and then choose **Next**.
3. In **Choose an identity provider**, choose the identity provider that you want to use to manage user access. You have the following options:
 - **Service managed** – You store user identities and keys in AWS Transfer Family.
 - **AWS Directory Service for Microsoft Active Directory** – You provide an AWS Directory Service directory to access the endpoint. By doing so, you can use credentials stored in your Active Directory to authenticate your users. To learn more about working with AWS Managed Microsoft AD identity providers, see [Using AWS Directory Service for Microsoft Active Directory](#).

 **Note**

- Cross-Account and Shared directories are not supported for AWS Managed Microsoft AD.
- To set up a server with Directory Service as your identity provider, you need to add some AWS Directory Service permissions. For details, see [Before you start using AWS Directory Service for Microsoft Active Directory](#).

- **Custom identity provider** – Choose either of the following options:
 - **Use AWS Lambda to connect your identity provider** – You can use an existing identity provider, backed by a Lambda function. You provide the name of the Lambda function. For more information, see [Using AWS Lambda to integrate your identity provider](#).
 - **Use Amazon API Gateway to connect your identity provider** – You can create an API Gateway method backed by a Lambda function for use as an identity provider. You provide an Amazon API Gateway URL and an invocation role. For more information, see [Using Amazon API Gateway to integrate your identity provider](#).

For either option, you can also specify how to authenticate.

- **Password OR Key** – users can authenticate with either their password or their key. This is the default value.
- **Password ONLY** – users must provide their password to connect.
- **Key ONLY** – users must provide their private key to connect.
- **Password AND Key** – users must provide both their private key and their password to connect. The server checks the key first, and then if the key is valid, the system prompts

for a password. If the private key provided does not match the public key that is stored, authentication fails.

Choose an identity provider

Identity Provider for SFTP, FTPS, or FTP

Identity provider type
An identity provider manages user access for authentication and authorization

Service managed
Create and manage users within the service

AWS Directory Service [Info](#)
Enable users in AWS Managed AD or use your own self-managed AD in your on-premises environment or in AWS

Custom Identity Provider [Info](#)
Manage users by integrating an identity provider of your choice

Use AWS Lambda to connect your identity provider [Info](#)
Invoke an AWS Lambda function to call your identity provider's API for user authentication and authorization

Use Amazon API Gateway to connect your identity provider [Info](#)
Use a RESTful API method to call your identity provider's API for user authentication and authorization

AWS Lambda function

Choose a Lambda function
▼
↻

Authentication methods
Choose which authentication methods are required for users to connect to your server

Password OR public key

Password ONLY

Public Key ONLY

Password AND public key

i Either a valid password or valid private key will be required during user authentication

Cancel
Previous
Next

4. Choose **Next**.
5. In **Choose an endpoint**, do the following:
 - a. For **Endpoint type**, choose the **Publicly accessible** endpoint type. For a **VPC hosted** endpoint, see [Create a server in a virtual private cloud](#).
 - b. (Optional) For **Custom hostname**, choose **None**.

You get a server hostname provided by AWS Transfer Family. The server hostname takes the form `serverId.server.transfer.regionId.amazonaws.com`.

For a custom hostname, you specify a custom alias for your server endpoint. To learn more about working with custom hostnames, see [Working with custom hostnames](#).

- c. (Optional) For **FIPS Enabled**, select the **FIPS Enabled endpoint** check box to ensure that the endpoint complies with Federal Information Processing Standards (FIPS).

 **Note**

FIPS-enabled endpoints are only available in North American AWS Regions. For available Regions, see [AWS Transfer Family endpoints and quotas](#) in the *AWS General Reference*. For more information about FIPS, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

- d. Choose **Next**.
6. On the **Choose domain** page, choose the AWS storage service that you want to use to store and access your data over the selected protocol:
 - Choose **Amazon S3** to store and access your files as objects over the selected protocol.
 - Choose **Amazon EFS** to store and access your files in your Amazon EFS file system over the selected protocol.

Choose **Next**.

7. In **Configure additional details**, do the following:
 - a. For logging, specify an existing log group or create a new one (the default option). If you choose an existing log group, you must select one that is associated with your AWS account.

Transfer Family > Servers > Create server

Step 1
Choose protocols

Step 2
Choose an identity provider

Step 3
Choose an endpoint

Step 4
Choose a domain

Step 5
Configure additional details

Step 6
Review and create

Configure additional details

Logging Info

Log group Info
Choose the CloudWatch log group where your events will be delivered in a structured JSON format

Create a new log group Choose an existing log group

Logging role Info
Choose the IAM role that will be used to deliver events to your CloudWatch logs

Create a new role Choose an existing role

Info Logging role is only required when selecting a workflow in the Managed workflows section below.

If you choose **Create log group**, the CloudWatch console (<https://console.aws.amazon.com/cloudwatch/>) opens to the **Create log group** page. For details, see [Create a log group in CloudWatch Logs](#).

- b. (Optional) For **Managed workflows**, choose workflow IDs (and a corresponding role) that Transfer Family should assume when executing the workflow. You can choose one workflow to execute upon a complete upload, and another to execute upon a partial upload. To learn more about processing your files by using managed workflows, see [AWS Transfer Family managed workflows](#).

Managed workflows Info


Workflow for complete file uploads
Select the workflow that AWS Transfer Family should run on all files that are uploaded in full via this server

Workflow for partial file uploads
Select the workflow that Transfer Family should run on all files that are only partially uploaded via this server

Managed workflows execution role Info
Select the role that AWS Transfer Family should assume when executing a workflow

- c. For **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server. Our latest security policy is the default: for details, see [Security policies for AWS Transfer Family servers](#).
- d. (Optional) For **Server Host Key**, enter an RSA, ED25519, or ECDSA private key that will be used to identify your server when clients connect to it over SFTP. You can also add a description to differentiate among multiple host keys.

After you create your server, you can add additional host keys. Having multiple host keys is useful if you want to rotate keys or if you want to have different types of keys, such as an RSA key and also an ECDSA key.

 **Note**

The **Server Host Key** section is used only for migrating users from an existing SFTP-enabled server.

- e. (Optional) For **Tags**, for **Key** and **Value**, enter one or more tags as key-value pairs, and then choose **Add tag**.
- f. Choose **Next**.
- g. You can optimize performance for your Amazon S3 directories. For example, suppose that you go into your home directory, and you have 10,000 subdirectories. In other words, your Amazon S3 bucket has 10,000 folders. In this scenario, if you run the `ls` (list) command, the list operation takes between six and eight minutes. However, if you optimize your directories, this operation takes only a few seconds.

When you create your server using the console, optimized directories is enabled by default. If you create your server using the API, this behavior is not enabled by default.

Optimized Directories [Info](#)

Your logical directories can now support mappings up to 2.1MB for both Amazon S3 and EFS

Select this option to improve performance of the listing of your folders in your S3 bucket

Enable

Turning this option off restores to the default performance to list your S3 directory

- h. (Optional) Configure AWS Transfer Family servers to display customized messages such as organizational policies or terms and conditions to your end users. For **Display banner**, in the **Pre-authentication display banner** text box, enter the text message that you want to display to your users before they authenticate.
- i. (Optional) You can configure the following additional options.
 - **SetStat option:** enable this option to ignore the error that is generated when a client attempts to use SETSTAT on a file you are uploading to an Amazon S3 bucket. For additional details, see the `SetStatOption` documentation in the [ProtocolDetails](#).
 - **TLS session resumption:** this option is only available if you have enabled FTPS as one of the protocols for this server.
 - **Passive IP:** this option is only available if you have enabled FTPS or FTP as one of the protocols for this server.


Additional configuration

SetStat option - optional [Info](#)
Select whether you want this server to ignore SetStat command

Enable


TLS session resumption - optional [Info](#)
Choose how you want your server to process TLS session resumption requests

Enforce
 Enable
 Disable

 To enable TLS session resumption, enable FTPS as one of the protocols selected in Step 1


Passive IP - optional [Info](#)
Provide passive IP (PASV) that file transfer clients can use to connect this server

1.2.3.4

 To enable Passive IP, enable FTP or FTPS as one of the protocols selected in Step 1

8. In **Review and create**, review your choices.

- If you want to edit any of them, choose **Edit** next to the step.

 **Note**

You must review each step after the step that you chose to edit.

- If you have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

It can take a couple of minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations, but you'll need to create a user first. For details on creating users, see [Managing users for server endpoints](#).

Create an FTPS-enabled server

File Transfer Protocol over SSL (FTPS) is an extension to FTP. It uses Transport Layer Security (TLS) and Secure Sockets Layer (SSL) cryptographic protocols to encrypt traffic. FTPS allows encryption of both the control and data channel connections either concurrently or independently.

To create an FTPS-enabled server

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/> and select **Servers** from the navigation pane, then choose **Create server**.
2. In **Choose protocols**, select **FTPS**.

For **Server certificate**, choose a certificate stored in AWS Certificate Manager (ACM) which will be used to identify your server when clients connect to it over FTPS and then choose **Next**.

To request a new public certificate, see [Request a public certificate](#) in the *AWS Certificate Manager User Guide*.

To import an existing certificate into ACM, see [Importing certificates into ACM](#) in the *AWS Certificate Manager User Guide*.

To request a private certificate to use FTPS through private IP addresses, see [Requesting a Private Certificate](#) in the *AWS Certificate Manager User Guide*.

Certificates with the following cryptographic algorithms and key sizes are supported:

- 2048-bit RSA (RSA_2048)
- 4096-bit RSA (RSA_4096)
- Elliptic Prime Curve 256 bit (EC_prime256v1)
- Elliptic Prime Curve 384 bit (EC_secp384r1)
- Elliptic Prime Curve 521 bit (EC_secp521r1)

Note

The certificate must be a valid SSL/TLS X.509 version 3 certificate with FQDN or IP address specified and contain information about the issuer.

3. In **Choose an identity provider**, choose the identity provider that you want to use to manage user access. You have the following options:

- **AWS Directory Service for Microsoft Active Directory** – You provide an AWS Directory Service directory to access the endpoint. By doing so, you can use credentials stored in your Active Directory to authenticate your users. To learn more about working with AWS Managed Microsoft AD identity providers, see [Using AWS Directory Service for Microsoft Active Directory](#).

Note

- Cross-Account and Shared directories are not supported for AWS Managed Microsoft AD.
- To set up a server with Directory Service as your identity provider, you need to add some AWS Directory Service permissions. For details, see [Before you start using AWS Directory Service for Microsoft Active Directory](#).

- **Custom identity provider** – Choose either of the following options:
 - **Use AWS Lambda to connect your identity provider** – You can use an existing identity provider, backed by a Lambda function. You provide the name of the Lambda function. For more information, see [Using AWS Lambda to integrate your identity provider](#).
 - **Use Amazon API Gateway to connect your identity provider** – You can create an API Gateway method backed by a Lambda function for use as an identity provider. You provide an Amazon API Gateway URL and an invocation role. For more information, see [Using Amazon API Gateway to integrate your identity provider](#).

Choose an identity provider

Identity Provider for SFTP, FTPS, or FTP

Identity provider type
An identity provider manages user access for authentication and authorization

Service managed
Create and manage users within the service

AWS Directory Service [Info](#)
Enable users in AWS Managed AD or use your own self-managed AD in your on-premises environment or in AWS

Custom Identity Provider [Info](#)
Manage users by integrating an identity provider of your choice

Use AWS Lambda to connect your identity provider [Info](#)
Invoke an AWS Lambda function to call your identity provider's API for user authentication and authorization

Use Amazon API Gateway to connect your identity provider [Info](#)
Use a RESTful API method to call your identity provider's API for user authentication and authorization

AWS Lambda function

Choose a Lambda function
▼
↻

Authentication methods
Choose which authentication methods are required for users to connect to your server

- Password OR public key
- Password ONLY
- Public Key ONLY
- Password AND public key

[i](#) To choose an authentication method, enable SFTP as one of the protocols selected in Step 1

Cancel
Previous
Next

4. Choose **Next**.
5. In **Choose an endpoint**, do the following:

[i](#) **Note**


FTPS servers for Transfer Family operate over Port 21 (Control Channel) and Port Range 8192–8200 (Data Channel).

- a. For **Endpoint type**, choose the **VPC hosted** endpoint type to host your server's endpoint. For information about setting up your VPC hosted endpoint, see [Create a server in a virtual private cloud](#).

 **Note**

Publicly accessible endpoints are not supported.

- b. (Optional) For **FIPS Enabled**, select the **FIPS Enabled endpoint** check box to ensure that the endpoint complies with Federal Information Processing Standards (FIPS).

 **Note**

FIPS-enabled endpoints are only available in North American AWS Regions. For available Regions, see [AWS Transfer Family endpoints and quotas](#) in the *AWS General Reference*. For more information about FIPS, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

- c. Choose **Next**.

Choose an endpoint

Endpoint configuration [Info](#)

Endpoint type
Select whether the endpoint will be publicly accessible or hosted inside your VPC

Publicly accessible
Accessible over the internet

VPC hosted [Info](#)
Access controlled using Security Groups

Access [Info](#)

Internal

Internet Facing

VPC
Select a VPC ID

FIPS Enabled
Select whether the endpoint should comply with Federal Information Processing Standards (FIPS)

FIPS Enabled endpoint

6. On the **Choose domain** page, choose the AWS storage service that you want to use to store and access your data over the selected protocol:
 - Choose **Amazon S3** to store and access your files as objects over the selected protocol.
 - Choose **Amazon EFS** to store and access your files in your Amazon EFS file system over the selected protocol.

Choose **Next**.

7. In **Configure additional details**, do the following:
 - a. For logging, specify an existing log group or create a new one (the default option).

Transfer Family > Servers > Create server

Step 1
Choose protocols

Step 2
Choose an identity provider

Step 3
Choose an endpoint

Step 4
Choose a domain

Step 5
Configure additional details

Step 6
Review and create

Configure additional details

Logging Info

Log group Info
Choose the CloudWatch log group where your events will be delivered in a structured JSON format

Create a new log group Choose an existing log group

Logging role Info
Choose the IAM role that will be used to deliver events to your CloudWatch logs

Create a new role Choose an existing role

Info Logging role is only required when selecting a workflow in the Managed workflows section below.

If you choose **Create log group**, the CloudWatch console (<https://console.aws.amazon.com/cloudwatch/>) opens to the **Create log group** page. For details, see [Create a log group in CloudWatch Logs](#).

- b. (Optional) For **Managed workflows**, choose workflow IDs (and a corresponding role) that Transfer Family should assume when executing the workflow. You can choose one workflow to execute upon a complete upload, and another to execute upon a partial upload. To learn more about processing your files by using managed workflows, see [AWS Transfer Family managed workflows](#).

Managed workflows Info

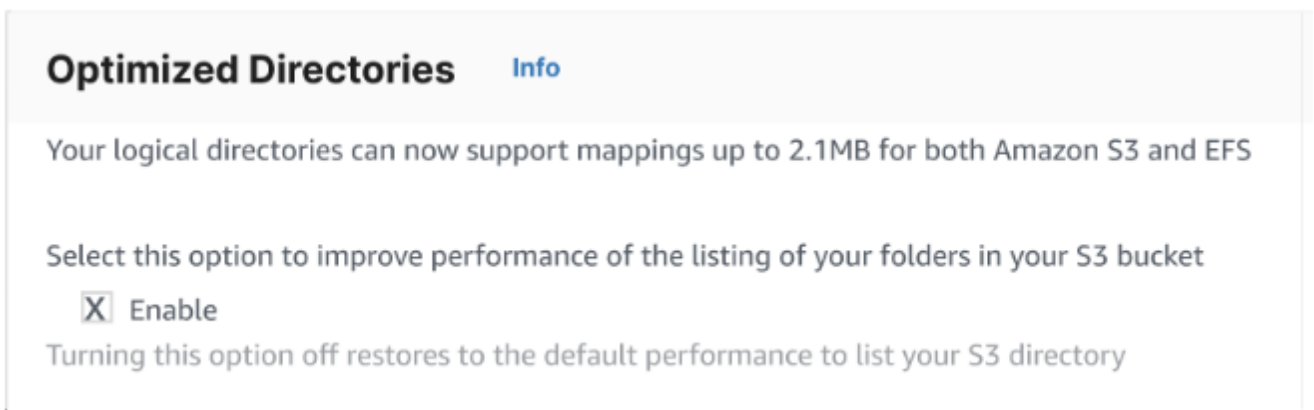
Workflow for complete file uploads
Select the workflow that AWS Transfer Family should run on all files that are uploaded in full via this server

Workflow for partial file uploads
Select the workflow that Transfer Family should run on all files that are only partially uploaded via this server

Managed workflows execution role Info
Select the role that AWS Transfer Family should assume when executing a workflow

- c. For **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server. Our latest security policy is the default: for details, see [Security policies for AWS Transfer Family servers](#).
- d. For **Server Host Key**, keep it blank.
- e. (Optional) For **Tags**, for **Key** and **Value**, enter one or more tags as key-value pairs, and then choose **Add tag**.
- f. You can optimize performance for your Amazon S3 directories. For example, suppose that you go into your home directory, and you have 10,000 subdirectories. In other words, your Amazon S3 bucket has 10,000 folders. In this scenario, if you run the `ls` (list) command, the list operation takes between six and eight minutes. However, if you optimize your directories, this operation takes only a few seconds.

When you create your server using the console, optimized directories is enabled by default. If you create your server using the API, this behavior is not enabled by default.



- g. Choose **Next**.
- h. (Optional) You can configure AWS Transfer Family servers to display customized messages such as organizational policies or terms and conditions to your end users. You can also display customized Message of The Day (MOTD) to users who have successfully authenticated.

For **Display banner**, in the **Pre-authentication display banner** text box, enter the text message that you want to display to your users before they authenticate, and in the **Post-authentication display banner** text box, enter the text that you want to display to your users after they successfully authenticate.

- i. (Optional) You can configure the following additional options.

- **SetStat option:** enable this option to ignore the error that is generated when a client attempts to use SETSTAT on a file you are uploading to an Amazon S3 bucket. For additional details, see the SetStatOption documentation in the [ProtocolDetails](#) topic.
- **TLS session resumption:** provides a mechanism to resume or share a negotiated secret key between the control and data connection for an FTPS session. For additional details, see the TlsSessionResumptionMode documentation in the [ProtocolDetails](#) topic.
- **Passive IP:** indicates passive mode, for FTP and FTPS protocols. Enter a single IPv4 address, such as the public IP address of a firewall, router, or load balancer. For additional details, see the PassiveIp documentation in the [ProtocolDetails](#) topic.

Additional configuration

SetStat option - optional [Info](#)
Select whether you want this server to ignore SetStat command

Enable

TLS session resumption - optional [Info](#)
Choose how you want your server to process TLS session resumption requests

Enforce
 Enable
 Disable

Passive IP - optional [Info](#)
Provide passive IP (PASV) that file transfer clients can use to connect this server

1.2.3.4

8. In **Review and create**, review your choices.

- If you want to edit any of them, choose **Edit** next to the step.

Note

You must review each step after the step that you chose to edit.

- If you have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

It can take a couple of minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations for your users.

Next steps: For the next step, continue on to [Working with custom identity providers](#) to set up users.

Create an FTP-enabled server

File Transfer Protocol (FTP) is a network protocol used for the transfer of data. FTP uses a separate channel for control and data transfers. The control channel is open until terminated or inactivity timeout. The data channel is active for the duration of the transfer. FTP uses clear text and does not support encryption of traffic.

Note

When you enable FTP, you must choose the internal access option for the VPC-hosted endpoint. If you need your server to have data traverse the public network, you must use secure protocols, such as SFTP or FTPS.

To create an FTP-enabled server

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/> and select **Servers** from the navigation pane, then choose **Create server**.
2. In **Choose protocols**, select **FTP**, and then choose **Next**.
3. In **Choose an identity provider**, choose the identity provider that you want to use to manage user access. You have the following options:
 - **AWS Directory Service for Microsoft Active Directory** – You provide an AWS Directory Service directory to access the endpoint. By doing so, you can use credentials stored in your Active Directory to authenticate your users. To learn more about working with AWS Managed Microsoft AD identity providers, see [Using AWS Directory Service for Microsoft Active Directory](#).

Note

- Cross-Account and Shared directories are not supported for AWS Managed Microsoft AD.
- To set up a server with Directory Service as your identity provider, you need to add some AWS Directory Service permissions. For details, see [Before you start using AWS Directory Service for Microsoft Active Directory](#).

- **Custom identity provider** – Choose either of the following options:
 - **Use AWS Lambda to connect your identity provider** – You can use an existing identity provider, backed by a Lambda function. You provide the name of the Lambda function. For more information, see [Using AWS Lambda to integrate your identity provider](#).
 - **Use Amazon API Gateway to connect your identity provider** – You can create an API Gateway method backed by a Lambda function for use as an identity provider. You provide an Amazon API Gateway URL and an invocation role. For more information, see [Using Amazon API Gateway to integrate your identity provider](#).

Choose an identity provider

Identity Provider for SFTP, FTPS, or FTP

Identity provider type
An identity provider manages user access for authentication and authorization

Service managed
Create and manage users within the service

AWS Directory Service [Info](#)
Enable users in AWS Managed AD or use your own self-managed AD in your on-premises environment or in AWS

Custom Identity Provider [Info](#)
Manage users by integrating an identity provider of your choice

Use AWS Lambda to connect your identity provider [Info](#)
Invoke an AWS Lambda function to call your identity provider's API for user authentication and authorization

Use Amazon API Gateway to connect your identity provider [Info](#)
Use a RESTful API method to call your identity provider's API for user authentication and authorization

AWS Lambda function

Choose a Lambda function
▼
↻

Authentication methods
Choose which authentication methods are required for users to connect to your server

Password OR public key

Password ONLY

Public Key ONLY

Password AND public key

[i](#) To choose an authentication method, enable SFTP as one of the protocols selected in Step 1

Cancel
Previous
Next

4. Choose **Next**.
5. In **Choose an endpoint**, do the following:

[i](#) **Note**

FTP servers for Transfer Family operate over Port 21 (Control Channel) and Port Range 8192–8200 (Data Channel).

- a. For **Endpoint type**, choose **VPC hosted** to host your server's endpoint. For information about setting up your VPC hosted endpoint, see [Create a server in a virtual private cloud](#).

Note

Publicly accessible endpoints are not supported.

- b. For **FIPS Enabled**, keep the **FIPS Enabled endpoint** check box cleared.

Note

FIPS-enabled endpoints are not supported for FTP servers.

- c. Choose **Next**.

Choose an endpoint

Endpoint configuration [Info](#)

Endpoint type
Select whether the endpoint will be publicly accessible or hosted inside your VPC

Publicly accessible
Accessible over the internet

VPC hosted [Info](#)
Access controlled using Security Groups

Access [Info](#)

Internal

Internet Facing

VPC
Select a VPC ID

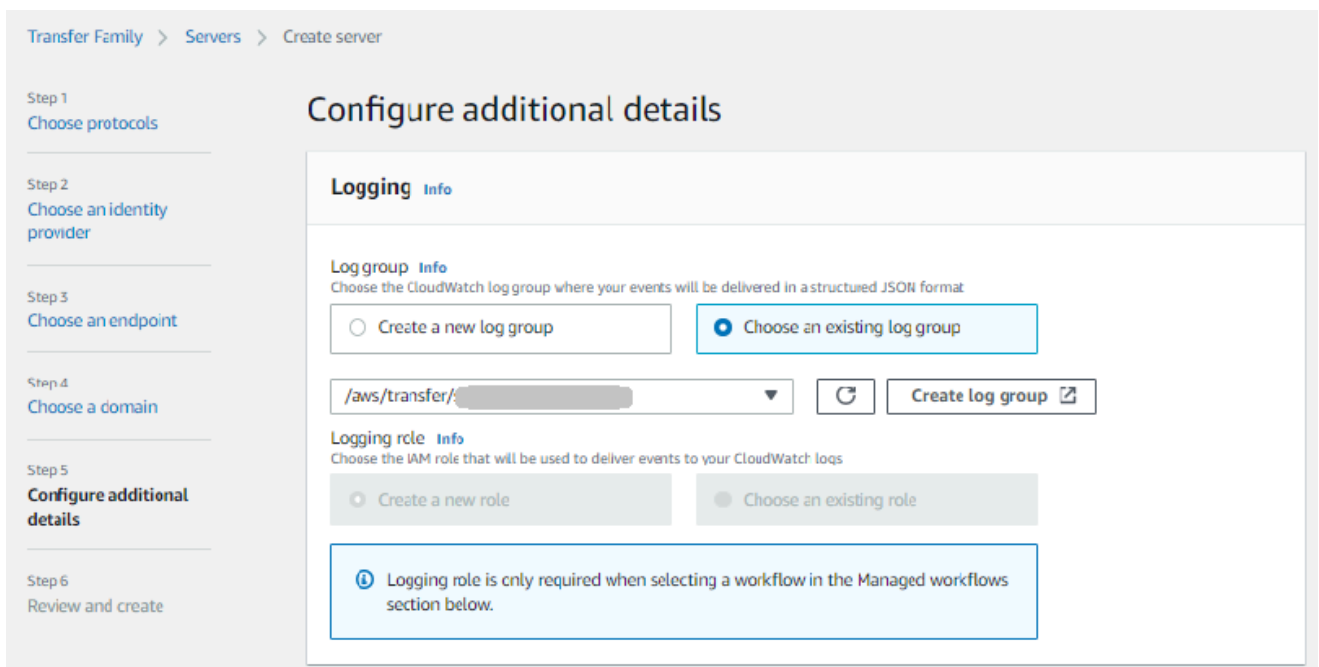
FIPS Enabled
Select whether the endpoint should comply with Federal Information Processing Standards (FIPS)

FIPS Enabled endpoint

6. On the **Choose domain** page, choose the AWS storage service that you want to use to store and access your data over the selected protocol.
 - Choose **Amazon S3** to store and access your files as objects over the selected protocol.
 - Choose **Amazon EFS** to store and access your files in your Amazon EFS file system over the selected protocol.

Choose **Next**.

7. In **Configure additional details**, do the following:
 - a. For logging, specify an existing log group or create a new one (the default option).



Transfer Family > Servers > Create server

Step 1
Choose protocols

Step 2
Choose an identity provider

Step 3
Choose an endpoint

Step 4
Choose a domain

Step 5
Configure additional details

Step 6
Review and create

Configure additional details

Logging Info

Log group Info
Choose the CloudWatch log group where your events will be delivered in a structured JSON format

Create a new log group Choose an existing log group

Logging role Info
Choose the IAM role that will be used to deliver events to your CloudWatch logs

Create a new role Choose an existing role

Info Logging role is only required when selecting a workflow in the Managed workflows section below.

If you choose **Create log group**, the CloudWatch console (<https://console.aws.amazon.com/cloudwatch/>) opens to the **Create log group** page. For details, see [Create a log group in CloudWatch Logs](#).

- b. (Optional) For **Managed workflows**, choose workflow IDs (and a corresponding role) that Transfer Family should assume when executing the workflow. You can choose one workflow to execute upon a complete upload, and another to execute upon a partial upload. To learn more about processing your files by using managed workflows, see [AWS Transfer Family managed workflows](#).

Managed workflows [Info](#)

Workflow for complete file uploads
Select the workflow that AWS Transfer Family should run on all files that are uploaded in full via this server

w- [redacted] ▼ [Refresh] [Create a new Workflow ↗]

Workflow for partial file uploads
Select the workflow that Transfer Family should run on all files that are only partially uploaded via this server

w- [redacted] ▼ [Refresh] [Create a new Workflow ↗]

Managed workflows execution role [Info](#)
Select the role that AWS Transfer Family should assume when executing a workflow

[redacted] ▼ [Refresh]

- c. For **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server.

Note

Transfer Family assigns the latest security policy to your FTP server. However, since the FTP protocol doesn't use any encryption, FTP servers do not use any of the security policy algorithms. Unless your server also uses the FTPS or SFTP protocol, the security policy remains unused.

- d. For **Server Host Key**, keep it blank.
- e. (Optional) For **Tags**, for **Key** and **Value**, enter one or more tags as key-value pairs, and then choose **Add tag**.
- f. You can optimize performance for your Amazon S3 directories. For example, suppose that you go into your home directory, and you have 10,000 subdirectories. In other words, your Amazon S3 bucket has 10,000 folders. In this scenario, if you run the `ls` (list) command, the list operation takes between six and eight minutes. However, if you optimize your directories, this operation takes only a few seconds.

When you create your server using the console, optimized directories is enabled by default. If you create your server using the API, this behavior is not enabled by default.

Optimized Directories [Info](#)

Your logical directories can now support mappings up to 2.1MB for both Amazon S3 and EFS

Select this option to improve performance of the listing of your folders in your S3 bucket

Enable

Turning this option off restores to the default performance to list your S3 directory

- g. Choose **Next**.
- h. (Optional) You can configure AWS Transfer Family servers to display customized messages such as organizational policies or terms and conditions to your end users. You can also display customized Message of The Day (MOTD) to users who have successfully authenticated.

For **Display banner**, in the **Pre-authentication display banner** text box, enter the text message that you want to display to your users before they authenticate, and in the **Post-authentication display banner** text box, enter the text that you want to display to your users after they successfully authenticate.

- i. (Optional) You can configure the following additional options.
 - **SetStat option:** enable this option to ignore the error that is generated when a client attempts to use SETSTAT on a file you are uploading to an Amazon S3 bucket. For additional details, see the `SetStatOption` documentation in the [ProtocolDetails](#) topic.
 - **TLS session resumption:** provides a mechanism to resume or share a negotiated secret key between the control and data connection for an FTPS session. For additional details, see the `TlsSessionResumptionMode` documentation in the [ProtocolDetails](#) topic.
 - **Passive IP:** indicates passive mode, for FTP and FTPS protocols. Enter a single IPv4 address, such as the public IP address of a firewall, router, or load balancer. For additional details, see the `PassiveIp` documentation in the [ProtocolDetails](#) topic.

Additional configuration

SetStat option - optional [Info](#)
Select whether you want this server to ignore SetStat command

Enable

TLS session resumption - optional [Info](#)
Choose how you want your server to process TLS session resumption requests

Enforce
 Enable
 Disable

Passive IP - optional [Info](#)
Provide passive IP (PASV) that file transfer clients can use to connect this server

1.2.3.4

8. In **Review and create**, review your choices.

- If you want to edit any of them, choose **Edit** next to the step.

Note

You must review each step after the step that you chose to edit.

- If you have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

It can take a couple of minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations for your users.

Next steps – For the next step, continue on to [Working with custom identity providers](#) to set up users.

Create a server in a virtual private cloud

You can host your server's endpoint inside a virtual private cloud (VPC) to use for transferring data to and from an Amazon S3 bucket or Amazon EFS file system without going over the public internet.

Note

After May 19, 2021, you won't be able to create a server using `EndpointType=VPC_ENDPOINT` in your AWS account if your account hasn't already done so before May 19, 2021. If you have already created servers with `EndpointType=VPC_ENDPOINT` in your AWS account on or before February 21, 2021, you will not be affected. After this date, use `EndpointType=VPC`. For more information, see [the section called "Discontinuing the use of VPC_ENDPOINT"](#).

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish a private connection between your VPC and a server. You can then use this server to transfer data over your client to and from your Amazon S3 bucket without using public IP addressing or requiring an internet gateway.

Using Amazon VPC, you can launch AWS resources in a custom virtual network. You can use a VPC to control your network settings, such as the IP address range, subnets, route tables, and network gateways. For more information about VPCs, see [What Is Amazon VPC?](#) in the *Amazon VPC User Guide*.

In the next sections, find instructions on how to create and connect your VPC to a server. As an overview, you do this as follows:

1. Set up a server using a VPC endpoint.
2. Connect to your server using a client that is inside your VPC through the VPC endpoint. Doing this enables you to transfer data that is stored in your Amazon S3 bucket over your client using AWS Transfer Family. You can perform this transfer even though the network is disconnected from the public internet.
3. In addition, if you choose to make your server's endpoint internet-facing, you can associate Elastic IP addresses with your endpoint. Doing this lets clients outside of your VPC connect to your server. You can use VPC security groups to control access to authenticated users whose requests originate only from allowed addresses.

Topics

- [Create a server endpoint that can be accessed only within your VPC](#)
- [Create an internet-facing endpoint for your server](#)
- [Change the endpoint type for your server](#)
- [Discontinuing the use of VPC_ENDPOINT](#)
- [Updating the AWS Transfer Family server endpoint type from VPC_ENDPOINT to VPC](#)

Create a server endpoint that can be accessed only within your VPC

In the following procedure, you create a server endpoint that is accessible only to resources within your VPC.

To create a server endpoint inside a VPC

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. From the navigation pane, select **Servers**, then choose **Create server**.
3. In **Choose protocols**, select one or more protocols, and then choose **Next**. For more information about protocols, see [Step 2: Create an SFTP-enabled server](#).
4. In **Choose an identity provider**, choose **Service managed** to store user identities and keys in AWS Transfer Family, and then choose **Next**.

Note


This procedure uses the service-managed option. If you choose **Custom**, you provide an Amazon API Gateway endpoint and an AWS Identity and Access Management (IAM) role to access the endpoint. By doing so, you can integrate your directory service to authenticate and authorize your users. To learn more about working with custom identity providers, see [Working with custom identity providers](#).

5. In **Choose an endpoint**, do the following:

Note


FTP and FTPS servers for Transfer Family operate over Port 21 (Control Channel) and Port Range 8192-8200 (Data Channel).

- a. For **Endpoint type**, choose the **VPC hosted** endpoint type to host your server's endpoint.
- b. For **Access**, choose **Internal** to make your endpoint only accessible to clients using the endpoint's private IP addresses.

 **Note**

For details on the **Internet Facing** option, see [Create an internet-facing endpoint for your server](#). A server that is created in a VPC for internal access only doesn't support custom hostnames.

- c. For **VPC**, choose an existing VPC ID or choose **Create a VPC** to create a new VPC.
- d. In the **Availability Zones** section, choose up to three Availability Zones and associated subnets.
- e. In the **Security Groups** section, choose an existing security group ID or IDs or choose **Create a security group** to create a new security group. For more information about security groups, see [Security groups for your VPC](#) in the *Amazon Virtual Private Cloud User Guide*. To create a security group, see [Creating a security group](#) in the *Amazon Virtual Private Cloud User Guide*.

 **Note**

Your VPC automatically comes with a default security group. If you don't specify a different security group or groups when you launch the server, we associate the default security group with your server.

For the inbound rules for the security group, you can configure SSH traffic to use port 22, 2222, 22000, or any combination. Port 22 is configured by default. To use port 2222 or port 22000, you add an inbound rule to your security group. For the type, choose **Custom TCP**, then enter either **2222** or **22000** for **Port range**, and for the source, enter the same CIDR range that you have for your SSH port 22 rule.

 **Note**

You can also use port 2223 for clients that require TCP "piggy-back" ACKs, or the ability for the final ack of the TCP 3-way handshake to also contain data.

Some client software may be incompatible with port 2223: for example, a client that requires the server to send the SFTP Identification String before the client does.

The screenshot shows the 'Edit inbound rules' interface in the AWS Management Console. The page title is 'Edit inbound rules' and it includes a breadcrumb trail: 'VPC > Security Groups > sg-...-default > Edit inbound rules'. Below the title, there is a sub-header 'Inbound rules' and a table of rules. The table has columns for 'Security group rule ID', 'Type', 'Protocol', 'Port range', and 'Source'. The fourth rule is highlighted with a red box. This rule is a 'Custom TCP' rule for port 2222, with a source of 72.21.196.64/32. Other rules include HTTP (port 80), RDP (port 3389), HTTPS (port 443), and SSH (port 22). An 'Add rule' button is visible at the bottom left of the table.

Security group rule ID	Type	Protocol	Port range	Source
sgr-...	HTTP	TCP	80	0.0.0.0/0
sgr-...	RDP	TCP	3389	0.0.0.0/0
sgr-...	HTTPS	TCP	443	0.0.0.0/0
sgr-...	Custom TCP	TCP	2222	72.21.196.64/32
sgr-...	SSH	TCP	22	72.21.196.64/32

- f. (Optional) For **FIPS Enabled**, select the **FIPS Enabled endpoint** check box to ensure the endpoint complies with Federal Information Processing Standards (FIPS).


Note

FIPS-enabled endpoints are only available in North American AWS Regions. For available Regions, see [AWS Transfer Family endpoints and quotas](#) in the *AWS General Reference*. For more information about FIPS, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

- g. Choose **Next**.
6. In **Configure additional details**, do the following:
- For **CloudWatch logging**, choose one of the following to enable Amazon CloudWatch logging of your user activity:
 - Create a new role** to allow Transfer Family to automatically create the IAM role, as long as you have the right permissions to create a new role. The IAM role that is created is called `AWSTransferLoggingAccess`.


- **Choose an existing role** to choose an existing IAM role from your account. Under **Logging role**, choose the role. This IAM role should include a trust policy with **Service** set to `transfer.amazonaws.com`.

For more information about CloudWatch logging, see [Configure CloudWatch logging role](#).

 **Note**

- You can't view end-user activity in CloudWatch if you don't specify a logging role.
- If you don't want to set up a CloudWatch logging role, select **Choose an existing role**, but don't select a logging role.

- b. For **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server.

 **Note**

By default, the `TransferSecurityPolicy-2020-06` security policy is attached to your server unless you choose a different one.

For more information about security policies, see [Security policies for AWS Transfer Family servers](#).

- c. (Optional: this section is only for migrating users from an existing SFTP-enabled server.) For **Server Host Key**, enter an RSA, ED25519, or ECDSA private key that will be used to identify your server when clients connect to it over SFTP.
- d. (Optional) For **Tags**, for **Key** and **Value**, enter one or more tags as key-value pairs, and then choose **Add tag**.
- e. Choose **Next**.
7. In **Review and create**, review your choices. If you:
- Want to edit any of them, choose **Edit** next to the step.

Note

You will need to review each step after the step that you chose to edit.

- Have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

It can take a couple of minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations, but you'll need to create a user first. For details on creating users, see [Managing users for server endpoints](#).

Create an internet-facing endpoint for your server

In the following procedure, you create a server endpoint. This endpoint is accessible over the internet only to clients whose source IP addresses are allowed in your VPC's default security group. Additionally, by using Elastic IP addresses to make your endpoint internet-facing, your clients can use the Elastic IP address to allow access to your endpoint in their firewalls.

Note

Only SFTP and FTPS can be used on an internet-facing VPC hosted endpoint.

To create an internet-facing endpoint

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. From the navigation pane, select **Servers**, then choose **Create server**.
3. In **Choose protocols**, select one or more protocols, and then choose **Next**. For more information about protocols, see [Step 2: Create an SFTP-enabled server](#).
4. In **Choose an identity provider**, choose **Service managed** to store user identities and keys in AWS Transfer Family, and then choose **Next**.

Note

This procedure uses the service-managed option. If you choose **Custom**, you provide an Amazon API Gateway endpoint and an AWS Identity and Access Management (IAM) role to access the endpoint. By doing so, you can integrate your directory service to

authenticate and authorize your users. To learn more about working with custom identity providers, see [Working with custom identity providers](#).

5. In **Choose an endpoint**, do the following:

- a. For **Endpoint type**, choose the **VPC hosted** endpoint type to host your server's endpoint.
- b. For **Access**, choose **Internet Facing** to make your endpoint accessible to clients over the internet.

Note

When you choose **Internet Facing**, you can choose an existing Elastic IP address in each subnet or subnets. Or you can go to the VPC console (<https://console.aws.amazon.com/vpc/>) to allocate one or more new Elastic IP addresses. These addresses can be owned either by AWS or by you. You can't associate Elastic IP addresses that are already in use with your endpoint.

- c. (Optional) For **Custom hostname**, choose one of the following:

Note

Customers in AWS GovCloud (US) need to connect via the Elastic IP address directly, or create a hostname record within Commercial Route 53 that points to their EIP. For more information about using Route 53 for GovCloud endpoints, see [Setting up Amazon Route 53 with your AWS GovCloud \(US\) resources](#) in the *AWS GovCloud (US) User Guide*.

- **Amazon Route 53 DNS alias** – if the hostname that you want to use is registered with Route 53. You can then enter the hostname.
- **Other DNS** – if the hostname that you want to use is registered with another DNS provider. You can then enter the hostname.
- **None** – to use the server's endpoint and not use a custom hostname. The server hostname takes the form `server-id.server.transfer.region.amazonaws.com`.

Note

For customers in AWS GovCloud (US), selecting **None** does not create a hostname in this format.

To learn more about working with custom hostnames, see [Working with custom hostnames](#).

- d. For **VPC**, choose an existing VPC ID or choose **Create a VPC** to create a new VPC.
- e. In the **Availability Zones** section, choose up to three Availability Zones and associated subnets. For **IPv4 Addresses**, choose an **Elastic IP address** for each subnet. This is the IP address that your clients can use to allow access to your endpoint in their firewalls.
- f. In the **Security Groups** section, choose an existing security group ID or IDs or choose **Create a security group** to create a new security group. For more information about security groups, see [Security groups for your VPC](#) in the *Amazon Virtual Private Cloud User Guide*. To create a security group, see [Creating a security group](#) in the *Amazon Virtual Private Cloud User Guide*.

Note

Your VPC automatically comes with a default security group. If you don't specify a different security group or groups when you launch the server, we associate the default security group with your server.

For the inbound rules for the security group, you can configure SSH traffic to use port 22, 2222, 22000, or any combination. Port 22 is configured by default. To use port 2222 or port 22000, you add an inbound rule to your security group. For the type, choose **Custom TCP**, then enter either **2222** or **22000** for **Port range**, and for the source, enter the same CIDR range that you have for your SSH port 22 rule.

Note

You can also use port 2223 for clients that require TCP "piggy-back" ACKs, or the ability for the final ack of the TCP 3-way handshake to also contain data.

Some client software may be incompatible with port 2223: for example, a client that requires the server to send the SFTP Identification String before the client does.

The screenshot shows the 'Edit inbound rules' interface in the AWS IAM console. The page title is 'Edit inbound rules' and it includes a sub-header 'Inbound rules control the incoming traffic that's allowed to reach the instance.' Below this is a table of inbound rules. The table has columns for 'Security group rule ID', 'Type', 'Protocol', 'Port range', and 'Source'. The fourth rule is highlighted with a red box. This rule is for 'Custom TCP' on port 2222, with a source IP of 72.21.196.64/32. Other rules include HTTP (port 80), RDP (port 3389), HTTPS (port 443), and SSH (port 22). There is an 'Add rule' button at the bottom left of the table.

Security group rule ID	Type	Protocol	Port range	Source
sg-...	HTTP	TCP	80	Custom 0.0.0.0/0
sg-...	RDP	TCP	3389	Custom 0.0.0.0/0
sg-...	HTTPS	TCP	443	Custom 0.0.0.0/0
sg-...	Custom TCP	TCP	2222	Custom 72.21.196.64/32
sg-...	SSH	TCP	22	Custom 72.21.196.64/32

- g. (Optional) For **FIPS Enabled**, select the **FIPS Enabled endpoint** check box to ensure the endpoint complies with Federal Information Processing Standards (FIPS).

Note

FIPS-enabled endpoints are only available in North American AWS Regions. For available Regions, see [AWS Transfer Family endpoints and quotas](#) in the *AWS General Reference*. For more information about FIPS, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

- h. Choose **Next**.
6. In **Configure additional details**, do the following:
- For **CloudWatch logging**, choose one of the following to enable Amazon CloudWatch logging of your user activity:
 - **Create a new role** to allow Transfer Family to automatically create the IAM role, as long as you have the right permissions to create a new role. The IAM role that is created is called `AWSTransferLoggingAccess`.


- **Choose an existing role** to choose an existing IAM role from your account. Under **Logging role**, choose the role. This IAM role should include a trust policy with **Service** set to `transfer.amazonaws.com`.

For more information about CloudWatch logging, see [Configure CloudWatch logging role](#).

 **Note**

- You can't view end-user activity in CloudWatch if you don't specify a logging role.
- If you don't want to set up a CloudWatch logging role, select **Choose an existing role**, but don't select a logging role.

- b. For **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server.

 **Note**

By default, the `TransferSecurityPolicy-2020-06` security policy is attached to your server unless you choose a different one.

For more information about security policies, see [Security policies for AWS Transfer Family servers](#).

- c. (Optional: this section is only for migrating users from an existing SFTP-enabled server.) For **Server Host Key**, enter an RSA, ED25519, or ECDSA private key that will be used to identify your server when clients connect to it over SFTP.
- d. (Optional) For **Tags**, for **Key** and **Value**, enter one or more tags as key-value pairs, and then choose **Add tag**.
- e. Choose **Next**.
- f. (Optional) For **Managed workflows**, choose workflow IDs (and a corresponding role) that Transfer Family should assume when executing the workflow. You can choose one workflow to execute upon a complete upload, and another to execute upon a partial

upload. To learn more about processing your files by using managed workflows, see [AWS Transfer Family managed workflows](#).

The screenshot displays the 'Managed workflows' configuration page. It is divided into three sections:

- Workflow for complete file uploads:** Includes a dropdown menu with a placeholder 'w-...', a refresh button, and a 'Create a new Workflow' button with an external link icon.
- Workflow for partial file uploads:** Includes a dropdown menu with a placeholder 'w-...', a refresh button, and a 'Create a new Workflow' button with an external link icon.
- Managed workflows execution role:** Includes a dropdown menu with a placeholder '...', a refresh button, and an 'Info' link.

7. In **Review and create**, review your choices. If you:

- Want to edit any of them, choose **Edit** next to the step.

Note

You will need to review each step after the step that you chose to edit.

- Have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

You can choose the server ID to see the detailed settings of the server that you just created. After the column **Public IPv4 address** has been populated, the Elastic IP addresses that you provided are successfully associated with your server's endpoint.

Note

When your server in a VPC is online, only the subnets can be modified and only through the [UpdateServer](#) API. You must [stop the server](#) to add or change the server endpoint's Elastic IP addresses.

Change the endpoint type for your server

If you have an existing server that is accessible over the internet (that is, has a public endpoint type), you can change its endpoint to a VPC endpoint.

Note

If you have an existing server in a VPC displayed as `VPC_ENDPOINT`, we recommend that you modify it to the new VPC endpoint type. With this new endpoint type, you no longer need to use a Network Load Balancer (NLB) to associate Elastic IP addresses with your server's endpoint. Also, you can use VPC security groups to restrict access to your server's endpoint. However, you can continue to use the `VPC_ENDPOINT` endpoint type as needed.

The following procedure assumes that you have a server that uses either the current public endpoint type or the older `VPC_ENDPOINT` type.

To change the endpoint type for your server

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the navigation pane, choose **Servers**.
3. Select the check box of the server that you want to change the endpoint type for.

Important


You must stop the server before you can change its endpoint.

4. For **Actions**, choose **Stop**.
5. In the confirmation dialog box that appears, choose **Stop** to confirm that you want to stop the server.

Note

Before proceeding to the next step, in **Endpoint details**, wait for the **Status** of the server to change to **Offline**; this can take a couple of minutes. You might have to choose **Refresh** on the **Servers** page to see the status change. You won't be able to make any edits until the server is **Offline**.

6. In **Endpoint details**, choose **Edit**.
7. In **Edit endpoint configuration**, do the following:
 - a. For **Edit endpoint type**, choose **VPC hosted**.
 - b. For **Access**, choose one of the following:
 - **Internal** to make your endpoint only accessible to clients using the endpoint's private IP addresses.
 - **Internet Facing** to make your endpoint accessible to clients over the public internet.

 **Note**

When you choose **Internet Facing**, you can choose an existing Elastic IP address in each subnet or subnets. Or, you can go to the VPC console (<https://console.aws.amazon.com/vpc/>) to allocate one or more new Elastic IP addresses. These addresses can be owned either by AWS or by you. You can't associate Elastic IP addresses that are already in use with your endpoint.

- c. (Optional for internet facing access only) For **Custom hostname**, choose one of the following:
 - **Amazon Route 53 DNS alias** – if the hostname that you want to use is registered with Route 53. You can then enter the hostname.
 - **Other DNS** – if the hostname that you want to use is registered with another DNS provider. You can then enter the hostname.
 - **None** – to use the server's endpoint and not use a custom hostname. The server hostname takes the form `serverId.server.transfer.regionId.amazonaws.com`.

To learn more about working with custom hostnames, see [Working with custom hostnames](#).
- d. For **VPC**, choose an existing VPC ID, or choose **Create a VPC** to create a new VPC.
- e. In the **Availability Zones** section, select up to three Availability Zones and associated subnets. If **Internet Facing** is chosen, also choose an Elastic IP address for each subnet.

Note

If you want the maximum of three Availability Zones, but there are not enough available, create them in the VPC console (<https://console.aws.amazon.com/vpc/>). If you modify the subnets or Elastic IP addresses, the server takes a few minutes to update. You can't save your changes until the server update is complete.

- f. Choose **Save**.
8. For **Actions**, choose **Start** and wait for the status of the server to change to **Online**; this can take a couple of minutes.

Note

If you changed a public endpoint type to a VPC endpoint type, notice that **Endpoint type** for your server has changed to **VPC**.

The default security group is attached to the endpoint. To change or add additional security groups, see [Creating Security Groups](#).

Discontinuing the use of VPC_ENDPOINT

AWS Transfer Family is discontinuing the ability to create servers with `EndpointType=VPC_ENDPOINT` for new AWS accounts. As of May 19, 2021, AWS accounts that don't own AWS Transfer Family servers with an endpoint type of `VPC_ENDPOINT` will not be able to create new servers with `EndpointType=VPC_ENDPOINT`. If you already own servers that use the `VPC_ENDPOINT` endpoint type, we recommend that you start using `EndpointType=VPC` as soon as possible. For details, see [Update your AWS Transfer Family server endpoint type from VPC_ENDPOINT to VPC](#).

We launched the new VPC endpoint type earlier in 2020. For more information, see [AWS Transfer Family for SFTP supports VPC Security Groups and Elastic IP addresses](#). This new endpoint is more feature rich and cost effective and there are no PrivateLink charges. For more information, see [AWS PrivateLink pricing](#).

This endpoint type is functionally equivalent to the previous endpoint type (`VPC_ENDPOINT`). You can attach Elastic IP addresses directly to the endpoint to make it internet facing and use security


groups for source IP filtering. For more information, see the [Use IP allow listing to secure your AWS Transfer Family for SFTP servers](#) blog post.

You can also host this endpoint in a shared VPC environment. For more information, see [AWS Transfer Family now supports shared services VPC environments](#).

In addition to SFTP, you can use the VPC EndpointType to enable FTPS and FTP. We don't plan to add these features and FTPS/FTP support to EndpointType=VPC_ENDPOINT. We have also removed this endpoint type as an option from the AWS Transfer Family console.

You can change the endpoint type for your server using the Transfer Family console, AWS CLI, API, SDKs, or AWS CloudFormation. To change your server's endpoint type, see [Updating the AWS Transfer Family server endpoint type from VPC_ENDPOINT to VPC](#).

If you have any questions, contact AWS Support or your AWS account team.

 **Note**

We do not plan to add these features and FTPS or FTP support to EndpointType=VPC_ENDPOINT. We are no longer offering it as an option on the AWS Transfer Family Console.

If you have additional questions, you can contact us through AWS Support or your account team.

Updating the AWS Transfer Family server endpoint type from VPC_ENDPOINT to VPC

You can use the AWS Management Console, AWS CloudFormation, or the Transfer Family API to update a server's EndpointType from VPC_ENDPOINT to VPC. Detailed procedures and examples for using each of these methods to update a server endpoint type are provided in the following sections. If you have servers in multiple AWS regions and in multiple AWS accounts, you can use the example script provided in the following section, with modifications, to identify servers using the VPC_ENDPOINT type that you will need to update.

Topics

- [Identifying servers using the VPC_ENDPOINT endpoint type](#)
- [Updating the server endpoint type using the AWS Management Console](#)

- [Updating the server endpoint type using AWS CloudFormation](#)
- [Updating the server EndpointType using the API](#)

Identifying servers using the VPC_ENDPOINT endpoint type

You can identify which servers are using the VPC_ENDPOINT using the AWS Management Console.

To identify servers using the VPC_ENDPOINT endpoint type using the console

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. Choose **Servers** in the navigation pane to display the list of servers in your account in that region.
3. Sort the list of servers by the **Endpoint type** to see all servers using VPC_ENDPOINT.

To identify servers using VPC_ENDPOINT across multiple AWS Regions and accounts

If you have servers in multiple AWS regions and in multiple AWS accounts, you can use the following example script, with modifications, to identify servers using the VPC_ENDPOINT endpoint type. The example script uses the Amazon EC2 [DescribeRegions](#) and the Transfer Family [ListServers](#) API calls to get a list of the server IDs and regions of all your servers using VPC_ENDPOINT. If you have many AWS accounts, you could loop through your accounts using an IAM Role with read only auditor access if you authenticate using session profiles to your identity provider.

1. Following is a simple example.

```
import boto3

profile = input("Enter the name of the AWS account you'll be working in: ")
session = boto3.Session(profile_name=profile)

ec2 = session.client("ec2")

regions = ec2.describe_regions()

for region in regions['Regions']:
    region_name = region['RegionName']
    if region_name=='ap-northeast-3': #https://github.com/boto/boto3/issues/1943
        continue
```

```
transfer = session.client("transfer", region_name=region_name)
servers = transfer.list_servers()
for server in servers['Servers']:
    if server['EndpointType']=='VPC_ENDPOINT':
        print(server['ServerId'], region_name)
```

2. After you have the list of the servers to update, you can use one of the methods described in the following sections to update the EndpointType to VPC.

Updating the server endpoint type using the AWS Management Console

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the navigation pane, choose **Servers**.
3. Select the check box of the server that you want to change the endpoint type for.

Important

You must stop the server before you can change its endpoint.

4. For **Actions**, choose **Stop**.
5. In the confirmation dialog box that appears, choose **Stop** to confirm that you want to stop the server.

Note

Before proceeding to the next step, wait for the **Status** of the server to change to **Offline**; this can take a couple of minutes. You might have to choose **Refresh** on the **Servers** page to see the status change.

6. After the status changes to **Offline**, choose the server to display the server details page.
7. In the **Endpoint details** section, choose **Edit**.
8. Choose **VPC hosted** for the **Endpoint type**.
9. Choose **Save**
10. For **Actions**, choose **Start** and wait for the status of the server to change to **Online**; this can take a couple of minutes.

Updating the server endpoint type using AWS CloudFormation

This section describes how to use AWS CloudFormation to update a server's EndpointType to VPC. Use this procedure for Transfer Family servers that you have deployed using AWS CloudFormation. In this example, the original AWS CloudFormation template used to deploy the Transfer Family server is shown as follows:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Create AWS Transfer Server with VPC_ENDPOINT endpoint type'
Parameters:
  SecurityGroupId:
    Type: AWS::EC2::SecurityGroup::Id
  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Type: AWS::EC2::VPC::Id
Resources:
  TransferServer:
    Type: AWS::Transfer::Server
    Properties:
      Domain: S3
      EndpointDetails:
        VpcEndpointId: !Ref VPCEndpoint
        EndpointType: VPC_ENDPOINT
        IdentityProviderType: SERVICE_MANAGED
      Protocols:
        - SFTP
  VPCEndpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: com.amazonaws.us-east-1.transfer.server
      SecurityGroupIds:
        - !Ref SecurityGroupId
      SubnetIds:
        - !Select [0, !Ref SubnetIds]
        - !Select [1, !Ref SubnetIds]
        - !Select [2, !Ref SubnetIds]
      VpcEndpointType: Interface
      VpcId: !Ref VpcId
```

The template is updated with the following changes:

- The EndpointType was changed to VPC.

- The `AWS::EC2::VPCEndpoint` resource is removed.
- The `SecurityGroupId`, `SubnetIds`, and `VpcId` were moved to the `EndpointDetails` section of the `AWS::Transfer::Server` resource,
- The `VpcEndpointId` property of `EndpointDetails` was removed.

The updated template looks as follows:

```

AWSTemplateFormatVersion: '2010-09-09'
Description: 'Create AWS Transfer Server with VPC endpoint type'
Parameters:
  SecurityGroupId:
    Type: AWS::EC2::SecurityGroup::Id
  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Type: AWS::EC2::VPC::Id
Resources:
  TransferServer:
    Type: AWS::Transfer::Server
    Properties:
      Domain: S3
      EndpointDetails:
        SecurityGroupIds:
          - !Ref SecurityGroupId
        SubnetIds:
          - !Select [0, !Ref SubnetIds]
          - !Select [1, !Ref SubnetIds]
          - !Select [2, !Ref SubnetIds]
        VpcId: !Ref VpcId
      EndpointType: VPC
      IdentityProviderType: SERVICE_MANAGED
      Protocols:
        - SFTP

```

To update the endpoint type of Transfer Family servers deployed using AWS CloudFormation


1. Stop the server that you want to update using the following steps.
 - a. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
 - b. In the navigation pane, choose **Servers**.

- c. Select the check box of the server that you want to change the endpoint type for.

 **Important**

You must stop the server before you can change its endpoint.

- d. For **Actions**, choose **Stop**.
- e. In the confirmation dialog box that appears, choose **Stop** to confirm that you want to stop the server.

 **Note**

Before proceeding to the next step, wait for the **Status** of the server to change to **Offline**; this can take a couple of minutes. You might have to choose **Refresh** on the **Servers** page to see the status change.

2. Update the CloudFormation stack

- a. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
- b. Choose the stack used to create the Transfer Family server.
- c. Choose **Update**.
- d. Choose **Replace current template**
- e. Upload the new template. CloudFormation Change Sets help you understand how template changes will affect running resources before you implement them. In this example, the Transfer server resource will be modified, and the VPCEndpoint resource will be removed. The VPC endpoint type server creates a VPC Endpoint on your behalf, replacing the original VPCEndpoint resource.

After uploading the new template, the change set will look similar to the following:

Change set preview

Changes (2)

Search changes

Action	Logical ID	Physical ID	Resource type	Replacement
Modify	TransferServer	arn:aws:transfer:us-east-1:364810874344:server/s-6a7d04e12d494ec98	AWS::Transfer::Server	Conditional
Remove	VPCEndpoint	vpce-04e685f8702849573	AWS::EC2::VPCEndpoint	-

- f. Update the stack.
3. Once the stack update is complete, navigate to the Transfer Family management console at <https://console.aws.amazon.com/transfer/>.
4. Restart the server. Choose the server you updated in AWS CloudFormation, and then choose **Start** from the **Actions** menu.

Updating the server EndpointType using the API

You can use the [describe-server](#) AWS CLI command, or the [UpdateServer](#) API command. The following example script stops the Transfer Family server, updates the EndpointType, removes the VPC_ENDPOINT, and starts the server.

```
import boto3
import time

profile = input("Enter the name of the AWS account you'll be working in: ")
region_name = input("Enter the AWS Region you're working in: ")
server_id = input("Enter the AWS Transfer Server Id: ")

session = boto3.Session(profile_name=profile)

ec2 = session.client("ec2", region_name=region_name)
transfer = session.client("transfer", region_name=region_name)

group_ids=[]

transfer_description = transfer.describe_server(ServerId=server_id)
```



```

if transfer_description['Server']['EndpointType']=='VPC_ENDPOINT':
    transfer_vpc_endpoint = transfer_description['Server']['EndpointDetails']
['VpcEndpointId']
    transfer_vpc_endpoint_descriptions =
ec2.describe_vpc_endpoints(VpcEndpointIds=[transfer_vpc_endpoint])
    for transfer_vpc_endpoint_description in
transfer_vpc_endpoint_descriptions['VpcEndpoints']:
        subnet_ids=transfer_vpc_endpoint_description['SubnetIds']
        group_id_list=transfer_vpc_endpoint_description['Groups']
        vpc_id=transfer_vpc_endpoint_description['VpcId']
        for group_id in group_id_list:
            group_ids.append(group_id['GroupId'])
    if transfer_description['Server']['State']=='ONLINE':
        transfer_stop = transfer.stop_server(ServerId=server_id)
        print(transfer_stop)
        time.sleep(300) #safe
        transfer_update =
transfer.update_server(ServerId=server_id,EndpointType='VPC',EndpointDetails={'SecurityGroupId
        print(transfer_update)
        time.sleep(10)
        transfer_start = transfer.start_server(ServerId=server_id)
        print(transfer_start)
        delete_vpc_endpoint =
ec2.delete_vpc_endpoints(VpcEndpointIds=[transfer_vpc_endpoint])

```

Working with custom hostnames

Your *server host name* is the hostname that your users enter in their clients when they connect to your server. You can use a custom domain that you have registered for your server hostname when you work with AWS Transfer Family. For example, you might use a custom hostname like `mysftpserver.mysubdomain.domain.com`.

To redirect traffic from your registered custom domain to your server endpoint, you can use Amazon Route 53 or any Domain Name System (DNS) provider. Route 53 is the DNS service that AWS Transfer Family natively supports.

Topics

- [Use Amazon Route 53 as your DNS provider](#)
- [Use other DNS providers](#)
- [Custom hostnames for non-console created servers](#)

On the console, you can choose one of these options for setting up a custom hostname:

- **Amazon Route 53 DNS alias** – if the hostname that you want to use is registered with Route 53. You can then enter the hostname.
- **Other DNS** – if the hostname that you want to use is registered with another DNS provider. You can then enter the hostname.
- **None** – to use the server's endpoint and not use a custom hostname.

You set this option when you create a new server or edit the configuration of an existing server. For more information about creating a new server, see [Step 2: Create an SFTP-enabled server](#). For more information about editing the configuration of an existing server, see [Edit server details](#).

For more details about using your own domain for the server hostname and how AWS Transfer Family uses Route 53, see the following sections.

Use Amazon Route 53 as your DNS provider

When you create a server, you can use Amazon Route 53 as your DNS provider. Before you use a domain with Route 53, you register the domain. For more information, see [How Domain registration works](#) in the *Amazon Route 53 Developer Guide*.

When you use Route 53 to provide DNS routing to your server, AWS Transfer Family uses the custom hostname that you entered to extract its hosted zone. When AWS Transfer Family extracts a hosted zone, three things can happen:

1. If you're new to Route 53 and don't have a hosted zone, AWS Transfer Family adds a new hosted zone and a CNAME record. The value of this CNAME record is the endpoint hostname for your server. A *CNAME* is an alternate domain name.
2. If you have a hosted zone in Route 53 without any CNAME records, AWS Transfer Family adds a CNAME record to the hosted zone.
3. If the service detects that a CNAME record already exists in the hosted zone, you see an error indicating that a CNAME record already exists. In this case, change the value of the CNAME record to the hostname of your server.

For more information about hosted zones in Route 53, see [Hosted zone](#) in the *Amazon Route 53 Developer Guide*.

Use other DNS providers

When you create a server, you can also use DNS providers other than Amazon Route 53. If you use an alternate DNS provider, make sure that traffic from your domain is directed to your server endpoint.

To do so, set your domain to the endpoint hostname for the server. An endpoint hostname looks like this in the console:

```
serverid.server.transfer.region.amazonaws.com
```

Note

If your server has a VPC endpoint, then the format for the hostname is different from the one described above. To find your VPC endpoint, select the VPC on the server's details page, then select the **VPC endpoint ID** on the VPC dashboard. The endpoint is the first DNS name of those listed.

Custom hostnames for non-console created servers

When you create a server using AWS Cloud Development Kit (AWS CDK), AWS CloudFormation, or through the CLI, you must add a tag if you want that server to have a custom hostname. When you create a Transfer Family server by using the console, the tagging is done automatically.

Note

You also need to create a DNS record to redirect traffic from your domain to your server endpoint. For details, see [Working with records](#) in the *Amazon Route 53 Developer Guide*.

Use the following keys for your custom hostname:

- Add `transfer:customHostname` to display the custom hostname in the console.
- If you are using Route 53 as your DNS provider, add `transfer:route53HostedZoneId`. This tag links the custom hostname to your Route 53 Hosted Zone ID.

To add the custom hostname, issue the following CLI command.

```
aws transfer tag-resource --arn arn:aws:transfer:region:AWS account:server/server-ID --
tags Key=transfer:customHostname,Value="custom-host-name"
```

For example:

```
aws transfer tag-resource --arn arn:aws:transfer:us-east-1:111122223333:server/
s-1234567890abcdef0 --tags Key=transfer:customHostname,Value="abc.example.com"
```

If you are using Route 53, issue the following command to link your custom hostname to your Route 53 Hosted Zone ID.

```
aws transfer tag-resource --arn server-ARN:server/server-ID --tags
Key=transfer:route53HostedZoneId,Value=HOSTED-ZONE-ID
```

For example:

```
aws transfer tag-resource --arn arn:aws:transfer:us-east-1:111122223333:server/
s-1234567890abcdef0 --tags Key=transfer:route53HostedZoneId,Value=ABCDE1111222233334444
```

Assuming the sample values from the previous command, run the following command to view your tags:

```
aws transfer list-tags-for-resource --arn arn:aws:transfer:us-
east-1:111122223333:server/s-1234567890abcdef0
```

```
"Tags": [
  {
    "Key": "transfer:route53HostedZoneId",
    "Value": "/hostedzone/ABCDE1111222233334444"
  },
  {
    "Key": "transfer:customHostname",
    "Value": "abc.example.com"
  }
]
```

Note

Your public, hosted zones and their IDs are available on Amazon Route 53.

Sign in to the AWS Management Console and open the Route 53 console at <https://console.aws.amazon.com/route53/>.

Transferring files over a server endpoint using a client

You transfer files over the AWS Transfer Family service by specifying the transfer operation in a client. AWS Transfer Family supports the following clients:

- We support version 3 of the SFTP protocol.
- OpenSSH (macOS and Linux)

Note

This client works only with servers that are enabled for Secure Shell (SSH) File Transfer Protocol (SFTP).

- WinSCP (Microsoft Windows only)
- Cyberduck (Windows, macOS, and Linux)
- FileZilla (Windows, macOS, and Linux)

The following limitations apply to every client:

- The maximum number of concurrent, multiplexed, SFTP sessions per connection is 10.
- There are two timeout values for SFTP/FTP/FTPS connections. For idle connections, the timeout value is 1800 seconds (30 minutes). If there is no activity after the period has passed the client may be disconnected. There is also a 300 seconds (5 minutes) timeout when a client is completely unresponsive.
- Amazon S3 and Amazon EFS (due to the NFSv4 protocol) require filenames to be in UTF-8 encoding. Using different encoding can lead to unexpected results. For Amazon S3, see [Object key naming guidelines](#).
- For File Transfer Protocol over SSL (FTPS), only Explicit mode is supported. Implicit mode is not supported.
- For File Transfer Protocol (FTP) and FTPS, only Passive mode is supported.
- For FTP and FTPS, only STREAM mode is supported.
- For FTP and FTPS, only Image/Binary mode is supported.

- For FTP and FTPS, TLS - PROT C (unprotected) TLS for the data connection is the default but PROT C is not supported in the AWS Transfer Family FTPS protocol. So for FTPS, you need to issue PROT P for your data operation to be accepted.
- If you are using Amazon S3 for your server's storage, and if your client contains an option to use multiple connections for a single transfer, make sure to disable the option. Otherwise, large file uploads can fail in unpredictable ways. Note that if you are using Amazon EFS as your storage backend, EFS *does* support multiple connections for a single transfer.

The following is a list of available commands for FTP and FTPS:

Available commands					
ABOR	FEAT	MLST	PASS	RETR	STOR
AUTH	LANG	MKD	PASV	RMD	STOU
CDUP	LIST	MODE	PBSZ	RNFR	STRU
CWD	MDTM	NLST	PROT	RNTO	SYST
DELE	MFMT	NOOP	PWD	SIZE	TYPE
EPSV	MLSD	OPTS	QUIT	STAT	USER

Note

APPE is not supported.

For SFTP, the following operations are currently not supported for users that are using the logical home directory on servers that are using Amazon Elastic File System (Amazon EFS).

Unsupported SFTP commands			
SSH_FXP_READLINK	SSH_FXP_SYMLINK	SSH_FXP_STAT when the requested file is a symlink	SSH_FXP_REALPATH when the requested

Unsupported SFTP commands

path contains any symlink components

Generate public-private key pair

Before you can transfer a file, you must have a public-private key pair available. If you have not previously generated a key pair, see [Generate SSH keys for service-managed users](#).

Topics

- [Available SFTP/FTPS/FTP Commands](#)
- [Find your Amazon VPC endpoint](#)
- [Avoid setstat errors](#)
- [Use OpenSSH](#)
- [Use WinSCP](#)
- [Use Cyberduck](#)
- [Use FileZilla](#)
- [Use a Perl client](#)
- [Post upload processing](#)

Available SFTP/FTPS/FTP Commands

The following table describes the available commands for AWS Transfer Family, for the SFTP, FTPS, and FTP protocols.

Note

The table mentions *files* and *directories* for Amazon S3, which only supports buckets and objects: there is no hierarchy. However, you can use prefixes in object key names to imply a hierarchy and organize your data in a way similar to folders. This behavior is described in [Working with object metadata](#) in the *Amazon Simple Storage Service User Guide*.

SFTP/FTPS/FTP Commands

Command	Amazon S3	Amazon EFS
cd	Supported	Supported
chgrp	Not supported	Supported (root or owner only)
chmod	Not supported	Supported (root only)
chmtime	Not supported	Supported
chown	Not supported	Supported (root only)
get	Supported	Supported (including resolving symbolic links)
ln -s	Not supported	Supported
ls/dir	Supported	Supported
mkdir	Supported	Supported
put	Supported	Supported
pwd	Supported	Supported
rename	Supported for files only	Supported
		<div data-bbox="1068 1367 1507 1682" style="border: 1px solid #add8e6; border-radius: 15px; padding: 10px;"> <p>Note</p> <p>Renaming that would overwrite an existing file or directory is not supported.</p> </div>
rm	Supported	Supported

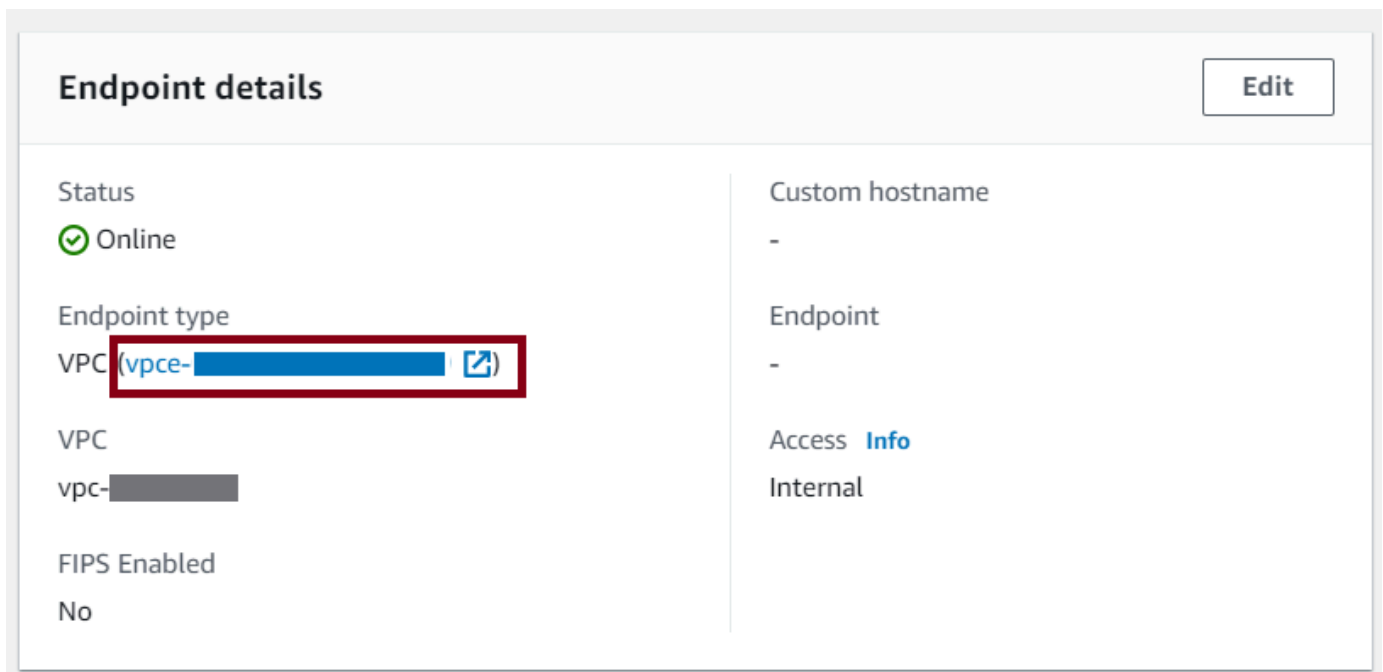
Command	Amazon S3	Amazon EFS
<code>rmdir</code>	Supported (empty directories only)	Supported
<code>version</code>	Supported	Supported

Find your Amazon VPC endpoint

If the endpoint type for your Transfer Family server is VPC, identifying the endpoint to use for transferring files is not straightforward. In this case, use the following procedure to find your Amazon VPC endpoint.

Find your Amazon VPC endpoint

1. Navigate to your server's details page.
2. In the **Endpoint details** pane, select the **VPC**.



3. In the Amazon VPC dashboard, select the **VPC endpoint ID**.
4. In the list of **DNS names**, your server endpoint is the first one listed.

The screenshot shows the AWS Management Console interface for a VPC endpoint. The breadcrumb navigation is 'VPC > Endpoints > vpce-XXXXXXXXXXXX'. The endpoint name is 'vpce-XXXXXXXXXXXX'. The 'Details' section is divided into four columns:

- Endpoint ID:** vpce-XXXXXXXXXXXX
- VPC ID:** vpc-XXXXXXXXXXXX (no-name-specified)
- DNS record IP type:** ipv4
- Status:** Available (with a green checkmark icon)
- Status message:** -
- IP address type:** ipv4
- Creation time:** Monday, April 4, 2022 at 10:51:31 EDT
- Service name:** com.amazonaws.us-east-2.transfer.server.c-0002
- DNS names:** A list of DNS names, with the first one 'vpce-XXXXXXXXXXXX' highlighted by a red box. Other names are partially visible as 'XXXXXXXXXXXX.us-east-2.' and 'XXXXXXXXXXXX.us-east-'.
- Endpoint type:** Interface
- Private DNS names enabled:** No

Avoid setstat errors

Some SFTP file transfer clients can attempt to change the attributes of remote files, including timestamp and permissions, using commands, such as SETSTAT when uploading the file. However, these commands are not compatible with object storage systems, such as Amazon S3. Due to this incompatibility, file uploads from these clients can result in errors even when the file is otherwise successfully uploaded.

- When you call the `CreateServer` or `UpdateServer` API, use the `ProtocolDetails` option `SetStatOption` to ignore the error that is generated when the client attempts to use SETSTAT on a file you are uploading to an S3 bucket.
- Set the value to `ENABLE_NO_OP` to have the Transfer Family server ignore the SETSTAT command, and upload files without needing to make any changes to your SFTP client.
- Note that while the `SetStatOption` `ENABLE_NO_OP` setting ignores the error, it *does* generate a log entry in CloudWatch Logs, so you can determine when the client is making a SETSTAT call.

For the API details for this option, see [ProtocolDetails](#).

Use OpenSSH

Use the instructions that follow to transfer files from the command line using OpenSSH.

Note

This client works only with an SFTP-enabled server.

To transfer files over AWS Transfer Family using the OpenSSH command line utility

1. On Linux, macOS, or Windows, open a command terminal.
2. At the prompt, enter the following command:

```
sftp -i transfer-key sftp_user@service_endpoint
```

In the preceding command, *sftp_user* is the username and *transfer-key* is the SSH private key. Here, *service_endpoint* is the server's endpoint as shown in the AWS Transfer Family console for the selected server.

Note

This command uses settings that are in the default `ssh_config` file. Unless you have previously edited this file, SFTP uses port 22. You can specify a different port (for example 2222) by adding a `-P` flag to the command, as follows.

```
sftp -P 2222 -i transfer-key sftp_user@service_endpoint
```

Alternatively, if you always want to use port 2222 or port 22000, you can update your default port in your `ssh_config` file.

An `sftp` prompt should appear.

3. (Optional) To view the user's home directory, enter the following command at the `sftp` prompt:

```
pwd
```

4. To upload a file from your file system to the Transfer Family server, use the `put` command. For example, to upload `hello.txt` (assuming that file is in your current directory on your file system), run the following command at the `sftp` prompt:

```
put hello.txt
```

A message similar to the following appears, indicating that the file transfer is in progress, or complete.

```
Uploading hello.txt to /my-bucket/home/sftp_user/hello.txt
```

```
hello.txt 100% 127 0.1KB/s 00:00
```

Note

After your server is created, it can take a few minutes for the server endpoint hostname to be resolvable by the DNS service in your environment.

Use WinSCP

Use the instructions that follow to transfer files from the command line using WinSCP.

Note

If you are using WinSCP 5.19, you can directly connect to Amazon S3 using your AWS credentials and upload/download files. For more details, see [Connecting to Amazon S3 service](#).


To transfer files over AWS Transfer Family using WinSCP

1. Open the WinSCP client.
2. In the **Login** dialog box, for **File protocol**, choose a protocol: **SFTP** or **FTP**.

If you chose FTP, for **Encryption**, choose one of the following:


- **No encryption** for FTP
- **TLS/SSL Explicit encryption** for FTPS

3. For **Host name**, enter your server endpoint. The server endpoint is located on the **Server details** page. For more information, see [View SFTP, FTPS, and FTP server details](#).

 **Note**

If your server uses a VPC endpoint, see [Find your Amazon VPC endpoint](#).


4. For **Port number**, enter the following:
 - **22** for SFTP
 - **21** for FTP/FTPS
5. For **User name**, enter the name for the user that you created for your specific identity provider.

 **Note**

The username should be one of the users you created or configured for your identity provider. AWS Transfer Family provides the following identity providers:

- [Working with service-managed users](#)
- [Using AWS Directory Service for Microsoft Active Directory](#)
- [Working with custom identity providers](#)

6. Choose **Advanced** to open the **Advanced Site Settings** dialog box. In the **SSH** section, choose **Authentication**.
7. For **Private key file**, browse for and choose the SSH private key file from your file system.

 **Note**

If WinSCP offers to convert your SSH private key to the PPK format, choose **OK**.

8. Choose **OK** to return to the **Login** dialog box, and then choose **Save**.
9. In the **Save session as site** dialog box, choose **OK** to complete your connection setup.
10. In the **Login** dialog box, choose **Tools**, and then choose **Preferences**.
11. In the **Preferences** dialog box, for **Transfer**, choose **Endurance**.

For the **Enable transfer resume/transfer to temporary filename for** option, choose **Disable**.

Note

If you leave this option enabled, it increases upload costs, substantially decreasing upload performance. It also can lead to failures of large file uploads.

12. For **Transfer**, choose **Background**, and clear the **Use multiple connections for single transfer** check box.

Note

If you leave this option selected, large file uploads can fail in unpredictable ways. For example, orphaned multipart uploads that incur Amazon S3 charges can be created. Silent data corruption can also occur.

13. Perform your file transfer.

You can use drag-and-drop methods to copy files between the target and source windows. You can use toolbar icons to upload, download, delete, edit, or modify the properties of files in WinSCP.

Note

This note does not apply if you are using Amazon EFS for storage. Commands that attempt to change attributes of remote files, including timestamps, are not compatible with object storage systems such as Amazon S3. Therefore, if you are using Amazon S3 for storage, be sure to disable WinSCP timestamp settings (or use the `SetStatOption` as described in [Avoid setstat errors](#)) before you perform file transfers. To do so, in the **WinSCP Transfer settings** dialog box, disable the **Set permissions** upload option and the **Preserve timestamp** common option.

Use Cyberduck

Use the instructions that follow to transfer files from the command line using Cyberduck.

To transfer files over AWS Transfer Family using Cyberduck

1. Open the [Cyberduck](#) client.
2. Choose **Open Connection**.
3. In the **Open Connection** dialog box, choose a protocol: **SFTP (SSH File Transfer Protocol)**, **FTP-SSL (Explicit AUTH TLS)**, or **FTP (File Transfer Protocol)**.
4. For **Server**, enter your server endpoint. The server endpoint is located on the **Server details** page. For more information, see [View SFTP, FTPS, and FTP server details](#).

Note

If your server uses a VPC endpoint, see [Find your Amazon VPC endpoint](#).

5. For **Port number**, enter the following:
 - **22** for SFTP
 - **21** for FTP/FTPS
6. For **Username**, enter the name for the user that you created in [Managing users for server endpoints](#).
7. If SFTP is selected, for **SSH Private Key**, choose or enter the SSH private key.
8. Choose **Connect**.
9. Perform your file transfer.

Depending on where your files are, do one of the following:

- In your local directory (the source), choose the files that you want to transfer, and drag and drop them into the Amazon S3 directory (the target).
- In the Amazon S3 directory (the source), choose the files that you want to transfer, and drag and drop them into your local directory (the target).

Use FileZilla

Use the instructions that follow to transfer files using FileZilla.


To set up FileZilla for a file transfer

1. Open the FileZilla client.

2. Choose **File**, and then choose **Site Manager**.
3. In the **Site Manager** dialog box, choose **New site**.
4. On the **General** tab, for **Protocol**, choose a protocol: **SFTP** or **FTP**.

If you chose FTP, for **Encryption**, choose one of the following:

- **Only use plain FTP (insecure)** – for FTP
 - **Use explicit FTP over TLS if available** – for FTPS
5. For **Host name**, enter the protocol that you are using, followed by your server endpoint. The server endpoint is located on the **Server details** page. For more information, see [View SFTP, FTPS, and FTP server details](#).

 **Note**

If your server uses a VPC endpoint, see [Find your Amazon VPC endpoint](#).

- If you are using SFTP, enter: `sftp://hostname`
- If you are using FTPS, enter: `ftps://hostname`

Make sure to replace *hostname* with your actual server endpoint.

6. For **Port number**, enter the following:
 - **22** for SFTP
 - **21** for FTP/FTPS
7. If SFTP is selected, for **Logon Type**, choose **Key file**.

For **Key file**, choose or enter the SSH private key.
8. For **User**, enter the name for the user that you created in [Managing users for server endpoints](#).
9. Choose **Connect**.
10. Perform your file transfer.

Note

If you interrupt a file transfer in progress, AWS Transfer Family might write a partial object in your Amazon S3 bucket. If you interrupt an upload, check that the file size in the Amazon S3 bucket matches the file size of the source object before continuing.

Use a Perl client

If you use the `Net::SFTP::Foreign` perl client, you must set the `queue_size` to 1. For example:

```
my $sftp = Net::SFTP::Foreign->new('user@s-12345.server.transfer.us-east-2.amazonaws.com', queue_size => 1);
```

Note

This workaround is needed for revisions of `Net::SFTP::Foreign` prior to [1.92.02](#).

Post upload processing

You can view post upload processing information including Amazon S3 object metadata and event notifications.

Topics

- [Amazon S3 object metadata](#)
- [Amazon S3 event notifications](#)

Amazon S3 object metadata

As a part of your object's metadata you see a key called `x-amz-meta-user-agent` whose value is `AWSTransfer` and `x-amz-meta-user-agent-id` whose value is `username@server-id`. The `username` is the Transfer Family user who uploaded the file and `server-id` is the server used for the upload. This information can be accessed using the [HeadObject](#) operation on the S3 object inside your Lambda function.

Key	Value
<input type="radio"/> Content-Type	text/plain
<input type="radio"/> x-amz-meta-user-agent	AWSTransfer
<input type="radio"/> x-amz-meta-user-agent-id	sftp_user@s-

Amazon S3 event notifications

When an object is uploaded to your S3 bucket using Transfer Family, RoleSessionName is contained in the Requester field in the [S3 event notification structure](#) as [AWS:Role Unique Identifier]/username.sessionid@server-id. For example, the following are the contents for a sample Requester field from an S3 access log for a file that was copied to the S3 bucket.

```
arn:aws:sts::AWS-Account-ID:assumed-role/IamRoleName/
username.sessionid@server-id
```

In the Requester field above, it shows the IAM Role called IamRoleName. For more information about configuring S3 event notifications, see [Configuring Amazon S3 event notifications](#) in the *Amazon Simple Storage Service Developer Guide*. For more information about AWS Identity and Access Management (IAM) role unique identifiers, see [Unique identifiers](#) in the *AWS Identity and Access Management User Guide*.

Managing users for server endpoints

In the following sections, you can find information about how to add users using AWS Transfer Family, AWS Directory Service for Microsoft Active Directory or a custom identity provider.

If you use a service-managed identity type, you add users to your file transfer protocol enabled server. When you do so, each username must be unique on your server.

As part of each user's properties, you also store that user's Secure Shell (SSH) public key. Doing so is required for key based authentication, which this procedure uses. The private key is stored locally

on your user's computer. When your user sends an authentication request to your server by using a client, your server first confirms that the user has access to the associated SSH private key. The server then successfully authenticates the user.

In addition, you specify a user's home directory, or landing directory, and assign an AWS Identity and Access Management (IAM) role to the user. Optionally, you can provide a session policy to limit user access only to the home directory of your Amazon S3 bucket.

Important

AWS Transfer Family blocks usernames that are 1 or 2 characters long from authenticating to SFTP servers. Additionally, we also block the `rootuser` name.

The reason behind this is due to the large volume of malicious login attempts by password scanners.

Amazon EFS vs. Amazon S3

Characteristics of each storage option:

- To limit access: Amazon S3 supports session policies; Amazon EFS supports POSIX user, group, and secondary group IDs
- Both support public/private keys
- Both support home directories
- Both support logical directories

Note

For Amazon S3, most of the support for logical directories is via API/CLI. You can use the **Restricted** check box in the console to lock down a user to their home directory, but you cannot specify a virtual directory structure.

Logical directories

If you are specifying logical directory values for your user, the parameter you use depends on the type of user.

- For service-managed users, provide logical directory values in `HomeDirectoryMappings`.

- For custom identity provider users, provide logical directory values in `HomeDirectoryDetails`.

Topics

- [Working with service-managed users](#)
- [Using AWS Directory Service for Microsoft Active Directory](#)
- [Using AWS Directory Service for Azure Active Directory Domain Services](#)
- [Working with custom identity providers](#)

Working with service-managed users

You can add either Amazon S3 or Amazon EFS service-managed users to your server, depending on the server's **Domain** setting. For more information, see [Configuring an SFTP, FTPS, or FTP server endpoint](#).

To add a service-managed user programmatically, see the [example](#) for the [CreateUser](#) API.

Note

For service-managed users there is a limit of 2,000 logical directory entries. For information about using logical directories see [Using logical directories to simplify your Transfer Family directory structures](#).

Topics

- [Adding Amazon S3 service-managed users](#)
- [Adding Amazon EFS service-managed users](#)
- [Managing service-managed users](#)

Adding Amazon S3 service-managed users

Note

If you want to configure a cross account Amazon S3 bucket, follow the steps mentioned in this Knowledge Center article: [How do I configure my AWS Transfer Family server to use an Amazon Simple Storage Service bucket that's in another AWS account?](#)

To add an Amazon S3 service-managed user to your server

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>, then select **Servers** from the navigation pane.
2. On the **Servers** page, select the check box of the server that you want to add a user to.
3. Choose **Add user**.
4. In the **User configuration** section, for **Username**, enter the username. This username must be a minimum of 3 and a maximum of 100 characters. You can use the following characters in the username: a–z, A–Z, 0–9, underscore '_', hyphen '-', period '.', and at sign "@". The username can't start with a hyphen '-', period '.', or at sign "@".
5. For **Access**, choose the IAM role that you previously created that provides access to your Amazon S3 bucket.

You created this IAM role using the procedure in [Create an IAM role and policy](#). That IAM role includes an IAM policy that provides access to your Amazon S3 bucket. It also includes a trust relationship with the AWS Transfer Family service, defined in another IAM policy. If you need fine-grained access control for your users, refer to the [Enhance data access control with AWS Transfer Family and Amazon S3](#) blog post.

6. (Optional) For **Policy**, select one of the following:
 - **None**
 - **Existing policy**
 - **Select a policy from IAM:** allows you to choose an existing session policy. Choose **View** to see a JSON object containing the details of the policy.
 - **Auto-generate policy based on home folder:** generates a session policy for you. Choose **View** to see a JSON object containing the details of the policy.


Note

If you choose **Auto-generate policy based on home folder**, do not select **Restricted** for this user.

To learn more about session policies, see [Create an IAM role and policy](#). To learn more about creating a session policy, see [Creating a session policy for an Amazon S3 bucket](#).

7. For **Home directory**, choose the Amazon S3 bucket to store the data to transfer using AWS Transfer Family. Enter the path to the home directory where your user lands when they log in using their client.

If you keep this parameter blank, the root directory of your Amazon S3 bucket is used. In this case, make sure that your IAM role provides access to this root directory.

 **Note**

We recommend that you choose a directory path that contains the user name of the user, which enables you to effectively use a session policy. The session policy limits user access in the Amazon S3 bucket to that user's home directory.

8. (Optional) For **Restricted**, select the check box so that your users can't access anything outside of that folder and can't see the Amazon S3 bucket or folder name.

 **Note**

Assigning the user a home directory and restricting the user to that home directory should be sufficient to lock down the user's access to the designated folder. If you need to apply further controls, use a session policy.

If you select **Restricted** for this user, you cannot select **Auto-generate policy based on home folder**, because home folder is not a defined value for Restricted users.

9. For **SSH public key**, enter the public SSH key portion of the SSH key pair.

Your key is validated by the service before you can add your new user.

 **Note**

For instructions on how to generate an SSH key pair, see [Generate SSH keys for service-managed users](#).

10. (Optional) For **Key** and **Value**, enter one or more tags as key-value pairs, and choose **Add tag**.
11. Choose **Add** to add your new user to the server that you chose.

The new user appears in the **Users** section of the **Server details** page.

Next steps – For the next step, continue on to [Transferring files over a server endpoint using a client](#).

Adding Amazon EFS service-managed users

Amazon EFS uses the Portable Operating System Interface (POSIX) file permission model to represent file ownership.

- For more details on Amazon EFS file ownership, see [Amazon EFS file ownership](#).
- For more details on setting up directories for your EFS users, see [Set up Amazon EFS users for Transfer Family](#).

To add an Amazon EFS service-managed user to your server

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>, then select **Servers** from the navigation pane.
2. On the **Servers** page, select the Amazon EFS server that you want to add a user to.
3. Choose **Add user** to display the **Add user** page.
4. In the **User configuration** section, use the following settings.
 - a. The **Username**, must be a minimum of 3 and a maximum of 100 characters. You can use the following characters in the username: a–z, A–Z, 0–9, underscore '_', hyphen '-', period '.', and at sign "@". The username can't start with a hyphen '-', period '.', or at sign "@".
 - b. For **User ID** and **Group ID**, note the following:
 - For the first user that you create, we recommend that you enter a value of **0** for both **Group ID** and **User ID**. This grants the user administrator privileges for Amazon EFS.
 - For additional users, enter the user's POSIX user ID and group ID. These IDs are used for all Amazon Elastic File System operations performed by the user.
 - For **User ID** and **Group ID**, do not use any leading zeroes. For example, **12345** is acceptable, **012345** is not.
 - c. (Optional) For **Secondary Group IDs**, enter one or more additional POSIX group IDs for each user, separated by commas.
 - d. For **Access**, choose the IAM role that:
 - Gives the user access to only the Amazon EFS resources (file systems) that you want them to access.

- Defines which file system operations that the user can and cannot perform.

We recommend that you use the IAM role for Amazon EFS file system selection with mount access and read/write permissions. For example, the combination of the following two AWS managed policies, while quite permissive, grants the necessary permissions for your user:

- AmazonElasticFileSystemClientFullAccess
- AWSTransferConsoleFullAccess

For more information, see the blog post [AWS Transfer Family support for Amazon Elastic File System](#).

e. For **Home directory**, do the following:

- Choose the Amazon EFS file system that you want to use for storing the data to transfer using AWS Transfer Family.
- Decide whether to set the home directory to **Restricted**. Setting the home directory to **Restricted** has the following effects:
 - Amazon EFS users can't access any files or directories outside of that folder.
 - Amazon EFS users can't see the Amazon EFS file system name (**fs-xxxxxxx**).

 **Note**

When you select the **Restricted** option, symlinks don't resolve for Amazon EFS users.

- (Optional) Enter the path to the home directory that you want users to be in when they log in using their client.

If you don't specify a home directory, the root directory of your Amazon EFS file system is used. In this case, make sure that your IAM role provides access to this root directory.

5. For **SSH public key**, enter the public SSH key portion of the SSH key pair.

Your key is validated by the service before you can add your new user.

Note

For instructions on how to generate an SSH key pair, see [Generate SSH keys for service-managed users](#).

6. (Optional) Enter any tags for the user. For **Key** and **Value**, enter one or more tags as key-value pairs, and choose **Add tag**.
7. Choose **Add** to add your new user to the server that you chose.

The new user appears in the **Users** section of the **Server details** page.

Issues that you might encounter when you first SFTP to your Transfer Family server:

- If you run the `sftp` command and the prompt doesn't appear, you might encounter the following message:

```
Couldn't canonicalize: Permission denied
```

```
Need cwd
```

In this case, you must increase the policy permissions for your user's role. You can add an AWS managed policy, such as `AmazonElasticFileSystemClientFullAccess`.

- If you enter `pwd` at the `sftp` prompt to view the user's home directory, you might see the following message, where `USER-HOME-DIRECTORY` is the home directory for the SFTP user:

```
remote readdir("/USER-HOME-DIRECTORY"): No such file or directory
```

In this case, you should be able to navigate to the parent directory (`cd ..`), and create the user's home directory (`mkdir username`).

Next steps – For the next step, continue on to [Transferring files over a server endpoint using a client](#).

Managing service-managed users

In this section, you can find information about how to view a list of users, how to edit user details, and how to add an SSH public key.

- [View a list of users](#)
- [View or edit user details](#)
- [Delete a user](#)
- [Add SSH public key](#)
- [Delete SSH public key](#)

To find a list of your users

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. Select **Servers** from the navigation pane to display the **Servers** page.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.
4. Under **Users**, view a list of users.

To view or edit user details

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. Select **Servers** from the navigation pane to display the **Servers** page.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.
4. Under **Users**, choose a username to see the **User details** page.

You can change the user's properties on this page by choosing **Edit**.

5. On the **Users details** page, choose **Edit** next to **User configuration**.

- On the **Edit configuration** page, for **Access**, choose the IAM role that you previously created that provides access to your Amazon S3 bucket.

You created this IAM role using the procedure in [Create an IAM role and policy](#). That IAM role includes an IAM policy that provides access to your Amazon S3 bucket. It also includes a trust relationship with the AWS Transfer Family service, defined in another IAM policy.

- (Optional) For **Policy**, choose one of the following:
 - None**
 - Existing policy**
 - Select a policy from IAM** to choose an existing policy. Choose **View** to see a JSON object containing the details of the policy.

To learn more about session policies, see [Create an IAM role and policy](#). To learn more about creating a session policy, see [Creating a session policy for an Amazon S3 bucket](#).

- For **Home directory**, choose the Amazon S3 bucket to store the data to transfer using AWS Transfer Family. Enter the path to the home directory where your user lands when they log in using their client.

If you leave this parameter blank, the `root` directory of your Amazon S3 bucket is used. In this case, make sure that your IAM role provides access to this `root` directory.

Note

We recommend that you choose a directory path that contains the user name of the user, which enables you to effectively use a session policy. The session policy limits user access in the Amazon S3 bucket to that user's home directory.

9. (Optional) For **Restricted**, select the check box so that your users can't access anything outside of that folder and can't see the Amazon S3 bucket or folder name.

Note

When assigning the user a home directory and restricting the user to that home directory, this should be sufficient enough to lock down the user's access to the designated folder. Use a session policy when you need to apply further controls.

10. Choose **Save** to save your changes.

To delete a user


1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. Select **Servers** from the navigation pane to display the **Servers** page.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.
4. Under **Users**, choose a username to see the **User details** page.
5. On the **Users details** page, choose **Delete** to the right of the username.
6. In the confirmation dialog box that appears, enter the word **delete**, and then choose **Delete** to confirm that you want to delete the user.

The user is deleted from the **users** list.

To add an SSH public key for a user

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the navigation pane, choose **Servers**.

3. Choose the identifier in the **Server ID** column to see the **Server details** page.
4. Under **Users**, choose a username to see the **User details** page.
5. Choose **Add SSH public key** to add a new SSH public key to a user.

 **Note**

SSH keys are used only by servers that are enabled for Secure Shell (SSH) File Transfer Protocol (SFTP). For information about how to generate an SSH key pair, see [Generate SSH keys for service-managed users](#).

6. For **SSH public key**, enter the SSH public key portion of the SSH key pair.

Your key is validated by the service before you can add your new user. The format of the SSH key is `ssh-rsa string`. To generate an SSH key pair, see [Generate SSH keys for service-managed users](#).

7. Choose **Add key**.

To delete an SSH public key for a user

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the navigation pane, choose **Servers**.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.
4. Under **Users**, choose a username to see the **User details** page.
5. To delete a public key, select its SSH key check box and choose **Delete**.

Using AWS Directory Service for Microsoft Active Directory

You can use AWS Transfer Family to authenticate your file transfer end users using AWS Directory Service for Microsoft Active Directory. It enables seamless migration of file transfer workflows that rely on Active Directory authentication without changing end users' credentials or needing a custom authorizer.

With AWS Managed Microsoft AD, you can securely provide AWS Directory Service users and groups access over SFTP, FTPS, and FTP for data stored in Amazon Simple Storage Service (Amazon S3) or Amazon Elastic File System (Amazon EFS). If you use Active Directory to store your users' credentials, you now have an easier way to enable file transfers for these users.

You can provide access to Active Directory groups in AWS Managed Microsoft AD in your on-premises environment or in the AWS Cloud using Active Directory connectors. You can give users that are already configured in your Microsoft Windows environment, either in the AWS Cloud or in their on-premises network, access to an AWS Transfer Family server that uses AWS Managed Microsoft AD for identity.

Note

- AWS Transfer Family does not support Simple AD.
- Transfer Family does not support cross-region Active Directory configurations: we only support Active Directory integrations that are in the same region as that of the Transfer Family server.
- Transfer Family does not support using either AWS Managed Microsoft AD or AD Connector to enable multi-factor authentication (MFA) for your existing RADIUS-based MFA infrastructure.
- AWS Transfer Family does not support replicated regions of Managed Active Directory.

To use AWS Managed Microsoft AD, you must perform the following steps:

1. Create one or more AWS Managed Microsoft AD directories using the AWS Directory Service console.
2. Use the Transfer Family console to create a server that uses AWS Managed Microsoft AD as its identity provider.
3. Set up AWS Directory using an Active Directory Connector.
4. Add access from one or more of your AWS Directory Service groups.
5. Although not required, we recommend that you test and verify user access.

Topics

- [Before you start using AWS Directory Service for Microsoft Active Directory](#)
- [Working with Active Directory realms](#)
- [Choosing AWS Managed Microsoft AD as your identity provider](#)
- [Connecting to on-prem Microsoft Active Directory](#)
- [Granting access to groups](#)

- [Testing users](#)
- [Deleting server access for a group](#)
- [Connecting to the server using SSH \(Secure Shell\)](#)
- [Connecting AWS Transfer Family to a self-managed Active Directory using forests and trusts](#)

Before you start using AWS Directory Service for Microsoft Active Directory

Provide a unique identifier for your AD groups

Before you can use AWS Managed Microsoft AD, you must provide a unique identifier for each group in your Microsoft AD directory. You can use the security identifier (SID) for each group to do this. The users of the group that you associate have access to your Amazon S3 or Amazon EFS resources over the enabled protocols using AWS Transfer Family.

Use the following Windows PowerShell command to retrieve the SID for a group, replacing *YourGroupName* with the name of the group.

```
Get-ADGroup -Filter {samAccountName -like "YourGroupName*"} -Properties * | Select SamAccountName, ObjectSid
```

Note

If you are using AWS Directory Service as your identity provider, and if `userPrincipalName` and `SamAccountName` have different values, AWS Transfer Family accepts the value in `SamAccountName`. Transfer Family does not accept the value specified in `userPrincipalName`.

Add AWS Directory Service permissions to your role

You also need AWS Directory Service API permissions to use AWS Directory Service as your identity provider. The following permissions are required or suggested:

- `ds:DescribeDirectories` is required for Transfer Family to look up the directory
- `ds:AuthorizeApplication` is required to add authorization for Transfer Family
- `ds:UnauthorizeApplication` is suggested to remove any resources that are provisionally created, in case something goes wrong during the server creation process

Add these permissions to the role you are using for creating your Transfer Family servers. For more details on these permissions, see [AWS Directory Service API permissions: Actions, resources, and conditions reference](#).

Working with Active Directory realms

When you are considering how to have your Active Directory users access AWS Transfer Family servers, keep in mind the user's realm, and their group's realm. Ideally, the user's realm and their group's realm should match. That is, both the user and the group are in the default realm, or both are in the trusted realm. If this is not the case, the user cannot be authenticated by Transfer Family.

You can test the user to ensure the configuration is correct. For details, see [Testing users](#). If there is a problem with the user/group realm, you receive the error, No associated access found for user's groups.

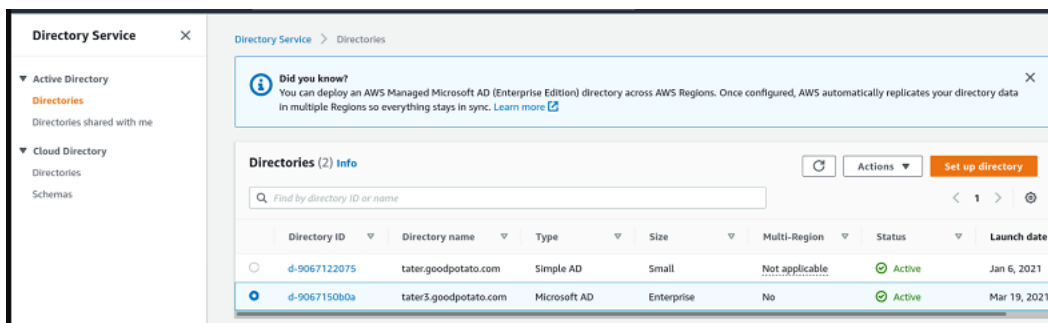
Choosing AWS Managed Microsoft AD as your identity provider

This section describes how to use AWS Directory Service for Microsoft Active Directory with a server.

To use AWS Managed Microsoft AD with Transfer Family

1. Sign in to the AWS Management Console and open the AWS Directory Service console at <https://console.aws.amazon.com/directoryservicev2/>.

Use the AWS Directory Service console to configure one or more managed directories. For more information, see [AWS Managed Microsoft AD](#) in the *AWS Directory Service Admin Guide*.



2. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>, and choose **Create server**.
3. On the **Choose protocols** page, choose one or more protocols from the list.

Note

If you select **FTPS**, you must provide the AWS Certificate Manager certificate.

4. For **Choose an identity provider**, choose **AWS Directory Service**.

Choose an identity provider

Identity provider

Identity provider type
An identity provider manages user access for authentication and authorization

Service managed
Create and manage users within the service

AWS Directory Service [Info](#)
Enable users in AWS Managed AD or use your own self-managed AD in your on-premises environment or in AWS

Custom Identity Provider [Info](#)
Manage users by integrating an identity provider of your choice

Directory

TATER3 ▼

Cancel Previous **Next**

5. The **Directory** list contains all the managed directories that you have configured. Choose a directory from the list, and choose **Next**.

Note

- Cross-Account and Shared directories are not supported for AWS Managed Microsoft AD.
- To set up a server with Directory Service as your identity provider, you need to add some AWS Directory Service permissions. For details, see [Before you start using AWS Directory Service for Microsoft Active Directory](#).

6. To finish creating the server, use one of the following procedures:
 - [Create an SFTP-enabled server](#)
 - [Create an FTPS-enabled server](#)

- [Create an FTP-enabled server](#)

In those procedures, continue with the step that follows choosing an identity provider.

Important

You can't delete a Microsoft AD directory in AWS Directory Service if you used it in a Transfer Family server. You must delete the server first, and then you can delete the directory.

Connecting to on-prem Microsoft Active Directory

This section describes how to set up an AWS Directory using an AD Connector

To set up your AWS Directory using AD Connector

1. Open the [Directory Service](#) console and select **Directories**.
2. Select **Set up directory**.
3. For directory type, choose **AD Connector**.
4. Select a directory size, select **Next**, then select your VPC and Subnets.
5. Select **Next**, then fill in the fields as follows:
 - **Directory DNS name:** enter the domain name you are using for your Microsoft Active Directory.
 - **DNS IP addresses:** enter you Microsoft Active Directory IP addresses.
 - **Server account username and password:** enter the details for the service account to use.
6. Complete the screens to create the directory service.

The next step is to create a Transfer Family server with the SFTP protocol, and the identity provider type of **AWS Directory Service**. From **Directory** drop down list, select the directory you added in the previous procedure.

Granting access to groups

After you create the server, you must choose which groups in the directory should have access to upload and download files over the enabled protocols using AWS Transfer Family. You do this by creating an *access*.

Note

Users must belong *directly* to the group to which you are granting access. For example, assume that Bob is a user and belongs to groupA, and groupA itself is included in groupB.

- If you grant access to groupA, Bob is granted access.
- If you grant access to groupB (and not to groupA), Bob does not have access.

To grant access to a group

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. Navigate to your server details page.
3. In the **Accesses** section, choose **Add access**.
4. Enter the SID for the AWS Managed Microsoft AD directory that you want to have access to this server.

Note

For information about how to find the SID for your group, see [the section called "Before you start using AWS Directory Service for Microsoft Active Directory"](#).

5. For **Access**, choose an AWS Identity and Access Management (IAM) role for the group.
6. In the **Policy** section, choose a policy. The default setting is **None**.
7. For **Home directory**, choose an Amazon S3 bucket that corresponds to the group's home directory.

Note

You can limit the portions of the bucket that users see by creating a session policy. For example, to limit users to their own folder under the `/filetest` directory, enter the following text in the box.

```
/filetest/${transfer:UserName}
```

To learn more about creating a session policy, see [Creating a session policy for an Amazon S3 bucket](#).

8. Choose **Add** to create the association.
9. Choose your server.
10. Choose **Add access**.
 - Enter the SID for the group.

Note

For information about how to find the SID, see [the section called “Before you start using AWS Directory Service for Microsoft Active Directory”](#).

11. Choose **Add access**.

In the **Accesses** section, the accesses for the server are listed.

Endpoint configuration

Availability Zone	Subnet ID	Private IPv4 Address
us-east-1a	subnet-...	172.31.80.36

Accesses (1)

Search:

Actions ▼ Associate access

<input checked="" type="checkbox"/>	External Id	Home directory	Role
<input checked="" type="checkbox"/>	S-...	/padbucket3	ADGuy_S3_And_EFS

Additional details Edit

Logging role Info Server activity not logged to Amazon CloudWatch	Security Policy Info TransferSecurityPolicy-2018-11
Server host key Info [Redacted]	Domain Amazon S3

Testing users

You can test whether a user has access to the AWS Managed Microsoft AD directory for your server.

Note

A user must be in exactly one group (an external ID) that is listed in the **Access** section of the **Endpoint configuration** page. If the user is in no groups, or is in more than a single group, that user is not granted access.

To test whether a specific user has access

1. On the server details page, choose **Actions**, and then choose **Test**.
2. For **Identity provider testing**, enter the sign-in credentials for a user that is in one of the groups that has access.
3. Choose **Test**.

You see a successful identity provider test, showing that the selected user has been granted access to the server.

Identity provider testing

User configuration [Info](#)

Username Password

Response

```
{
  "Response": {
    "homeDirectory": "\\\\padbucket3", "homeDirectoryDetails": null, "homeDirectoryType": "PATH", "posixProfile":
    null, "publicKeys": null, "role": "arn:aws:iam::195886157073:role/WDGuy_53_Ard_EFS", "policy": null, "userName":
    "transferuser1", "identityProviderType": null, "userConfigMessage": null,
    "StatusCode": 200,
    "Message": ""
  }
}
```

Cancel Test

If the user belongs to more than one group that has access, you receive the following response.

```
"Response": "",
"StatusCode": 200,
"Message": "More than one associated access found for user's groups."
```

Deleting server access for a group

To delete server access for a group

1. On the server details page, choose **Actions**, and then choose **Delete Access**.
2. In the dialog box, confirm that you want to remove access for this group.

When you return to the server details page, you see that the access for this group is no longer listed.

Connecting to the server using SSH (Secure Shell)

After you configure your server and users, you can connect to the server using SSH and use the fully qualified username for a user that has access.

```
sftp user@active-directory-domain@vpc-endpoint
```

For example: `transferuserexample@mycompany.com@vpce-0123456abcdef-789xyz.vpc-svc-987654zyxabc.us-east-1.vpce.amazonaws.com`.

This format targets the search of the federation, limiting the search of a potentially large Active Directory.

Note

You can specify the simple username. However, in this case, the Active Directory code has to search all the directories in the federation. This might limit the search, and authentication might fail even if the user should have access.

After authenticating, the user is located in the home directory that you specified when you configured the user.

Connecting AWS Transfer Family to a self-managed Active Directory using forests and trusts

Users in your self-managed Active Directory (AD) can also use AWS IAM Identity Center for single sign-on access to AWS accounts and Transfer Family servers. To do that, AWS Directory Service has the following options available:

- One-way forest trust (outgoing from AWS Managed Microsoft AD and incoming for on-premises Active Directory) works only for the root domain.
- For child domains, you can use either of the following:
 - Use two-way trust between AWS Managed Microsoft AD and on-premises Active Directory
 - Use one-way external trust to each child domain.

When connecting to the server using a trusted domain, the user needs to specify the trusted domain, for example `transferuserexample@mycompany.com`.

Using AWS Directory Service for Azure Active Directory Domain Services

- To take advantage of your existing Active Directory forest for your SFTP Transfer needs, you can use [Active Directory Connector](#).
- If you want the benefits of Active Directory and high availability in a fully managed service, you can use AWS Directory Service for Microsoft Active Directory. For details, see [Using AWS Directory Service for Microsoft Active Directory](#).

This topic describes how to use an Active Directory Connector and [Azure Active Directory Domain Services \(Azure ADDS\)](#) to authenticate SFTP Transfer users with [Azure Active Directory](#).

Topics

- [Before you start using AWS Directory Service for Azure Active Directory Domain Services](#)
- [Step 1: Adding Azure Active Directory Domain Services](#)
- [Step 2: Creating a service account](#)
- [Step 3: Setting up AWS Directory using AD Connector](#)
- [Step 4: Setting up AWS Transfer Family server](#)
- [Step 5: Granting access to groups](#)
- [Step 6: Testing users](#)

Before you start using AWS Directory Service for Azure Active Directory Domain Services

For AWS, you need the following:

- A virtual private cloud (VPC) in an AWS region where you are using your Transfer Family servers
- At least two private subnets in your VPC
- The VPC must have internet connectivity
- A customer gateway and Virtual private gateway for site-to-site VPN connection with Microsoft Azure

For Microsoft Azure, you need the following:

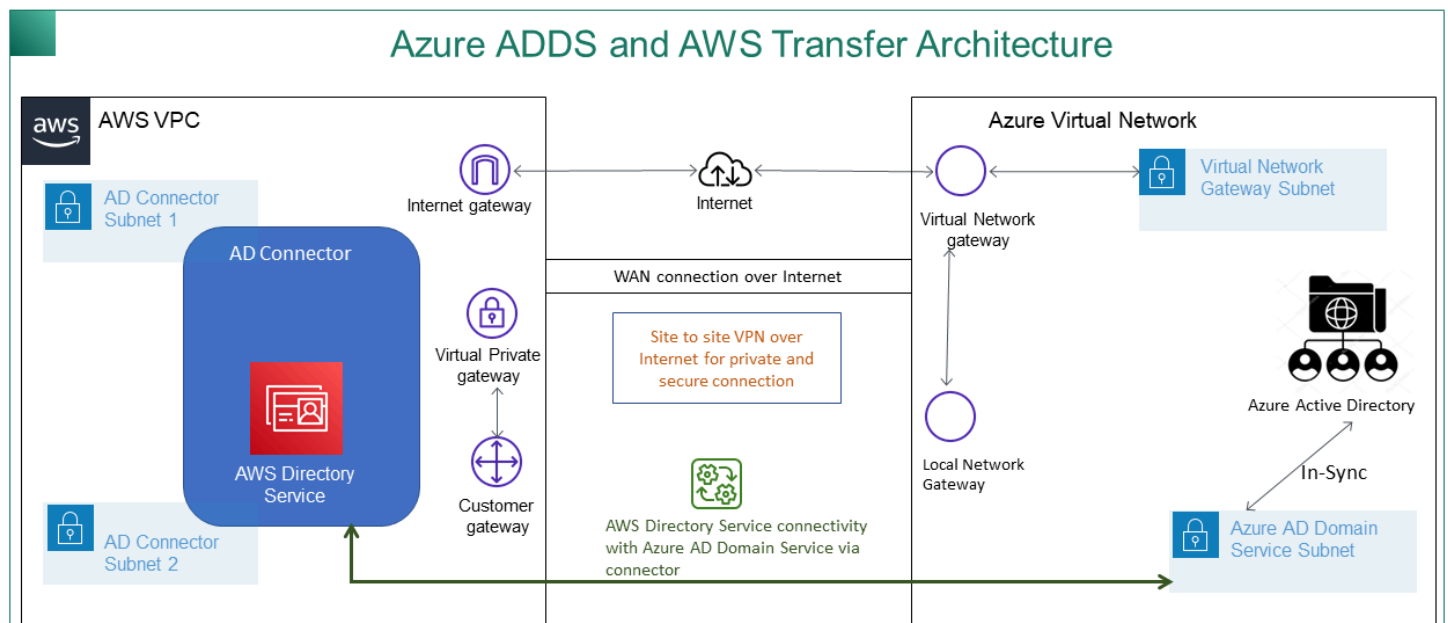
- An Azure Active Directory and Active directory domain service (Azure ADDS)
- An Azure resource group
- An Azure virtual network
- VPN connectivity between your Amazon VPC and your Azure resource group

Note

This can be through native IPSEC tunnels or using VPN appliances. In this topic, we use IPSEC tunnels between an Azure Virtual network gateway and local network gateway. The tunnels must be configured to allow traffic between your Azure ADDS endpoints and the subnets that house your AWS VPC.

- A customer gateway and Virtual private gateway for site-to-site VPN connection with Microsoft Azure

The following diagram shows the configuration needed before you begin.



Step 1: Adding Azure Active Directory Domain Services

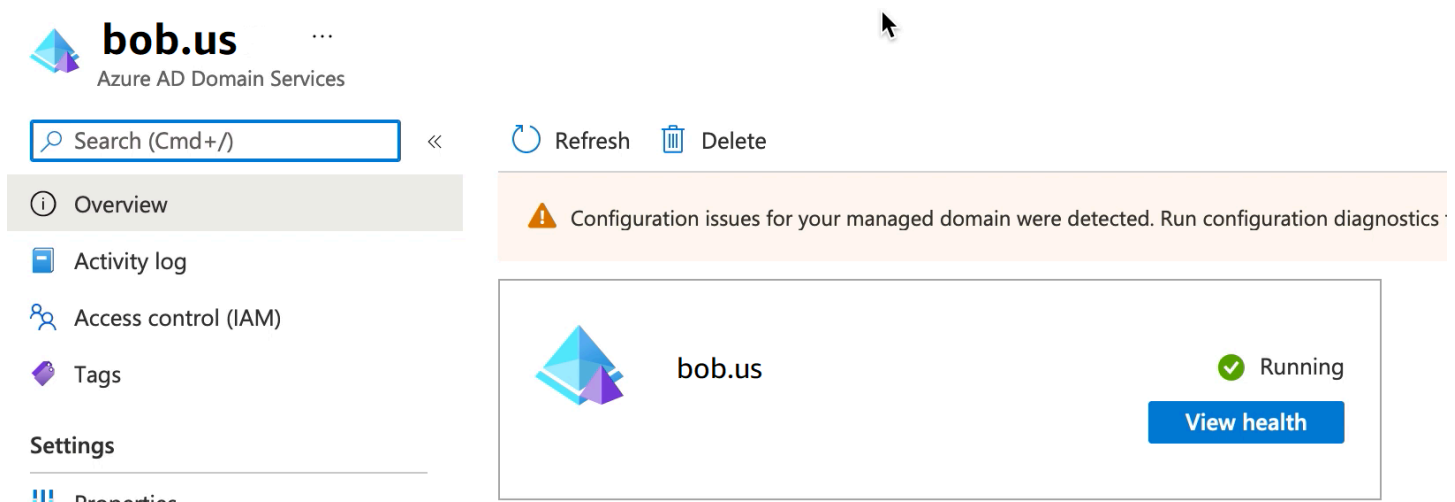
Azure AD does not support Domain joining instances by default. To perform actions like Domain Join, and to use tools such as Group Policy, administrators must enable Azure Active Directory Domain Services. If you have not already added Azure AD DS, or your existing implementation is

not associated with the domain that you want your SFTP Transfer server to use, you must add a new instance.

For information about enabling Azure Active Directory Domain Services (Azure ADDS), see [Tutorial: Create and configure an Azure Active Directory Domain Services managed domain](#).

Note

When you enable Azure ADDS, make sure it is configured for the resource group and the Azure AD domain to which you are connecting your SFTP Transfer server.



The screenshot shows the Azure AD Domain Services console for the domain **bob.us**. The page title is "bob.us Azure AD Domain Services". Below the title is a search bar with the text "Search (Cmd+/)". To the right of the search bar are "Refresh" and "Delete" buttons. A navigation menu on the left includes "Overview" (selected), "Activity log", "Access control (IAM)", "Tags", "Settings", and "Diagnostics". A warning banner at the top right states: "Configuration issues for your managed domain were detected. Run configuration diagnostics". Below the banner is a card for the domain **bob.us**, which shows a green checkmark and the status "Running". A "View health" button is located at the bottom right of the card.

Step 2: Creating a service account

Azure AD must have one service account that is part of an Admin group in Azure ADDS. This account is used with the AWS Active Directory connector. Make sure this account is in sync with Azure ADDS.










bobatusa | Profile

User

<<  Edit  Reset password  Revoke sessions  Delete  Refresh |  Got feedback?

 Diagnose and solve problems

Manage

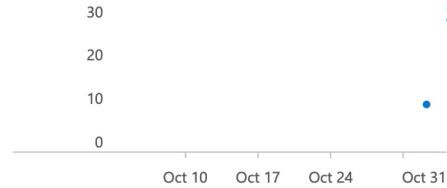
-  Profile
-  Assigned roles
-  Administrative units
-  Groups
-  Applications
-  Licenses
-  Devices
-  Azure role assignments
-  Authentication methods

bobatusa

bobsmith@xyz.com



User Sign-ins



Group memberships



2

Creation time
10/6/2021, 1:32:27 AM

Identity

Name	First name	Last name
bobatusa	Bob	Smith
User Principal Name	User type	
bobsmith@xyz.com	Member	

Activity

-  Sign-in logs
-  Audit logs


Tip

Multi-factor authentication for Azure Active Directory is not supported for Transfer Family servers that use the SFTP protocol. The Transfer Family server cannot provide the MFA token after a user authenticates to SFTP. Make sure to disable MFA before you attempt to connect.

multi-factor authentication

users service settings

Note: only users licensed to use Microsoft Online Services are eligible for Multi-Factor Authentication. [Learn more about how to license other users.](#)
Before you begin, take a look at the [multi-factor auth deployment guide](#).

View: Sign-in allowed users  Multi-Factor Auth status: Any bulk update

<input type="checkbox"/>	DISPLAY NAME ^	USER NAME	MULTI-FACTOR AUTH STATUS
<input type="checkbox"/>	Christopher	admin@christopher[REDACTED].com	Disabled
<input type="checkbox"/>	Robert	test@christopher[REDACTED].com	Disabled

Select a user

Step 3: Setting up AWS Directory using AD Connector

After you have configured Azure ADDS, and created a service account with IPSEC VPN tunnels between your AWS VPC and Azure Virtual network, you can test the connectivity by pinging the Azure ADDS DNS IP address from any AWS EC2 instance.

After you verify the connection is active, you can continue below.

To set up your AWS Directory using AD Connector

1. Open the [Directory Service](#) console and select **Directories**.
2. Select **Set up directory**.
3. For directory type, choose **AD Connector**.
4. Select a directory size, select **Next**, then select your VPC and Subnets.
5. Select **Next**, then fill in the fields as follows:
 - **Directory DNS name:** enter the domain name you are using for your Azure ADDS.
 - **DNS IP addresses:** enter you Azure ADDS IP addresses.
 - **Server account username and password:** enter the details for the service account you created in *Step 2: Create a service account*.
6. Complete the screens to create the directory service.

Now the directory status should be **Active**, and it is ready to be used with an SFTP Transfer server.

The screenshot shows the AWS Directory Service console. At the top, there is a breadcrumb trail: "Directory Service > Directories". Below this is a "Did you know?" notification box with an information icon and a close button. The main content area is titled "Directories (1) Info" and includes a search bar with the placeholder text "Find by directory ID or name". To the right of the search bar are buttons for "Refresh", "Actions", and "Set up directory". Below the search bar is a table with the following columns: Directory ID, Directory name, Type, Size, Multi-Region, Status, and Launch date. The table contains one entry with the following details:

Directory ID	Directory name	Type	Size	Multi-Region	Status	Launch date
d-906752c0d7	[Redacted]	AD Connector	Small	Not applicable	Active	Nov 3, 2021

Step 4: Setting up AWS Transfer Family server

Create a Transfer Family server with the SFTP protocol, and the identity provider type of **AWS Directory Service**. From **Directory** drop down list, select the directory you added in *Step 3: Setup AWS Directory using AD Connector*.

Note

You can't delete a Microsoft AD directory in AWS Directory Service if you used it in a Transfer Family server. You must delete the server first, and then you can delete the directory.

Step 5: Granting access to groups

After you create the server, you must choose which groups in the directory should have access to upload and download files over the enabled protocols using AWS Transfer Family. You do this by creating an *access*.

Note

Users must belong *directly* to the group to which you are granting access. For example, assume that Bob is a user and belongs to groupA, and groupA itself is included in groupB.

- If you grant access to groupA, Bob is granted access.
- If you grant access to groupB (and not to groupA), Bob does not have access.

In order to grant access you need to retrieve the SID for the group.

Use the following Windows PowerShell command to retrieve the SID for a group, replacing *YourGroupName* with the name of the group.

```
Get-ADGroup -Filter {samAccountName -like "YourGroupName*"} -Properties * | Select SamAccountName, ObjectSid
```

```

> Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\bobatusa> Get-ADGroup -Filter {samAccountName -like "AAD DC Administrators"}
SamAccountName      ObjectSid
-----
AAD DC Administrators S-1-5-21-375932292-1747164136-3628472596-1104

```

Grant access to groups

1. Open <https://console.aws.amazon.com/transfer/>.
2. Navigate to your server details page and in the **Accesses** section, choose **Add access**.
3. Enter the SID you received from the output of the previous procedure.
4. For **Access**, choose an AWS Identity and Access Management role for the group.
5. In the **Policy** section, choose a policy. The default value is **None**.
6. For **Home directory**, choose an Amazon S3 bucket that corresponds to the group's home directory.
7. Choose **Add** to create the association.

The details from your Transfer server should look similar to the following:

The screenshot displays the AWS Transfer Family console interface. It is divided into two main sections: 'Protocols' and 'Identity provider'. Below these is the 'Accesses (1)' section.

- Protocols:** Shows 'SFTP' as the protocol over which clients can connect to the server's endpoint.
- Identity provider:** Shows 'AWS Directory Service' as the identity provider type and 'd-123456789a' as the Directory ID.
- Accesses (1):** A table with one entry:

External Id	Home directory	Role
S-1-5-21-375932292-1747164136-3628472596-1104	/s3/transfer	sftp-user-role

Step 6: Testing users

You can test ([Testing users](#)) whether a user has access to the AWS Managed Microsoft AD directory for your server. A user must be in exactly one group (an external ID) that is listed in the **Access** section of the **Endpoint configuration** page. If the user is in no groups, or is in more than a single group, that user is not granted access.

Working with custom identity providers

To authenticate your users, you can use your existing identity provider with AWS Transfer Family. You integrate your identity provider using an AWS Lambda function, which authenticates and authorizes your users for access to Amazon S3 or Amazon Elastic File System (Amazon EFS). For details, see [Using AWS Lambda to integrate your identity provider](#). You can also access CloudWatch graphs for metrics such as number of files and bytes transferred in the AWS Transfer Family Management Console, giving you a single pane of glass to monitor file transfers using a centralized dashboard.

Alternatively, you can provide a RESTful interface with a single Amazon API Gateway method. Transfer Family calls this method to connect to your identity provider, which authenticates and authorizes your users for access to Amazon S3 or Amazon EFS. Use this option if you need a RESTful API to integrate your identity provider or if you want to use AWS WAF to leverage its capabilities for geo-blocking or rate-limiting requests. For details, see [Using Amazon API Gateway to integrate your identity provider](#).

In either case, you can create a new server using the [AWS Transfer Family console](#) or the [CreateServer](#) API operation.

Note

Transfer Family provides a blog post and a workshop that walk you through building a file transfer solution. This solution leverages AWS Transfer Family for managed SFTP/FTPS endpoints and Amazon Cognito and DynamoDB for user management.

The blog post is available at [Using Amazon Cognito as an identity provider with AWS Transfer Family and Amazon S3](#). You can view the details for the workshop [here](#).

AWS Transfer Family provides the following options for working with custom identity providers.

- **Use AWS Lambda to connect your identity provider** – You can use an existing identity provider, backed by a Lambda function. You provide the name of the Lambda function. For more information, see [Using AWS Lambda to integrate your identity provider](#).
- **Use Amazon API Gateway to connect your identity provider** – You can create an API Gateway method backed by a Lambda function for use as an identity provider. You provide an Amazon API Gateway URL and an invocation role. For more information, see [Using Amazon API Gateway to integrate your identity provider](#).

For either option, you can also specify how to authenticate.

- **Password OR Key** – users can authenticate with either their password or their key. This is the default value.
- **Password ONLY** – users must provide their password to connect.
- **Key ONLY** – users must provide their private key to connect.
- **Password AND Key** – users must provide both their private key and their password to connect. The server checks the key first, and then if the key is valid, the system prompts for a password. If the private key provided does not match the public key that is stored, authentication fails.

Using multiple authentication methods to authenticate with your custom identity provider

The Transfer Family server controls the AND logic when you use multiple authentication methods. Transfer Family treats this as two separate requests to your custom identity provider: however, their effect is combined.

Both requests must return successfully with the correct response to allow the authentication to complete. Transfer Family requires the two responses to be complete, meaning they contain all of the required elements (role, home directory, policy and the POSIX profile if you're using Amazon EFS for storage). Transfer Family also requires that the password response must not include public keys.

The public key request must have a separate response from the identity provider. That behavior is unchanged when using **Password OR Key** or **Password AND Key**.

The SSH/SFTP protocol challenges the software client first with a public key authentication, then requests a password authentication. This operation mandates both are successful before the user is allowed to complete the authentication.

Topics

- [Using AWS Lambda to integrate your identity provider](#)
- [Using Amazon API Gateway to integrate your identity provider](#)

Using AWS Lambda to integrate your identity provider

Create an AWS Lambda function that connects to your custom identity provider. You can use any custom identity provider, such as Okta, Secrets Manager, OneLogin, or a custom data store that includes authorization and authentication logic.

Note

Before you create a Transfer Family server that uses Lambda as the identity provider, you must create the function. For an example Lambda function, see [Example Lambda functions](#). Or, you can deploy a CloudFormation stack that uses one of the [Lambda function templates](#). Also, make sure your Lambda function uses a resource-based policy that trusts Transfer Family. For an example policy, see [Lambda resource-based policy](#).

1. Open the [AWS Transfer Family console](#).
2. Choose **Create server** to open the **Create server** page. For **Choose an identity provider**, choose **Custom Identity Provider**, as shown in the following screenshot.

Choose an identity provider

Identity Provider for SFTP, FTPS, or FTP

Identity provider type
An identity provider manages user access for authentication and authorization

Service managed
Create and manage users within the service

AWS Directory Service Info
Enable users in AWS Managed AD or use your own self-managed AD in your on-premises environment or in AWS

Custom Identity Provider Info
Manage users by integrating an identity provider of your choice

Use AWS Lambda to connect your identity provider Info
Invoke an AWS Lambda function to call your identity provider's API for user authentication and authorization

Use Amazon API Gateway to connect your identity provider Info
Use a RESTful API method to call your identity provider's API for user authentication and authorization

AWS Lambda function

Authentication methods
Choose which authentication methods are required for users to connect to your server

Password OR public key

Password ONLY

Public Key ONLY

Password AND public key

Info Either a valid password or valid private key will be required during user authentication

Info Note

The choice of authentication methods is only available if you enable SFTP as one of the protocols for your Transfer Family server.

3. Make sure the default value, **Use AWS Lambda to connect your identity provider**, is selected.
4. For **AWS Lambda function**, choose the name of your Lambda function.
5. Fill in the remaining boxes, and then choose **Create server**. For details on the remaining steps for creating a server, see [Configuring an SFTP, FTPS, or FTP server endpoint](#).

Lambda resource-based policy

You must have a policy that references the Transfer Family server and Lambda ARNs. For example, you could use the following policy with your Lambda function that connects to your identity provider. The policy is escaped JSON as a string.

```
"Policy":
"{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "AllowTransferInvocation",
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:transfer:region:account-id:function:my-lambda-auth-
function",
      "Condition": {
        "ArnLike": {
          "AWS:SourceArn": "arn:aws:transfer:region:account-id:server/server-id"
        }
      }
    }
  ]
}"
```

Note

In the example policy above, replace each *user input placeholder* with your own information.

Event message structure

The event message structure from SFTP server sent to the authorizer Lambda function for a custom IDP is as follows.

```
{
  "username": "value",
```

```
"password": "value",
"protocol": "SFTP",
"serverId": "s-abcd123456",
"sourceIp": "192.168.0.100"
}
```

Where `username` and `password` are the values for the sign-in credentials that are sent to the server.

For example, you enter the following command to connect:

```
sftp bobusa@server_hostname
```

You are then prompted to enter your password:

```
Enter password:
mysecretpassword
```

You can check this from your Lambda function by printing the passed event from within the Lambda function. It should look similar to the following text block.

```
{
  "username": "bobusa",
  "password": "mysecretpassword",
  "protocol": "SFTP",
  "serverId": "s-abcd123456",
  "sourceIp": "192.168.0.100"
}
```

The event structure is similar for FTP and FTPS: the only difference is those values are used for the `protocol` parameter, rather than SFTP.

Lambda functions for authentication

To implement different authentication strategies, edit the Lambda function. To help you meet your application's needs, you can deploy a CloudFormation stack. For more information about Lambda, see the [AWS Lambda Developer Guide](#) or [Building Lambda functions with Node.js](#).

Topics

- [Lambda function templates](#)
- [Valid Lambda values](#)

- [Example Lambda functions](#)
- [Testing your configuration](#)

Lambda function templates

You can deploy an AWS CloudFormation stack that uses a Lambda function for authentication. We provide several templates that authenticate and authorize your users using sign-in credentials. You can modify these templates or AWS Lambda code to further customize user access.

Note

You can create a FIPS-enabled AWS Transfer Family server through AWS CloudFormation by specifying a FIPS-enabled security policy in your template. Available security policies are described in [Security policies for AWS Transfer Family servers](#)

To create an AWS CloudFormation stack to use for authentication

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. Follow the instructions for deploying an AWS CloudFormation stack from an existing template in [Selecting a stack template](#) in the *AWS CloudFormation User Guide*.
3. Use one of the following templates to create a Lambda function to use for authentication in Transfer Family.

- [Classic \(Amazon Cognito\) stack template](#)

A basic template for creating a AWS Lambda for use as a custom identity provider in AWS Transfer Family. It authenticates against Amazon Cognito for password-based authentication and public keys are returned from an Amazon S3 bucket if public key based authentication is used. After deployment, you can modify the Lambda function code to do something different.

- [AWS Secrets Manager stack template](#)

A basic template that uses AWS Lambda with an AWS Transfer Family server to integrate Secrets Manager as an identity provider. It authenticates against an entry in AWS Secrets Manager of the format `aws/transfer/server-id/username`. Additionally, the secret must hold the key-value pairs for all user properties returned to Transfer Family. After deployment, you can modify the Lambda function code to do something different.

- [Okta stack template](#): A basic template that uses AWS Lambda with an AWS Transfer Family server to integrate Okta as a custom identity provider.
- [Okta-mfa stack template](#): A basic template that uses AWS Lambda with an AWS Transfer Family server to integrate Okta, with MultiFactor Authentication, as a custom identity provider.
- [Azure Active Directory template](#): details for this stack are described in the blog post [Authenticating to AWS Transfer Family with Azure Active Directory and AWS Lambda](#).

After the stack has been deployed, you can view details about it on the **Outputs** tab in the CloudFormation console.

Deploying one of these stacks is the easiest way to integrate a custom identity provider into the Transfer Family workflow.

Valid Lambda values

The following table describes details for the values that Transfer Family accepts for Lambda functions that are used for custom identity providers.

Value	Description	Required
Role	Specifies the Amazon Resource Name (ARN) of the IAM role that controls your users' access to your Amazon S3 bucket or Amazon EFS file system. The policies attached to this role determine the level of access that you want to provide your users when transferring files into and out of your Amazon S3 or Amazon EFS file system. The IAM role should also contain a trust relationship that allows the server to access your	Required

Value	Description	Required
	<p>resources when servicing your users' transfer requests.</p> <p>For details on establishing a trust relationship, see To establish a trust relationship.</p>	
PosixProfile	<p>The full POSIX identity, including user ID (Uid), group ID (Gid), and any secondary group IDs (Secondary Gids), that controls your users' access to your Amazon EFS file systems. The POSIX permissions that are set on files and directories in your file system determine the level of access your users get when transferring files into and out of your Amazon EFS file systems.</p>	Required for Amazon EFS backing storage
PublicKeys	<p>A list of SSH public key values that are valid for this user. An empty list implies that this is not a valid login. Must not be returned during password authentication.</p>	Optional
Policy	<p>A session policy for your user so that you can use the same IAM role across multiple users. This policy scopes down user access to portions of their Amazon S3 bucket.</p>	Optional

Value	Description	Required
HomeDirectoryType	<p>The type of landing directory (folder) that you want your users' home directory to be when they log in to the server.</p> <ul style="list-style-type: none"> If you set it to PATH, the user sees the absolute Amazon S3 bucket or Amazon EFS paths as is in their file transfer protocol clients. If you set it to LOGICAL, you must provide mappings in the HomeDirectoryDetails parameter to make Amazon S3 or Amazon EFS paths visible to your users. 	Optional
HomeDirectoryDetails	<p>Logical directory mappings that specify which Amazon S3 or Amazon EFS paths and keys should be visible to your user and how you want to make them visible. You must specify the Entry and Target pair, where Entry shows how the path is made visible and Target is the actual Amazon S3 or Amazon EFS path.</p>	Required if HomeDirectoryType has a value of LOGICAL

Value	Description	Required
HomeDirectory	The landing directory for a user when they log in to the server using the client.	Optional

Note

HomeDirectoryDetails is a string representation of a JSON map. This is in contrast to PosixProfile, which is an actual JSON map object, and PublicKeys which is a JSON array of strings. See the code examples for the language-specific details.

Example Lambda functions

This section presents some example Lambda functions, in both NodeJS and Python.

Note

In these examples, the user, role, POSIX profile, password, and home directory details are all examples, and must be replaced with your actual values.

Logical home directory, NodeJS

The following NodeJS example function provides the details for a user that has a [logical home directory](#).

```
// GetUserConfig Lambda

exports.handler = (event, context, callback) => {
  console.log("Username:", event.username, "ServerId: ", event.serverId);

  var response;
  // Check if the username presented for authentication is correct. This doesn't
  // check the value of the server ID, only that it is provided.
  if (event.serverId !== "" && event.username == 'example-user') {
    var homeDirectoryDetails = [
      {
```

```

    Entry: "/",
    Target: "/fs-faa1a123"
  }
];
response = {
  Role: 'arn:aws:iam::123456789012:role/transfer-access-role', // The user is
authenticated if and only if the Role field is not blank
  PosixProfile: {"Gid": 65534, "Uid": 65534}, // Required for EFS access, but
not needed for S3
  HomeDirectoryDetails: JSON.stringify(homeDirectoryDetails),
  HomeDirectoryType: "LOGICAL",
};

// Check if password is provided
if (!event.password) {
  // If no password provided, return the user's SSH public key
  response['PublicKeys'] = [ "ssh-
rsa abcdef0123456789abcdef0123456789abcdef0123456789" ];
  // Check if password is correct
} else if (event.password !== 'Password1234') {
  // Return HTTP status 200 but with no role in the response to indicate
authentication failure
  response = {};
}
} else {
  // Return HTTP status 200 but with no role in the response to indicate
authentication failure
  response = {};
}
callback(null, response);
};

```

Path-based home directory, NodeJS

The following NodeJS example function provides the details for a user that has a path-based home directory.

```

// GetUserConfig Lambda

exports.handler = (event, context, callback) => {
  console.log("Username:", event.username, "ServerId: ", event.serverId);

  var response;

```

```

// Check if the username presented for authentication is correct. This doesn't
check the value of the server ID, only that it is provided.
// There is also event.protocol (one of "FTP", "FTPS", "SFTP") and event.sourceIp
(e.g., "127.0.0.1") to further restrict logins.
if (event.serverId !== "" && event.username == 'example-user') {
  response = {
    Role: 'arn:aws:iam::123456789012:role/transfer-access-role', // The user is
authenticated if and only if the Role field is not blank
    Policy: '', // Optional, JSON stringified blob to further restrict this user's
permissions
    HomeDirectory: '/fs-faa1a123' // Not required, defaults to '/'
  };

  // Check if password is provided
  if (!event.password) {
    // If no password provided, return the user's SSH public key
    response['PublicKeys'] = [ "ssh-
rsa abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789" ];
    // Check if password is correct
  } else if (event.password !== 'Password1234') {
    // Return HTTP status 200 but with no role in the response to indicate
authentication failure
    response = {};
  }
} else {
  // Return HTTP status 200 but with no role in the response to indicate
authentication failure
  response = {};
}
callback(null, response);
};

```

Logical home directory, Python

The following Python example function provides the details for a user that has a [logical home directory](#).

```

# GetUserConfig Python Lambda with LOGICAL HomeDirectoryDetails
import json

def lambda_handler(event, context):
    print("Username: {}, ServerId: {}".format(event['username'], event['serverId']))

```

```

response = {}

# Check if the username presented for authentication is correct. This doesn't
check the value of the server ID, only that it is provided.
if event['serverId'] != '' and event['username'] == 'example-user':
    homeDirectoryDetails = [
        {
            'Entry': '/',
            'Target': '/fs-faa1a123'
        }
    ]
    response = {
        'Role': 'arn:aws:iam::123456789012:role/transfer-access-role', # The user will
        be authenticated if and only if the Role field is not blank
        'PosixProfile': {"Gid": 65534, "Uid": 65534}, # Required for EFS access, but
        not needed for S3
        'HomeDirectoryDetails': json.dumps(homeDirectoryDetails),
        'HomeDirectoryType': "LOGICAL"
    }

    # Check if password is provided
    if event.get('password', '') == '':
        # If no password provided, return the user's SSH public key
        response['PublicKeys'] = [ "ssh-
rsa abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789" ]
        # Check if password is correct
        elif event['password'] != 'Password1234':
            # Return HTTP status 200 but with no role in the response to indicate
            authentication failure
            response = {}
        else:
            # Return HTTP status 200 but with no role in the response to indicate
            authentication failure
            response = {}

    return response

```

Path-based home directory, Python

The following Python example function provides the details for a user that has a path-based home directory.

```
# GetUserConfig Python Lambda with PATH HomeDirectory
```

```

def lambda_handler(event, context):
    print("Username: {}, ServerId: {}".format(event['username'], event['serverId']))

    response = {}

    # Check if the username presented for authentication is correct. This doesn't
    # check the value of the server ID, only that it is provided.
    # There is also event.protocol (one of "FTP", "FTPS", "SFTP") and event.sourceIp
    # (e.g., "127.0.0.1") to further restrict logins.
    if event['serverId'] != '' and event['username'] == 'example-user':
        response = {
            'Role': 'arn:aws:iam::123456789012:role/transfer-access-role', # The user will
            # be authenticated if and only if the Role field is not blank
            'Policy': '', # Optional, JSON stringified blob to further restrict this
            # user's permissions
            'HomeDirectory': '/fs-fs-faa1a123',
            'HomeDirectoryType': "PATH" # Not strictly required, defaults to PATH
        }

    # Check if password is provided
    if event.get('password', '') == '':
        # If no password provided, return the user's SSH public key
        response['PublicKeys'] = [ "ssh-
rsa abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789" ]
    # Check if password is correct
    elif event['password'] != 'Password1234':
        # Return HTTP status 200 but with no role in the response to indicate
        # authentication failure
        response = {}
    else:
        # Return HTTP status 200 but with no role in the response to indicate
        # authentication failure
        response = {}

    return response

```

Testing your configuration

After you create your custom identity provider, you should test your configuration.

Console

To test your configuration by using the AWS Transfer Family console

1. Open the [AWS Transfer Family console](#).
2. On the **Servers** page, choose your new server, choose **Actions**, and then choose **Test**.
3. Enter the text for **Username** and **Password** that you set when you deployed the AWS CloudFormation stack. If you kept the default options, the username is `myuser` and the password is `MySuperSecretPassword`.
4. Choose the **Server protocol** and enter the IP address for **Source IP**, if you set them when you deployed the AWS CloudFormation stack.

CLI

To test your configuration by using the AWS CLI

1. Run the [test-identity-provider](#) command. Replace each *user input placeholder* with your own information, as described in the subsequent steps.

```
aws transfer test-identity-provider --server-id s-1234abcd5678efgh --user-name myuser --user-password MySuperSecretPassword --server-protocol FTP --source-ip 127.0.0.1
```

2. Enter the server ID.
3. Enter the username and password that you set when you deployed the AWS CloudFormation stack. If you kept the default options, the username is `myuser` and the password is `MySuperSecretPassword`.
4. Enter the server protocol and source IP address, if you set them when you deployed the AWS CloudFormation stack.

If user authentication succeeds, the test returns a `StatusCode: 200` HTTP response, an empty string `Message: ""` (which would contain a reason for failure otherwise), and a `Response` field.

Note

In the response example below, the Response field is a JSON object that has been "stringified" (converted into a flat JSON string that can be used inside a program), and contains the details of the user's roles and permissions.

```
{
  "Response": "{\\\"Policy\\\":\\\"{\\\"Version\\\":\\\"2012-10-17\\\",\\\"Statement\\\":[
  {\\\"Sid\\\":\\\"ReadAndListAllBuckets\\\",\\\"Effect\\\":\\\"Allow\\\",\\\"Action\\\":[
  \\\"s3:ListAllMybuckets\\\",\\\"s3:GetBucketLocation\\\",\\\"s3:ListBucket\\\",\\\"s3:
  GetObjectVersion\\\",\\\"s3:GetObjectVersion\\\"],\\\"Resource\\\":\\\"*\\\"}]}\\\",
  \\\"Role\\\":\\\"arn:aws:iam:000000000000:role/MyUserS3AccessRole\\\",\\\"HomeDirectory\\\":\\\"/
  \\\"}\\\",
  \\\"StatusCode\\\": 200,
  \\\"Message\\\": \\\"\\\"
}"
```

Using Amazon API Gateway to integrate your identity provider

This topic describes how to use an AWS Lambda function to back an API Gateway method. Use this option if you need a RESTful API to integrate your identity provider or if you want to use AWS WAF to leverage its capabilities for geo-blocking or rate-limiting requests.

Limitations if using an API Gateway to integrate your identity provider

- This configuration does not support custom domains.
- This configuration does not support a private API Gateway URL.

If you need either of these, you can use Lambda as an identity provider, without API Gateway. For details, see [Using AWS Lambda to integrate your identity provider](#).

Authenticating using an API Gateway method

You can create an API Gateway method for use as an identity provider for Transfer Family. This approach provides a highly secure way for you to create and provide APIs. With API Gateway, you can create an HTTPS endpoint so that all incoming API calls are transmitted with greater security. For more details about the API Gateway service, see the [API Gateway Developer Guide](#).

API Gateway offers an authorization method named `AWS_IAM`, which gives you the same authentication based on AWS Identity and Access Management (IAM) that AWS uses internally. If you enable authentication with `AWS_IAM`, only callers with explicit permissions to call an API can reach that API's API Gateway method.

To use your API Gateway method as a custom identity provider for Transfer Family, enable IAM for your API Gateway method. As part of this process, you provide an IAM role with permissions for Transfer Family to use your gateway.

Note

To improve security, you can configure a web application firewall. AWS WAF is a web application firewall that lets you monitor the HTTP and HTTPS requests that are forwarded to an Amazon API Gateway. For details, see [Add a web application firewall](#).

To use your API Gateway method for custom authentication with Transfer Family

1. Create an AWS CloudFormation stack. To do this:

Note

The stack templates have been updated to use BASE64-encoded passwords: for details, see [Improvements to the AWS CloudFormation templates](#).

- a. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
- b. Follow the instructions for deploying an AWS CloudFormation stack from an existing template in [Selecting a stack template](#) in the *AWS CloudFormation User Guide*.
- c. Use one of the following basic templates to create an AWS Lambda-backed API Gateway method for use as a custom identity provider in Transfer Family.
 - [Basic stack template](#)

By default, your API Gateway method is used as a custom identity provider to authenticate a single user in a single server using a hard-coded SSH (Secure Shell)

key or password. After deployment, you can modify the Lambda function code to do something different.

- [AWS Secrets Manager stack template](#)

By default, your API Gateway method authenticates against an entry in Secrets Manager of the format `aws/transfer/server-id/username`. Additionally, the secret must hold the key-value pairs for all user properties returned to Transfer Family. After deployment, you can modify the Lambda function code to do something different. For more information, see the blog post [Enable password authentication for AWS Transfer Family using AWS Secrets Manager](#).

- [Okta stack template](#)

Your API Gateway method integrates with Okta as a custom identity provider in Transfer Family. For more information, see the blog post [Using Okta as an identity provider with AWS Transfer Family](#).

Deploying one of these stacks is the easiest way to integrate a custom identity provider into the Transfer Family workflow. Each stack uses the Lambda function to support your API method based on API Gateway. You can then use your API method as a custom identity provider in Transfer Family. By default, the Lambda function authenticates a single user called `myuser` with a password of `MySuperSecretPassword`. After deployment, you can edit these credentials or update the Lambda function code to do something different.

Important

We recommend that you edit the default user and password credentials.

After the stack has been deployed, you can view details about it on the **Outputs** tab in the CloudFormation console. These details include the stack's Amazon Resource Name (ARN), the ARN of the IAM role that the stack created, and the URL for your new gateway.

Note

If you are using the custom identity provider option to enable password-based authentication for your users, and you enable the request and response logging provided by API Gateway, API Gateway logs your users' passwords to your Amazon

CloudWatch Logs. We don't recommend using this log in your production environment. For more information, see [Set up CloudWatch API logging in API Gateway](#) in the *API Gateway Developer Guide*.

2. Check the API Gateway method configuration for your server. To do this:
 - a. Open the API Gateway console at <https://console.aws.amazon.com/apigateway/>.
 - b. Choose the **Transfer Custom Identity Provider basic template API** that the AWS CloudFormation template generated. You might need to select your region to see your gateways.
 - c. In the **Resources** pane, choose **GET**. The following screenshot shows the correct method configuration.

The screenshot displays the AWS API Gateway console interface for configuring a method. On the left, a navigation pane shows a tree structure with the 'GET' method selected under the '/config' resource. The main area is titled 'Method request settings' and includes an 'Edit' button. The configuration is as follows:

- Authorization:** AWS_IAM
- API key required:** False
- Request validator:** None
- SDK operation name:** Generated based on method and path
- Request paths (0):** No request paths defined.
- URL query string parameters (2):**

Name	Required	Caching
protocol	False	Inactive
sourceIp	False	Inactive
- HTTP request headers (1):**

Name	Required	Caching
PasswordBase64	False	Inactive
- Request body (0):** No request body defined.

At this point, your API gateway is ready to be deployed.

3. For **Actions**, choose **Deploy API**. For **Deployment stage**, choose **prod**, and then choose **Deploy**.

After the API Gateway method is successfully deployed, view its performance in **Stages** > **Stage details**, as shown in the following screenshot.

Note

Copy the **Invoke URL** address that appears at the top of the screen. You might need it for the next step.

The screenshot shows the AWS API Gateway console interface. The breadcrumb navigation at the top reads: API Gateway > APIs > Transfer Custom Identity Provider basic template API > Stages. The main heading is 'Stages' with a 'Stage actions' dropdown and a 'Create stage' button. On the left, a sidebar shows a list of stages with 'prod' selected. The main content area is titled 'Stage details' and includes an 'Edit' button. The details are organized into sections: 'Stage details' (with sub-sections for Rate, Burst, Web ACL, and Client certificate), 'Invoke URL' (highlighted with a red box), and 'Active deployment' (showing a deployment on December 12, 2023). Below this is the 'Logs and tracing' section with options for CloudWatch logs, Detailed metrics, X-Ray tracing, and Custom access logging. At the bottom, there are tabs for 'Stage variables', 'Deployment history', 'Documentation history', 'Canary', and 'Tags'. The 'Stage variables' section shows 'Stage variables (0/0)' with a search bar and a table that currently contains no variables.

4. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
5. A Transfer Family should have been created for you, when you created the stack. If not, configure your server using these steps.

- a. Choose **Create server** to open the **Create server** page. For **Choose an identity provider**, choose **Custom**, then select **Use Amazon API Gateway to connect to your identity provider**, as shown in the following screenshot.

Choose an identity provider

Identity provider

Identity provider type
An identity provider manages user access for authentication and authorization

Service managed
Create and manage users within the service

AWS Directory Service [Info](#)
Enable users in AWS Managed AD or use your own self-managed AD in your on-premises environment or in AWS

Custom Identity Provider [Info](#)
Manage users by integrating an identity provider of your choice

Use AWS Lambda to connect your identity provider [Info](#)
Invoke an AWS Lambda function to call your identity provider's API for user authentication and authorization

Use Amazon API Gateway to connect your identity provider [Info](#)
Use a RESTful API method to call your identity provider's API for user authentication and authorization

Provide an Amazon API Gateway URL

Role
IAM role for the service to invoke your Amazon API Gateway URL

- b. In the **Provide an Amazon API Gateway URL** text box, paste the **Invoke URL** address of the API Gateway endpoint that you created in step 3 of this procedure.
- c. For **Role**, choose the IAM role that was created by the AWS CloudFormation template. This role allows Transfer Family to invoke your API gateway method.

The invocation role contains the AWS CloudFormation stack name that you selected for the stack that you created in step 1. It has the following format: *CloudFormation-stack-name*-TransferIdentityProviderRole-*ABC123DEF456GHI*.

- d. Fill in the remaining boxes, and then choose **Create server**. For details on the remaining steps for creating a server, see [Configuring an SFTP, FTPS, or FTP server endpoint](#).

Implementing your API Gateway method

To create a custom identity provider for Transfer Family, your API Gateway method must implement a single method that has a resource path of `/servers/serverId/users/username/config`. The *serverId* and *username* values come from the RESTful resource path. Also, add `sourceIp` and `protocol` as **URL Query String Parameters** in the **Method Request**, as shown in the following image.

The screenshot displays the AWS API Gateway console interface for configuring a method. On the left, a tree view shows the resource hierarchy: `/` > `/servers` > `/servers/{serverId}` > `/users` > `/users/{username}` > `/config`. The selected resource is `/servers/{serverId}/users/{username}/config` with the `GET` method.

The main configuration area shows the following details:

- ARN:** `arn:aws:execute-api-east-1:.....:*/GET/servers/{serverId}/users/{username}/config`
- Resource ID:** `aw4ihv`
- Method request settings:**
 - Authorization: `AWS_IAM`
 - Request validator: `None`
 - API key required: `False`
 - SDK operation name: `Generated based on method and path`
- Request paths (0):** No request paths defined.
- URL query string parameters (2):**

Name	Required	Caching
<code>protocol</code>	<code>False</code>	Inactive
<code>sourceIp</code>	<code>False</code>	Inactive

Note

The username must be a minimum of 3 and a maximum of 100 characters. You can use the following characters in the username: a–z, A–Z, 0–9, underscore (`_`), hyphen (`-`), period (`.`), and at sign (`@`). However, the username can't start with a hyphen (`-`), period (`.`), or at sign (`@`).

If Transfer Family attempts password authentication for your user, the service supplies a `Password:` header field. In the absence of a `Password:` header, Transfer Family attempts public key authentication to authenticate your user.

When you are using an identity provider to authenticate and authorize end users, in addition to validating their credentials, you can allow or deny access requests based on the IP addresses of the clients used by your end users. You can use this feature to ensure that data stored in your S3 buckets or your Amazon EFS file system can be accessed over the supported protocols only from IP addresses that you have specified as trusted. To enable this feature, you must include `sourceIp` in the Query string.

If you have multiple protocols enabled for your server and want to provide access using the same username over multiple protocols, you can do so as long as the credentials specific to each protocol have been set up in your identity provider. To enable this feature, you must include the `protocol` value in the RESTful resource path.

Your API Gateway method should always return HTTP status code `200`. Any other HTTP status code means that there was an error accessing the API.

Amazon S3 example response

The example response body is a JSON document of the following form for Amazon S3.

```
{
  "Role": "IAM role with configured S3 permissions",
  "PublicKeys": [
    "ssh-rsa public-key1",
    "ssh-rsa public-key2"
  ],
  "Policy": "STS Assume role session policy",
  "HomeDirectory": "/DOC-EXAMPLE-BUCKET/path/to/home/directory"
}
```

Note

The policy is escaped JSON as a string. For example:

```
"Policy":
"{
  \"Version\": \"2012-10-17\",
  \"Statement\":
```

```
[
  {"Condition":
    {"StringLike":
      {"s3:prefix":
        ["user/*", "user/"]}},
    "Resource": "arn:aws:s3::DOC-EXAMPLE-BUCKET",
    "Action": "s3:ListBucket",
    "Effect": "Allow",
    "Sid": "ListHomeDir"},
  {"Resource": "arn:aws:s3::*",
    "Action": ["s3:PutObject",
      "s3:GetObject",
      "s3:DeleteObjectVersion",
      "s3:DeleteObject",
      "s3:GetObjectVersion",
      "s3:GetObjectACL",
      "s3:PutObjectACL"],
    "Effect": "Allow",
    "Sid": "HomeDirObjectAccess"}]
]"
```

The following example response shows that a user has a logical home directory type.

```
{
  "Role": "arn:aws:iam::123456789012:role/transfer-access-role-s3",
  "HomeDirectoryType": "LOGICAL",
  "HomeDirectoryDetails": "[{"Entry": "\", \"Target\": \"/DOC-EXAMPLE-BUCKET1\"}]",
  "PublicKeys": [""]
}
```

Amazon EFS example response

The example response body is a JSON document of the following form for Amazon EFS.

```
{
  "Role": "IAM role with configured EFS permissions",
  "PublicKeys": [
    "ssh-rsa public-key1",
    "ssh-rsa public-key2"
  ],
  "PosixProfile": {
```

```

    "Uid": "POSIX user ID",
    "Gid": "POSIX group ID",
    "SecondaryGids": [Optional list of secondary Group IDs],
  },
  "HomeDirectory": "/fs-id/path/to/home/directory"
}

```

The `Role` field shows that successful authentication occurred. When doing password authentication (when you supply a `Password:` header), you don't need to provide SSH public keys. If a user can't be authenticated, for example, if the password is incorrect, your method should return a response without `Role` set. An example of such a response is an empty JSON object.

The following example response shows a user that has a logical home directory type.

```

{
  "Role": "arn:aws:iam::123456789012:role/transfer-access-role-efs",
  "HomeDirectoryType": "LOGICAL",
  "HomeDirectoryDetails": "[{"Entry": "\"/^", "Target": "\"/faa1a123"}]",
  "PublicKeys": [""],
  "PosixProfile": {"Uid": "65534", "Gid": "65534"}
}

```

You can include user policies in the Lambda function in JSON format. For more information about configuring user policies in Transfer Family, see [Managing access controls](#).

Default Lambda function

To implement different authentication strategies, edit the Lambda function that your gateway uses. To help you meet your application's needs, you can use the following example Lambda functions in Node.js. For more information about Lambda, see the [AWS Lambda Developer Guide](#) or [Building Lambda functions with Node.js](#).

The following example Lambda function takes your username, password (if you're performing password authentication), server ID, protocol, and client IP address. You can use a combination of these inputs to look up your identity provider and determine if the login should be accepted.

Note

If you have multiple protocols enabled for your server and want to provide access using the same username over multiple protocols, you can do so as long as the credentials specific to the protocol have been set up in your identity provider.

For File Transfer Protocol (FTP), we recommend maintaining separate credentials from Secure Shell (SSH) File Transfer Protocol (SFTP) and File Transfer Protocol over SSL (FTPS). We recommend maintaining separate credentials for FTP because, unlike SFTP and FTPS, FTP transmits credentials in clear text. By isolating FTP credentials from SFTP or FTPS, if FTP credentials are shared or exposed, your workloads using SFTP or FTPS remain secure.

This example function returns the role and logical home directory details, along with the public keys (if it performs public key authentication).

When you create service-managed users, you set their home directory, either logical or physical. Similarly, we need the Lambda function results to convey the desired user physical or logical directory structure. The parameters you set depend on the value for the [HomeDirectoryType](#) field.

- `HomeDirectoryType` set to `PATH` – the `HomeDirectory` field must then be an absolute Amazon S3 bucket prefix or Amazon EFS absolute path that is visible to your users.
- `HomeDirectoryType` set to `LOGICAL` – Do *not* set a `HomeDirectory` field. Instead, we set a `HomeDirectoryDetails` field that provides the desired Entry/Target mappings, similar to the described values in the [HomeDirectoryDetails](#) parameter for service-managed users.

The example functions are listed in [Example Lambda functions](#).

Lambda function for use with AWS Secrets Manager

To use AWS Secrets Manager as your identity provider, you can work with the Lambda function in the sample AWS CloudFormation template. The Lambda function queries the Secrets Manager service with your credentials and, if successful, returns a designated secret. For more information about Secrets Manager, see the [AWS Secrets Manager User Guide](#).

To download a sample AWS CloudFormation template that uses this Lambda function, go to the [Amazon S3 bucket provided by AWS Transfer Family](#).

Improvements to the AWS CloudFormation templates

Improvements to the API Gateway interface have been made to the published CloudFormation templates. The templates now use BASE64-encoded passwords with the API Gateway. Your existing deployments continue to work without this enhancement, but don't allow for passwords with characters outside the basic US-ASCII character set.

The changes in the template that enable this capability are as follows:

- The `GetUserConfigRequest` `AWS::ApiGateway::Method` resource has to have this `RequestTemplates` code (the line in italics is the updated line)

```
RequestTemplates:
  application/json: |
    {
      "username": "$util.urlDecode($input.params('username'))",
      "password":
"$util.escapeJavaScript($util.base64Decode($input.params('PasswordBase64'))).replaceAll('\\"',"'")",
      "protocol": "$input.params('protocol')",
      "serverId": "$input.params('serverId')",
      "sourceIp": "$input.params('sourceIp')"
    }
  }
```

- The `RequestParameters` for the `GetUserConfig` resource must change to use the `PasswordBase64` header (the line in italics is the updated line):

```
RequestParameters:
  method.request.header.PasswordBase64: false
  method.request.querystring.protocol: false
  method.request.querystring.sourceIp: false
```

To check if the template for your stack is the latest

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. From the list of stacks, choose your stack.
3. From the details panel, choose the **Template** tab.
4. Look for the following:
 - Search for `RequestTemplates`, and make sure you have this line:

```
"password":
  "$util.escapeJavaScript($util.base64Decode($input.params('PasswordBase64'))).replaceAll('\\"',"'")",
```

- Search for `RequestParameters`, and make sure you have this line:

```
method.request.header.PasswordBase64: false
```

If you don't see the updated lines, edit your stack. For details on how to update your AWS CloudFormation stack, see [Modifying a stack template](#) in the *AWS CloudFormation; User Guide*.

Using logical directories to simplify your Transfer Family directory structures

To simplify your AWS Transfer Family server directory structure, you can use logical directories. With logical directories, you can construct a virtual directory structure that uses user-friendly names that your users navigate when they connect to your Amazon S3 bucket or Amazon EFS file system. When you use logical directories, you can avoid disclosing absolute directory paths, Amazon S3 bucket names, and EFS file system names to your end users.

Note

You should use session policies so that your end users can only perform operations that you allow them to perform.

You should use logical directories to create a user-friendly, virtual directory for your end users and abstract away bucket names. Logical directory mappings only allow users to access their designated logical paths and subdirectories, and forbid relative paths that traverse the logical roots.

Transfer Family validates every path that might include relative elements and actively blocks these paths from resolving before we pass these paths to Amazon S3; this prevents your users from moving beyond their logical mappings.

Even though Transfer Family prevents your end users from accessing directories outside of their logical directory, we recommend you also use unique roles or session policies to enforce least privilege at the storage level.

You can use logical directories to set the user's root directory to a desired location within your storage hierarchy, by performing what is known as a **chroot** operation. In this mode, users are not able to navigate to a directory outside of the home or root directory that you've configured for them.

For example, although an Amazon S3 user has been scoped down to access only `/mybucket/home/${transfer:UserName}`, some clients allow users to traverse up a folder to `/mybucket/home`. In this situation, the user lands back on their intended home directory only

after logging out of and back in to the Transfer Family server again. Performing a **chroot** operation can prevent this situation from occurring.

You can create your own directory structure across buckets and prefixes. This feature is useful if you have a workflow that is expecting a specific directory structure that you can't replicate through bucket prefixes. You can also link to multiple non-contiguous locations within Amazon S3, similar to creating a symbolic link in a Linux file system where your directory path references a different location in the file system.

Logical directory FILE mappings

The `HomeDirectoryMapEntry` data type now includes a `Type` parameter. Before this parameter existed, you could have created a logical directory mapping where the target was a file. If you have previously created any of these kinds of logical directory mappings, you must explicitly set the `Type` to `FILE`, or these mappings won't work correctly going forward.

One way to do this is to call the `UpdateUser` API, and set the `Type` to `FILE` for the existing mapping.

Rules for using logical directories

Before you build your logical directory mappings, you should understand the following rules:

- When `Entry` is `"/`, you can have only one mapping because overlapping paths are not allowed.
- Logical directories support mappings of up to 2.1 MB (for service-managed users, this limit is 2,000 entries). That is, the data structure that contains the mappings has a maximum size of 2.1 MB. If you have a lot of mappings, you can calculate the size of your mappings as follows:
 1. Write out a typical mapping in the format `{"Entry": "/entry-path", "Target": "/target-path"}`, where *entry-path* and *target-path* are the actual values that you will use.
 2. Count the characters in that string, then add one (1).
 3. Multiply that number by the approximate number of mappings that you have for your server.

If the number that you estimated in step 3 is less than 2.1 MB, then your mappings are within the acceptable limit.

- Targets can use the `${transfer:UserName}` variable if the bucket or file system path has been parameterized based on the username.
- Targets can be paths in different buckets or file systems, but you must make sure that the mapped AWS Identity and Access Management (IAM) role (the `Role` parameter in the response) provides access to those buckets or file systems.
- Don't specify the `HomeDirectory` parameter, because this value is implied by the `EntryTarget` pairs when you're using the `LOGICAL` value for the `HomeDirectoryType` parameter.
- Targets must begin with a forward slash (/) character, but don't use trailing forward slashes (/) when you specify the `Target`. For example, `/DOC-EXAMPLE-BUCKET/images` is acceptable, but `DOC-EXAMPLE-BUCKET/images` and `/DOC-EXAMPLE-BUCKET/images/` are not.
- Amazon S3 is an object store, which means that folders are a virtual concept, and there is no actual directory hierarchy. If your application issues a `stat` operation from a client, everything is classified as a file when you're using Amazon S3 for storage. This behavior is described in [Organizing objects in the Amazon S3 console using folders](#) in the *Amazon Simple Storage Service User Guide*. If your application requires that `stat` accurately show whether something is a file or folder, you can use Amazon Elastic File System (Amazon EFS) as the storage option for your Transfer Family servers.
- If you're specifying logical directory values for your user, the parameter that you use depends on the type of user:
 - For service-managed users, provide logical directory values in `HomeDirectoryMappings`.
 - For custom identity provider users, provide logical directory values in `HomeDirectoryDetails`.

Important

Unless you choose to optimize performance for your Amazon S3 directories (when you create or update a server), the root directory must exist on startup. For Amazon S3, this means that you must have already created a zero-byte object ending with a forward slash (/) to create the root folder. Avoiding this issue is a reason to consider optimizing Amazon S3 performance.

Implementing logical directories and `chroot`

To use logical directories and `chroot` features, you must do the following:

Turn on logical directories for each user. Do this by setting the `HomeDirectoryType` parameter to `LOGICAL` when you create or update your user.

```
"HomeDirectoryType": "LOGICAL"
```

chroot

For **chroot**, create a directory structure that consists of a single `Entry` and `Target` pairing for each user. The root folder is the `Entry` point, and the `Target` is the location in your bucket or file system to map to.

Example for Amazon S3

```
[{"Entry": "/", "Target": "/mybucket/jane"}]
```

Example for Amazon EFS

```
[{"Entry": "/", "Target": "/fs-faa1a123/jane"}]
```

You can use an absolute path as in the previous example, or you can use a dynamic substitution for the username with `${transfer:UserName}`, as in the following example.

```
[{"Entry": "/", "Target":  
"/mybucket/${transfer:UserName}"}]
```

In the preceding example, the user is locked to their root directory and cannot traverse up higher in the hierarchy.

Virtual directory structure

For a virtual directory structure, you can create multiple `Entry Target` pairings, with targets anywhere in your S3 buckets or EFS file systems, including across multiple buckets or file systems, as long as the user's IAM role mapping has permissions to access them.

In the following virtual structure example, when the user logs into AWS SFTP, they are in the root directory with sub-directories of `/pics`, `/doc`, `/reporting`, and `/anotherpath/subpath/financials`.

Note

Unless you choose to optimize performance for your Amazon S3 directories (when you create or update a server), either the user or an administrator needs to create the directories if they don't already exist. Avoiding this issue is a reason to consider optimizing Amazon S3 performance.

For Amazon EFS, you still need the administrator to create the logical mappings or the / directory.

```
[  
{"Entry": "/pics", "Target": "/bucket1/pics"},  
{"Entry": "/doc", "Target": "/bucket1/anotherpath/docs"},  
{"Entry": "/reporting", "Target": "/reportingbucket/Q1"},  
{"Entry": "/anotherpath/subpath/financials", "Target": "/reportingbucket/financials"}]
```

Note

You can only upload files to the specific folders that you map. This means that in the previous example, you cannot upload to /anotherpath or anotherpath/subpath directories; only anotherpath/subpath/financials. You also cannot map to those paths directly, as overlapping paths are not allowed.

For example, assume that you create the following mappings:

```
{  
  "Entry": "/pics",  
  "Target": "/mybucket/pics"  
},  
{  
  "Entry": "/doc",  
  "Target": "/mybucket/mydocs"  
},  
{  
  "Entry": "/temp",  
  "Target": "/mybucket"  
}
```

You can only upload files to those buckets. When you first connect through `sftp`, you are dropped into the root directory, `/`. If you attempt to upload a file to that directory, the upload fails. The following commands show an example sequence:

```
sftp> pwd
Remote working directory: /
sftp> put file
Uploading file to /file
remote open("/file"): No such file or directory
```

To upload to any directory/sub-directory, you must explicitly map the path to the sub-directory.

For more information about configuring logical directories and **chroot** for your users, including an AWS CloudFormation template that you can download and use, see [Simplify your AWS SFTP Structure with chroot and logical directories](#) in the AWS Storage Blog.

Configure logical directories example

In this example, we create a user and assign two logical directories. The following command creates a new user (for an existing Transfer Family server) with logical directories `pics` and `doc`.

```
aws transfer create-user --user-name marymajor-logical --server-id s-11112222333344445
--role arn:aws:iam::1234abcd5678:role/marymajor-role --home-directory-type LOGICAL \
--home-directory-mappings "[{"Entry":"\pics\", \"Target\":\"/DOC-EXAMPLE-BUCKET1/
pics\"}, {"Entry":"\doc\", \"Target\":\"/DOC-EXAMPLE-BUCKET2/test/mydocs\"}]" \
--ssh-public-key-body file://~/.ssh/id_rsa.pub
```

If **marymajor** is an existing user and her home directory type is `PATH`, you can change it to `LOGICAL` with a similar command as the previous one.

```
aws transfer update-user --user-name marymajor-logical \
--server-id s-11112222333344445 --role arn:aws:iam::1234abcd5678:role/marymajor-role \
--home-directory-type LOGICAL --home-directory-mappings "[{"Entry":"\pics\",
\"Target\":\"/DOC-EXAMPLE-BUCKET1/pics\"}, \
{"Entry":"\doc\", \"Target\":\"/DOC-EXAMPLE-BUCKET2/test/mydocs\"}]"
```

Note the following:

- If the directories `/DOC-EXAMPLE-BUCKET1/pics` and `/DOC-EXAMPLE-BUCKET2/test/mydocs` don't already exist, the user (or an administrator) needs to create them.
- When **marymajor** connects to the server, and runs the `ls -l` command, Mary sees the following:

```
drwxr--r--  1      -      -      0 Mar 17 15:42 doc
drwxr--r--  1      -      -      0 Mar 17 16:04 pics
```

- **marymajor** cannot create any files or directories at this level. However, within `pics` and `doc`, she can add sub-directories.
- Files that Mary adds to `pics` and `doc` are added to Amazon S3 paths `/DOC-EXAMPLE-BUCKET1/pics` and `/DOC-EXAMPLE-BUCKET2/test/mydocs` respectively.
- In this example, we specify two different buckets to illustrate that possibility. However, you can use the same bucket for several or all of the logical directories that you specify for the user.

Configure logical directories for Amazon EFS

If your Transfer Family server uses Amazon EFS, the home directory for the user must be created with read and write access before the user can work in their logical home directory. The user cannot create this directory themselves, as they would lack permissions for `mkdir` on their logical home directory.

If the user's home directory does not exist, and they run an `ls` command, the system responds as follows:

```
sftp> ls
remote readdir ("/"): No such file or directory
```

A user with administrative access to the parent directory needs to create the user's logical home directory.

Custom AWS Lambda response


You can use logical directories with a Lambda function that connects to your custom identity provider. To do so, in your Lambda function, you specify the `HomeDirectoryType` as **LOGICAL**, and add `Entry` and `Target` values for the `HomeDirectoryDetails` parameter. For example:

```
HomeDirectoryType: "LOGICAL"
```

```
HomeDirectoryDetails: "[{"Entry": "\\", \"Target\": \"/DOC-EXAMPLE-BUCKET/theRealFolder"}]"
```

The following code is an example of a successful response from a custom Lambda authentication call.

```
aws transfer test-identity-provider --server-id s-1234567890abcdef0 --user-name myuser {
  "Url": "https://a1b2c3d4e5.execute-api.us-east-2.amazonaws.com/prod/servers/s-1234567890abcdef0/users/myuser/config",
  "Message": "",
  "Response": "{\"Role\": \"arn:aws:iam::123456789012:role/bob-usa-role\", \"HomeDirectoryType\": \"LOGICAL\", \"HomeDirectoryDetails\": \"[{{\\\"Entry\\\": \\\"/myhome\\\", \\\"Target\\\": \\\"/DOC-EXAMPLE-BUCKET/theRealFolder\\\"}]\\\", \"PublicKeys\": \"[ssh-rsa myrsapubkey]\"\", \"StatusCode\": 200
}
```

 **Note**

The "Url": line is returned only if you are using an API Gateway method as your custom identity provider.

AWS Transfer Family SFTP connectors

AWS Transfer Family SFTP connectors establish a relationship for sending files and messages between Amazon storage and an external partner, using the SFTP protocol. You can send files from Amazon S3 to an external, partner-owned destination. You can also use an SFTP connector to retrieve files from a partner's SFTP server.

Note

Currently, SFTP connectors can only be used to connect to remote SFTP servers that offer an internet accessible endpoint.

The following blog posts provides a reference architecture to build an MFT workflow using SFTP connectors, including encryption of files using PGP before sending them to a remote SFTP server using SFTP connectors: [Architecting secure and compliant managed file transfers with AWS Transfer Family SFTP connectors and PGP encryption](#).

View [AWS Transfer Family SFTP connectors](#) for a brief introduction to Transfer Family SFTP connectors.

Topics

- [Configure SFTP connectors](#)
- [Send and retrieve files by using an SFTP connector](#)
- [List contents of a remote directory](#)
- [Manage SFTP connectors](#)
- [Quotas for SFTP connectors](#)

Configure SFTP connectors

This topic describes how to create SFTP connectors, the security algorithms associated with them, how to store a secret to hold credentials, details about formatting the private key, and instructions for testing your connectors.

Topics

- [Create an SFTP connector](#)
- [Store a secret for use with an SFTP connector](#)
- [Generate and format the SFTP connector private key](#)
- [Test an SFTP connector](#)

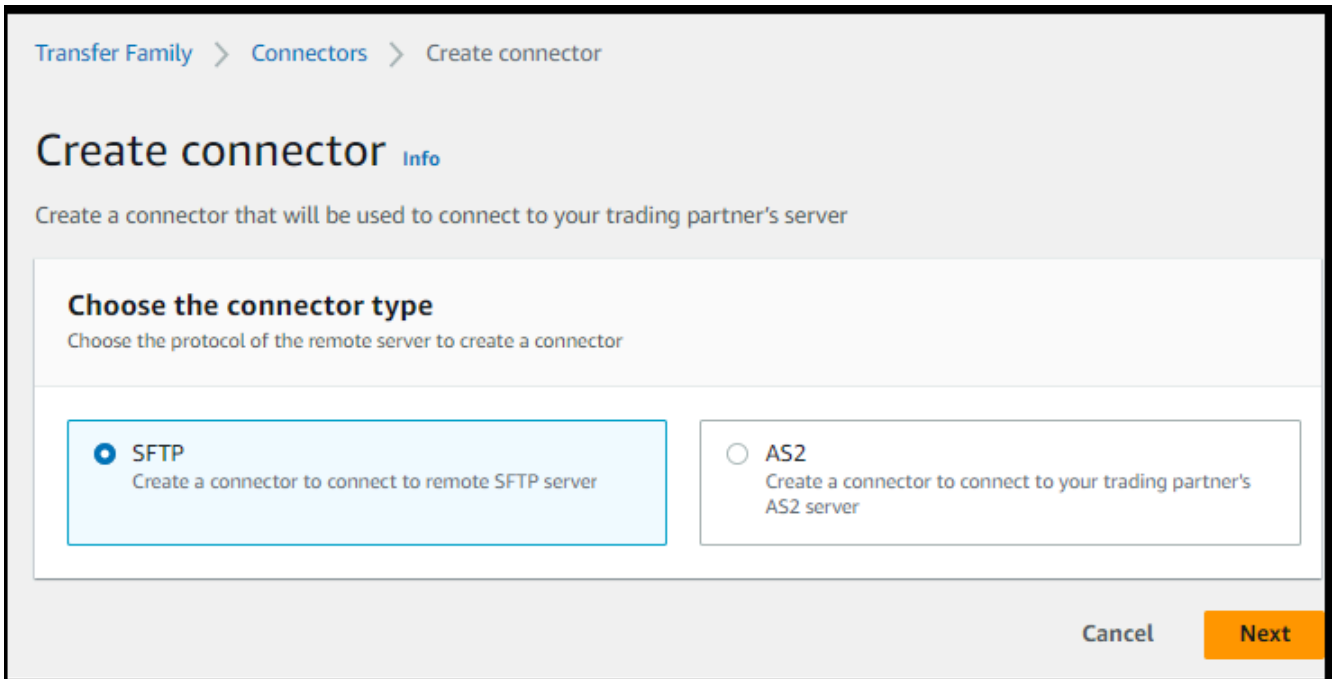
Create an SFTP connector

This procedure explains how to create SFTP connectors by using the AWS Transfer Family console or AWS CLI.

Console

To create an SFTP connector

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Connectors**, then choose **Create connector**.
3. Choose **SFTP** for the connector type to create an SFTP connector, and then choose **Next**.



4. In the **Connector configuration** section, provide the following information:
 - For the **URL**, enter the URL for a remote SFTP server. This URL must be formatted as `sftp://partner-SFTP-server-url`, for example `sftp://AnyCompany.com`.

Note

Optionally, you can provide a port number in your URL. The format is `sftp://partner-SFTP-server-url:port-number`. The default port number (when no port is specified) is port 22.

- For the **Access role**, choose the Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role to use.
- **Make sure that this role provides read and write access** to the parent directory of the file location that's used in the `StartFileTransfer` request.
- **Make sure that this role provides permission** for `secretsmanager:GetSecretValue` to access the secret.

Note

In the policy, you must specify the ARN for the secret. The ARN contains the secret name, but appends the name with six, random, alphanumeric characters. An ARN for a secret has the following format.

```
arn:aws:secretsmanager:region:account-id:secret:aws/transfer/SecretName-6RandomCharacters
```

- **Make sure this role contains a trust relationship** that allows the connector to access your resources when servicing your users' transfer requests. For details on establishing a trust relationship, see [To establish a trust relationship](#).

The following example grants the necessary permissions to access the *DOC-EXAMPLE-BUCKET* in Amazon S3, and the specified secret stored in Secrets Manager.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListingOfUserFolder",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
    }
  ],
}
```

```

    "Effect": "Allow",
    "Resource": [
      "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
    ]
  },
  {
    "Sid": "HomeDirObjectAccess",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:DeleteObject",
      "s3:DeleteObjectVersion",
      "s3:GetObjectVersion",
      "s3:GetObjectACL",
      "s3:PutObjectACL"
    ],
    "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
  },
  {
    "Sid": "GetConnectorSecretValue",
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/
transfer/SecretName-6RandomCharacters"
  }
]
}

```

Note

For the access role, the example grants access to a single secret. However, you can use a wildcard character, which can save work if you want to reuse the same IAM role for multiple users and secrets. For example, the following resource statement grants permissions for all secrets that have names beginning with `aws/transfer`.

```

"Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/
transfer/*"

```

You can also store secrets containing your SFTP credentials in another AWS account. For details on enabling cross-account secret access, see [Permissions to AWS Secrets Manager secrets for users in a different account](#).

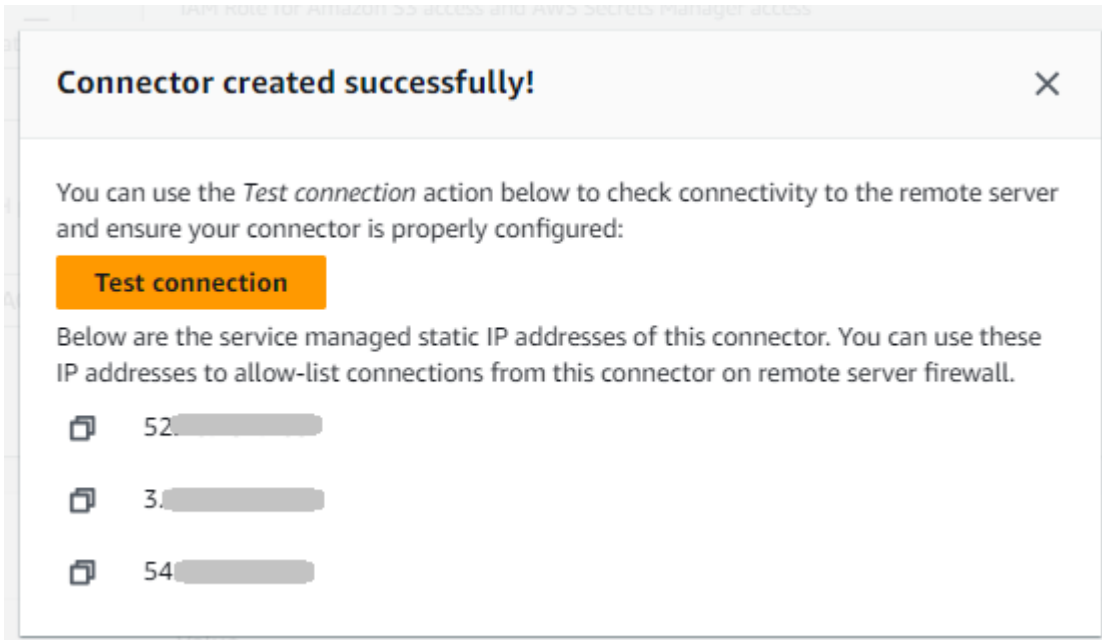
- (Optional) For the **Logging role**, choose the IAM role for the connector to use to push events to your CloudWatch logs. The following example policy lists the necessary permissions to log events for SFTP connectors.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "SFTPConnectorPermissions",
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogStream",
      "logs:DescribeLogStreams",
      "logs:CreateLogGroup",
      "logs:PutLogEvents"
    ],
    "Resource": [
      "arn:aws:logs:*:*:log-group:/aws/transfer/*"
    ]
  }]
}
```

5. In the **SFTP Configuration** section, provide the following information:
 - For **Connector credentials**, from the dropdown list, choose the name of a secret in AWS Secrets Manager that contains the SFTP user's private key or password. You must create a secret and store it in a specific manner. For details, see [Store a secret for use with an SFTP connector](#).
 - For **Trusted host keys**, paste in the public portion of the host key that is used to identify the external server. You can add more than one key, by choosing **Add trusted host key** to add an additional key. You can use the `ssh-keyscan` command against the SFTP server to retrieve the necessary key. For details about the format and type of trusted host keys that Transfer Family supports, see [SFTPConnectorConfig](#).
6. In the **Cryptographic algorithm options** section, choose a **Security policy** from the dropdown list in the **Security Policy** field. The security policy enables you to select the

cryptographic algorithms that your connector supports. For details on the available security policies and algorithms, see [Security policies for AWS Transfer Family SFTP connectors](#).

- (Optional) In the **Tags** section, for **Key** and **Value**, enter one or more tags as key-value pairs.
- After you have confirmed all of your settings, choose **Create connector** to create the SFTP connector. If the connector is created successfully, a screen appears with a list of the assigned static IP addresses and a **Test connection** button. Use the button to test the configuration for your new connector.



The **Connectors** page appears, with the ID of your new SFTP connector added to the list. To view the details for your connectors, see [View SFTP connector details](#).

CLI

You use the [create-connector](#) command to create a connector. To use this command to create an SFTP connector, you must provide the following information.

- The URL for a remote SFTP server. This URL must be formatted as `sftp://partner-SFTP-server-url`, for example `sftp://AnyCompany.com`.
- The access role. Choose the Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role to use.
 - Make sure that this role provides read and write access** to the parent directory of the file location that's used in the `StartFileTransfer` request.

- **Make sure that this role provides permission** for `secretsmanager:GetSecretValue` to access the secret.

Note

In the policy, you must specify the ARN for the secret. The ARN contains the secret name, but appends the name with six, random, alphanumeric characters. An ARN for a secret has the following format.

```
arn:aws:secretsmanager:region:account-id:secret:aws/  
transfer/SecretName-6RandomCharacters
```

- **Make sure this role contains a trust relationship** that allows the connector to access your resources when servicing your users' transfer requests. For details on establishing a trust relationship, see [To establish a trust relationship](#).

The following example grants the necessary permissions to access the *DOC-EXAMPLE-BUCKET* in Amazon S3, and the specified secret stored in Secrets Manager.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowListingOfUserFolder",  
      "Action": [  
        "s3:ListBucket",  
        "s3:GetBucketLocation"  
      ],  
      "Effect": "Allow",  
      "Resource": [  
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET"  
      ]  
    },  
    {  
      "Sid": "HomeDirObjectAccess",  
      "Effect": "Allow",  
      "Action": [  
        "s3:PutObject",  
        "s3:GetObject",  
        "s3:DeleteObject",  
        "s3:DeleteObjectVersion",  
      ]  
    }  
  ]  
}
```

```

        "s3:GetObjectVersion",
        "s3:GetObjectACL",
        "s3:PutObjectACL"
    ],
    "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
},
{
    "Sid": "GetConnectorSecretValue",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue"
    ],
    "Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/transfer/SecretName-6RandomCharacters"
}
]
}

```

Note

For the access role, the example grants access to a single secret. However, you can use a wildcard character, which can save work if you want to reuse the same IAM role for multiple users and secrets. For example, the following resource statement grants permissions for all secrets that have names beginning with `aws/transfer`.

```
"Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/transfer/*"
```

You can also store secrets containing your SFTP credentials in another AWS account. For details on enabling cross-account secret access, see [Permissions to AWS Secrets Manager secrets for users in a different account](#).

- (Optional) Choose the IAM role for the connector to use to push events to your CloudWatch logs. The following example policy lists the necessary permissions to log events for SFTP connectors.

```

{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "SFTPConnectorPermissions",

```

```

    "Effect": "Allow",
    "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:CreateLogGroup",
        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/transfer/*"
    ]
  }]
}

```

- Provide the following SFTP configuration information.
 - The ARN of a secret in AWS Secrets Manager that contains the SFTP user's private key or password.
 - The public portion of the host key that is used to identify the external server. You can provide multiple trusted host keys if you like.

The easiest way to provide the SFTP information is to save it to a file. For example, copy the following example text to a file named `testSFTPConfig.json`.

```

// Listing for testSFTPConfig.json
{
  "UserSecretId": "arn:aws::secretsmanager:us-east-2:123456789012:secret:aws/transfer/example-username-key",
  "TrustedHostKeys": [
    "sftp.example.com ssh-rsa AAAAbbbb...EEEE="
  ]
}

```

- Specify a security policy for your connector, entering the security policy name.

Note

The `SecretId` can be either the entire ARN or the name of the secret (*example-username-key* in the previous listing).

Then run the following command to create the connector.

```
aws transfer create-connector --url "sftp://partner-SFTP-server-url" \  
--access-role your-IAM-role-for-bucket-access \  
--logging-role arn:aws:iam::your-account-id:role/service-role/  
AWSTransferLoggingAccess \  
--sftp-config file:///path/to/testSFTPConfig.json  
--security-policy-name security-policy-name
```

Store a secret for use with an SFTP connector

You can use Secrets Manager to store user credentials for your SFTP connectors. When you create your secret, you must provide a username. Additionally, you can provide either a password, a private key, or both. For details, see [Quotas for SFTP connectors](#).

Note

When you store secrets in Secrets Manager, your AWS account incurs charges. For information about pricing, see [AWS Secrets Manager Pricing](#).


To store user credentials in Secrets Manager for an SFTP connector

1. Sign in to the AWS Management Console and open the AWS Secrets Manager console at <https://console.aws.amazon.com/secretsmanager/>.
2. In the left navigation pane, choose **Secrets**.
3. On the **Secrets** page, choose **Store a new secret**.
4. On the **Choose secret type** page, for **Secret type**, choose **Other type of secret**.
5. In the **Key/value pairs** section, choose the **Key/value** tab.
 - **Key** – Enter **Username**.
 - **value** – Enter the name of the user that is authorized to connect to the partner' server.
6. If you want to provide a password, choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.

Choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.

- **Key** – Enter **Password**.
- **value** – Enter the password for the user.

7. If you want to provide a private key, see [Generate and format the SFTP connector private key](#), which describes how to enter private key data.

 **Note**

The private key data that you enter must correspond to the public key that is stored for this user in the remote SFTP server.

8. Choose **Next**.
9. On the **Configure secret** page, enter a name and description for your secret. We recommend that you use a prefix of **aws/transfer/** for the name. For example, you could name your secret **aws/transfer/connector-1**.
10. Choose **Next**, and then accept the defaults on the **Configure rotation** page. Then choose **Next**.
11. On the **Review** page, choose **Store** to create and store the secret.

Generate and format the SFTP connector private key

Complete details for generating a public/private key pair are described in [Creating SSH keys on macOS, Linux, or Unix](#).

As an example, to generate a private key for use with SFTP connectors, the following sample command produces the correct type of key (replace *key_name* with the actual file name for your key pair):

```
ssh-keygen -t rsa -b 4096 -m PEM -f key_name -N ""
```

 **Note**

When you create your key pair for use with SFTP connectors, do not use a passphrase. An empty passphrase is necessary for the SFTP configuration to function correctly.

This command creates an RSA key pair, with a key size of 4096 bits. The key is generated in the legacy PEM format, which is required by Transfer Family for use with the SFTP connector secret. The keys are saved in *key_name* (private key) and *key_name*.pub (public key) in the current directory: that is, the directory where you run the `ssh-keygen` command.

Note

Transfer Family does not support the OpenSSH format (-----BEGIN OPENSSSH PRIVATE KEY-----) for the keys used for your SFTP connector. The key must be in *legacy PEM* format (-----BEGIN RSA PRIVATE KEY----- or -----BEGIN EC PRIVATE KEY-----). You can use the **ssh-keygen** tool to convert your key, by supplying the **-m PEM** option when you run the command.

After you generate the key, you must make sure that the private key is formatted with embedded newline characters ("\n") in JSON format.

Use a command to convert your existing private key into the correct format—JSON format with embedded newline characters. Here we provide examples for jq and Powershell. You can use any tool or command that you'd like to convert the private key into JSON format with embedded newline characters.

jq command

This example uses the jq command, which is available for download from [Download jq](#).

```
jq -sR . path-to-private-key-file
```

For example, if your private key file is located in `~/.ssh/my_private_key`, the command is as follows.

```
jq -sR . ~/.ssh/my_private_key
```

This outputs the key in the correct format (with embedded newline characters) to standard output.

PowerShell

If you are using Windows, you can use PowerShell to convert the key to the correct format. The following Powershell command converts the private key to the correct format.

```
Get-Content -Raw path-to-private-key-file | ConvertTo-Json
```

To add private key data to the secret for use with SFTP connectors

1. In the Secrets Manager console, when storing **Other type of Secret**, choose the **Plaintext** tab. The text should be empty, with only an opening and closing brace, {}.
2. Paste in your username, private key data, and/or password using the following format. For your private key data, paste the output from the command that you ran in step 1.

```
{"Username": "SFTP-USER", "Password": "SFTP-USER-PASSWORD", "PrivateKey": "PASTE-PRIVATE-KEY-DATA-HERE"}
```



If you paste the private key data correctly, you should see the following upon selecting the **Key/value** tab. Notice that the private key data is displayed line-by-line, rather than as a continuous string of text.

Secret value [Info](#)
Retrieve and view the secret value.

Key/value | Plaintext

Secret key	Secret value
Username	SFTP-USER
Password	SFTP-USER-PASSWORD
PrivateKey	-----BEGIN RSA PRIVATE KEY----- MITI... g... a... U... G... g... T... a... I... W... I... A... e... 5... 7... H... i... By...

- Continue the procedure in [Store a secret for use with an SFTP connector](#) at step 8, and follow that procedure until the end.

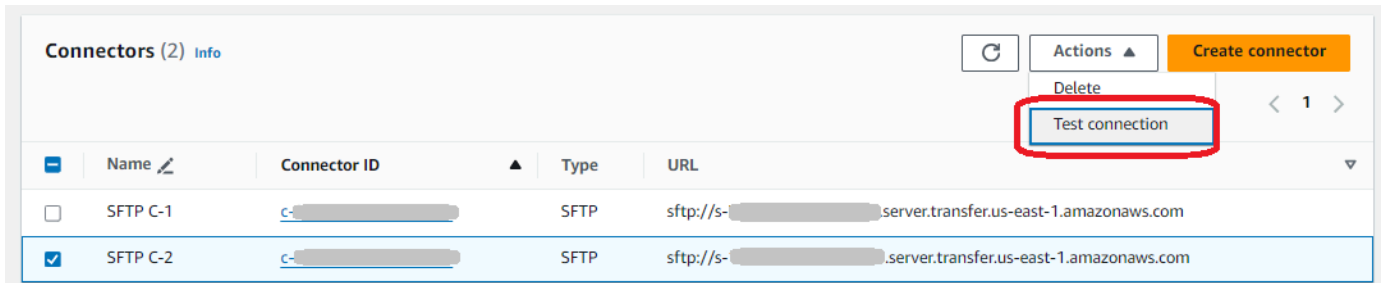
Test an SFTP connector

After you create an SFTP connector, we recommend that you test it before you attempt to transfer any files using your new connector.

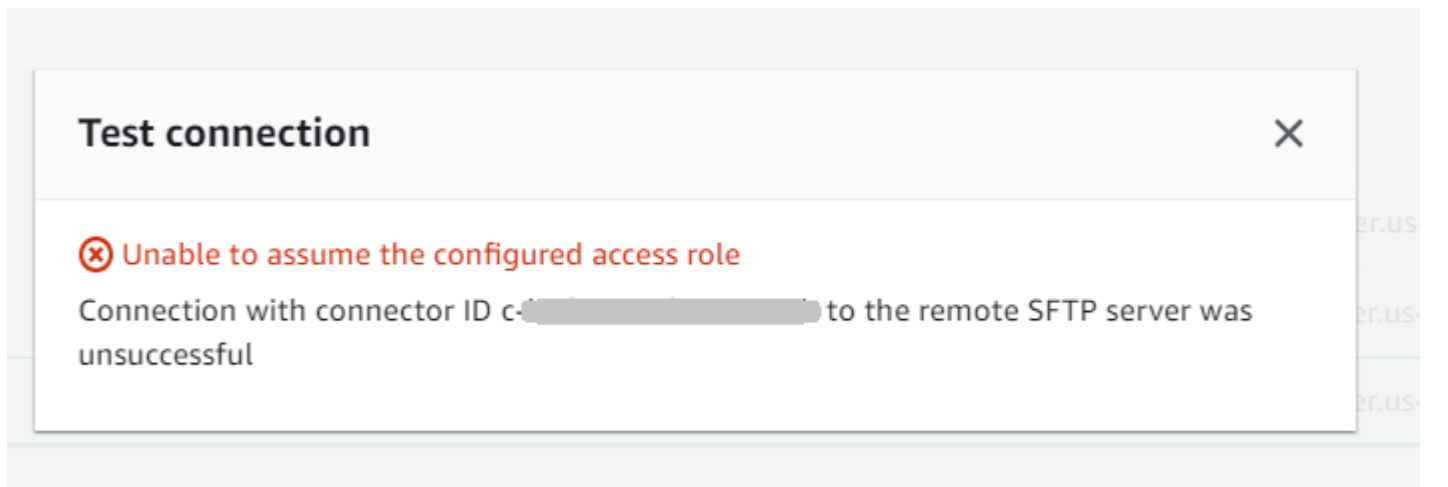
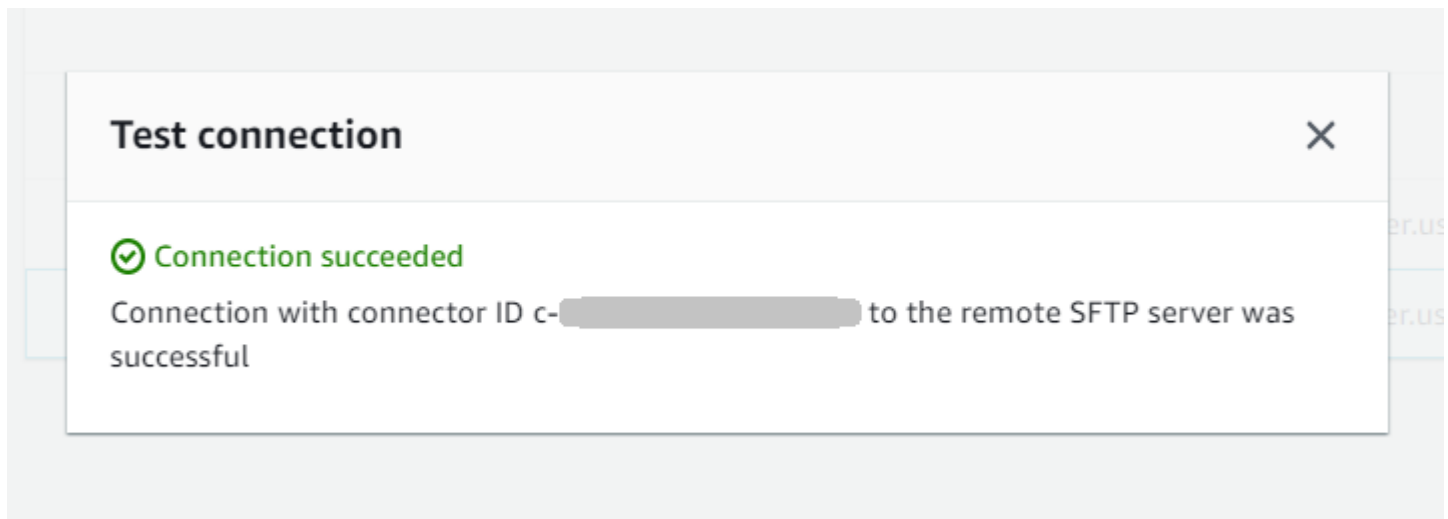
To test an SFTP connector

- Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
- In the left navigation pane, choose **Connectors**, and select a connector.

3. From the **Actions** menu, choose **Test connection**.



The system returns a message, indicating whether the test passes or fails. If the test fails, the system provides an error message based on the reason the test failed.



Note

To use the API to test your connector, see the [TestConnection](#) API documentation.

Send and retrieve files by using an SFTP connector

SFTP connectors extend the capabilities of AWS Transfer Family to communicate with remote servers both in the cloud and on-premises. You can integrate data that's generated and stored in remote sources with your AWS hosted data warehouses for analytics, business applications, reporting, and auditing.

To initiate a file transfer to a remote SFTP server, you use the [StartFileTransfer](#) API operation, which uses SFTP connectors to perform the transfer. Each `StartFileTransfer` request can contain 10 distinct paths.

You can monitor your file transfers by checking your server logs. Connector activity is logged to log streams that have the format of `aws/transfer/connector-id`, for example, `aws/transfer/c-1234567890abcdef0`. If you don't see any logs for your connector, make sure that you have specified a logging role with the correct permissions for your connector.

For details on creating connectors, see [Configure SFTP connectors](#).

To send and retrieve files by using an SFTP connector, you use the `start-file-transfer` AWS Command Line Interface (AWS CLI) command. You specify the following parameters, depending on whether you're sending files (outbound transfers) or receiving files (inbound transfers).

- **Outbound transfers**

- `send-file-paths` contains from one to ten source file paths, for files to transfer to the partner's SFTP server.
- `remote-directory-path` is the remote path to send a file to on the customer's SFTP server.

- **Inbound transfers**

- `retrieve-file-paths` contains from one to ten remote paths. Each path specifies a location for transferring files from the partner's SFTP server to your Transfer Family server.
- `local-directory-path` is the Amazon S3 location (bucket and optional prefix) where your files are stored.

To send files, you specify the `send-file-paths` and `remote-directory-path` parameters. You can specify up to 10 files for the `send-file-paths` parameter. The following example command sends the files named `/DOC-EXAMPLE-SOURCE-BUCKET/file1.txt` and `/DOC-EXAMPLE-SOURCE-BUCKET/file2.txt`, located in Amazon S3 storage, to the `/tmp` directory on your

partner's SFTP server. To use this example command, replace the *DOC-EXAMPLE-SOURCE-BUCKET* with your own bucket.

```
aws transfer start-file-transfer --send-file-paths /DOC-EXAMPLE-SOURCE-BUCKET/
file1.txt /DOC-EXAMPLE-SOURCE-BUCKET/file2.txt \
  --remote-directory-path /tmp --connector-id c-1111AAAA2222BBBB3 --region us-east-2
```

To receive files, you specify the `retrieve-file-paths` and `local-directory-path` parameters. The following example retrieves the files `/my/remote/file1.txt` and `/my/remote/file2.txt` on the partner's SFTP server, and places it in the Amazon S3 location `/DOC-EXAMPLE-BUCKET/prefix`. To use this example command, replace the *user input placeholders* with your own information.

```
aws transfer start-file-transfer --retrieve-file-paths /my/remote/file1.txt /my/
remote/file2.txt \
  --local-directory-path /DOC-EXAMPLE-BUCKET/prefix --connector-id c-2222BBBB3333CCCC4
--region us-east-2
```

The previous examples specify absolute paths on the SFTP server. You can also use relative paths: that is, paths that are relative to the SFTP user's home directory. For example, if the SFTP user is `marymajor` and their home directory on the SFTP server is `/users/marymajor/`, the following command sends `/DOC-EXAMPLE-SOURCE-BUCKET/file1.txt` to `/users/marymajor/test-connectors/file1.txt`

```
aws transfer start-file-transfer --send-file-paths /DOC-EXAMPLE-SOURCE-BUCKET/file1.txt
\
  --remote-directory-path test-connectors --connector-id c-2222BBBB3333CCCC4 --
region us-east-2
```

List contents of a remote directory

Before you retrieve files from a remote SFTP server, you can retrieve the contents of a directory on the remote SFTP server. To do this, you use the [StartDirectoryListing](#) API call.

The following example lists the contents of the home folder on the remote SFTP server, which is specified in the connector's configuration. The results are placed into the Amazon S3 location `/DOC-EXAMPLE-BUCKET/connector-files`, and into a file named `c-AAAA1111BBBB2222C-6666abcd-11aa-22bb-cc33-0000aaaa3333.json`.

```
aws transfer start-directory-listing \
  --connector-id c-AAAA1111BBBB2222C \
  --output-directory-path /DOC-EXAMPLE-BUCKET/example/connector-files \
  --remote-directory-path /home
```

This AWS CLI command returns a listing ID and the name of the file that contains the results.

```
{
  "ListingId": "6666abcd-11aa-22bb-cc33-0000aaaa3333",
  "OutputFileName": "c-AAAA1111BBBB2222C-6666abcd-11aa-22bb-cc33-0000aaaa3333.json"
}
```

Note

The naming convention for the output file is *connector-ID-listing-ID.json*.

The JSON file contains the following information:

- **filePath**: the complete path of a remote file, relative to the directory of the listing request for your SFTP connector on the remote server.
- **modifiedTimestamp**: the last time the file was modified, in seconds, Coordinated Universal Time (UTC) format. This field is optional. If the remote file attributes don't contain a timestamp, it is omitted from the file listing.
- **size**: the size of the file, in bytes. This field is optional. If the remote file attributes don't contain a file size, it is omitted from the file listing.
- **path**: the complete path of a remote directory, relative to the directory of the listing request for your SFTP connector on the remote server.
- **truncated**: a flag indicating whether the list output contains all of the items contained in the remote directory or not. If your **truncated** output value is true, you can increase the value provided in the optional **max-items** input attribute to be able to list more items (up to the maximum allowed list size of 10,000 items).

The following is an example of the contents of the output file (*c-AAAA1111BBBB2222C-6666abcd-11aa-22bb-cc33-0000aaaa3333.json*), where the remote directory contains two files and two sub-directories (paths).

```
{
  "files": [
    {
      "filePath": "/home/what.txt",
      "modifiedTimestamp": "2024-01-30T20:34:54Z",
      "size" : 2323
    },
    {
      "filePath": "/home/how.pgp",
      "modifiedTimestamp": "2024-01-30T20:34:54Z",
      "size" : 4691
    }
  ],
  "paths": [
    {
      "path": "/home/magic"
    },
    {
      "path": "/home/aws"
    }
  ],
  "truncated": "false"
}
```

Manage SFTP connectors

This topic describes how to view and update SFTP connectors, and lists quotas that are relevant for SFTP connectors.

Note

Each connector is automatically assigned static IP addresses that remain unchanged over the lifetime of the connector. This allows you to connect with remote SFTP servers that only accept inbound connections from known IP addresses. Your connectors are assigned a set of static IP addresses that are shared by all connectors using the same protocol (SFTP or AS2) in your AWS account.

Topics

- [Update SFTP connectors](#)

- [View SFTP connector details](#)

Update SFTP connectors

To change the existing parameter values for your connectors, you can run the `update-connector` command. The following command updates the secret for the connector *connector-id*, in the Region *region-id* to *secret-ARN*. To use this example command, replace the *user input placeholders* with your own information.

```
aws transfer update-connector --sftp-config '{"UserSecretId":"secret-ARN"}' \  
  --connector-id connector-id --region region-id
```

View SFTP connector details

You can find a list of details and properties for an SFTP connector in the AWS Transfer Family console.

To view connector details

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Connectors**.
3. Choose the identifier in the **Connector ID** column to see the details page for the selected connector.

You can change the properties for the SFTP connector by choosing **Edit** on the connector details page.

Note

You can get much of this information, albeit in a different format, by running the following AWS Command Line Interface (AWS CLI) command. To use this example command, replace the *user input placeholders* with your own information.

```
aws transfer describe-connector --connector-id your-connector-id
```

For more information, see [DescribeConnector](#) in the API reference.

Quotas for SFTP connectors

The following quotas are in place for SFTP connectors.

Note

More service quotas for SFTP connectors are listed in [AWS Transfer Family endpoints and quotas](#) in the *Amazon Web Services General Reference*.

SFTP connector quotas

Name	Default	Adjustable
Maximum test connection transactions per second (TPS)	1 request per second, per account	No
Maximum queue size for pending file transfers	1000	No
Maximum file size	50 gibibytes (GiB)	No
Maximum transfer time per file	6 hours	No
Maximum request wait time per file	6 hours	No
Maximum bandwidth for connectors per account (both SFTP and AS2 connectors contribute to this value)	50 MBps	No

For storing the credentials for SFTP connectors, there are quotas associated with each Secrets Manager secret. If you use the same secret to store multiple types of keys, for multiple purposes, you may encounter these quotas.

- Total length for a single secret: 12,000 characters
- Maximum length of the **Password** string: 1024 characters

- Maximum length of the **PrivateKey** string: 8192 characters
- Maximum length of the **Username** string: 100 characters

AWS Transfer Family for AS2

Applicability Statement 2 (AS2) is an RFC-defined file-transmission specification that includes strong message protection and verification mechanisms. The AS2 protocol is critical to workflows with compliance requirements that rely on having data protection and security features built into the protocol.

Note

AS2 for Transfer Family is [Drummond certified](#).

Customers in industries such as retail, life sciences, manufacturing, financial services, and utilities that rely on AS2 for supply chain, logistics, and payments workflows can use AWS Transfer Family AS2 endpoints to securely transact with their business partners. The transacted data is natively accessible in AWS for processing, analysis, and machine learning. This data is also available for integrations with enterprise resource planning (ERP) and customer relationship management (CRM) systems that run on AWS. With AS2, customers can run their business-to-business (B2B) transactions at scale in AWS while maintaining existing business partner integrations and compliance.

If you are a Transfer Family customer who wants to exchange files with a partner who has a configured AS2-enabled server, the setup involves generating one public-private key pair for encryption and another for signing and exchanging the public keys with the partner.

Transfer Family provides a workshop that you can attend, in which you can configure a Transfer Family endpoint with AS2 enabled, and a Transfer Family AS2 connector. You can view the details for this workshop [here](#).

Protecting an AS2 payload in transit typically involves the use of Cryptographic Message Syntax (CMS) and commonly uses encryption and a digital signature to provide data protection and peer authentication. A signed Message Disposition Notice (MDN) response payload provides verification (non-repudiation) that a message was received and successfully decrypted.

Transport of these CMS payloads and MDN responses occurs over HTTP.

Note

HTTPS AS2 server endpoints are not currently supported. TLS termination is currently the responsibility of the customer.

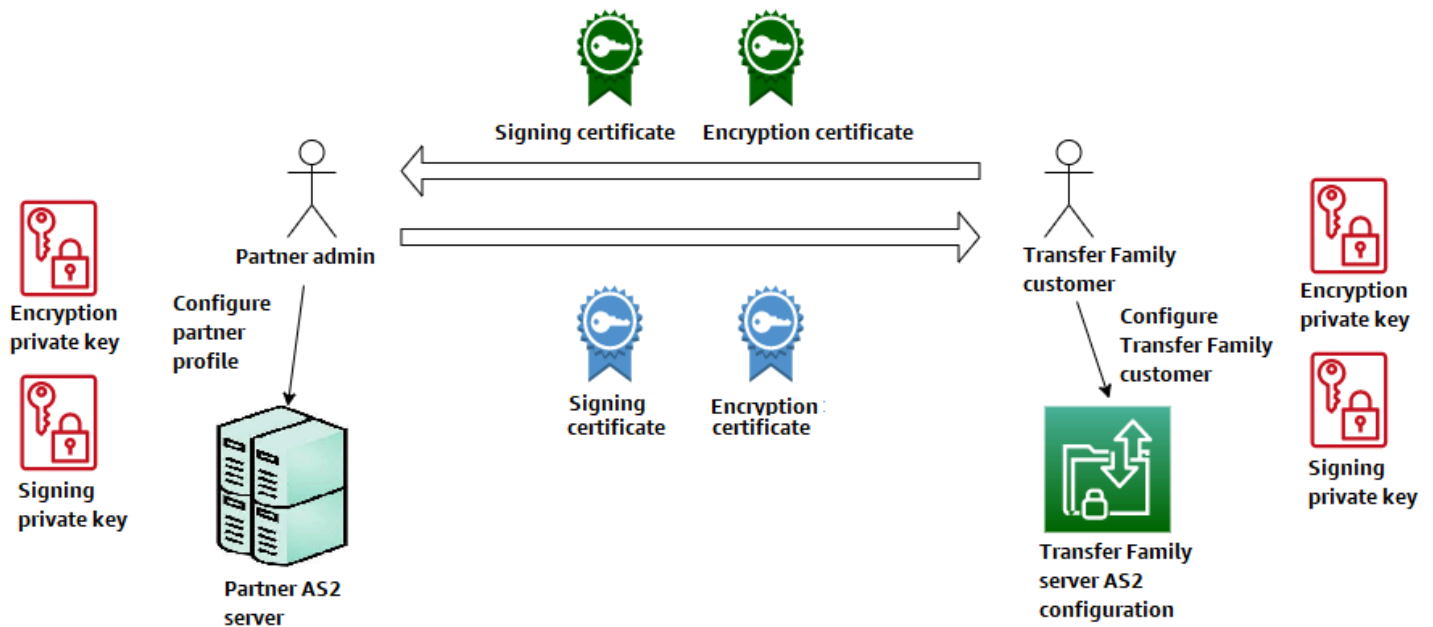
For a detailed, step-by-step walkthrough of setting up an Applicability Statement 2 (AS2) configuration, see the tutorial, [Setting up an AS2 configuration](#).

Topics

- [AS2 use cases](#)
- [Configuring AS2](#)
- [Configure AS2 connectors](#)
- [Manage AS2 partners](#)
- [Sending and receiving AS2 messages](#)
- [Monitoring AS2 usage](#)

AS2 use cases

If you are an AWS Transfer Family customer who wants to exchange files with a partner who has a configured AS2 server, the most complex part of the setup involves generating one public-private key pair for encryption and another for signing and exchanging the public keys with the partner.



Consider the following variations for using AWS Transfer Family with AS2.

Note

Trading partner is the partner associated with that partner profile.
All mentions of *MDN* in the following table assume *signed MDNs*.

AS2 use cases

Inbound-only use cases

- **Transfer encrypted AS2 messages from a trading partner to a Transfer Family server.**

In this case, you do the following:

1. Create profiles for your trading partner and yourself.
2. Create a Transfer Family server that uses the AS2 protocol.
3. Create an agreement and add it to your server.
4. Import a certificate with a private key and add it to your profile, and then import the public key to your partner profile for encryption.
5. After you have these items, send the public key for your certificate to your trading partner.

Now your partner can send you encrypted messages and you can decrypt them and store them in your Amazon S3 bucket.

- **Transfer encrypted AS2 messages from a trading partner to a Transfer Family server and add signing.**

In this scenario, you are still doing only inbound transfers, but now you want to have your partner sign the messages that they send. In this case, import the trading partner's signing public key (as a signing certificate added to your partner's profile).

- **Transfer encrypted AS2 messages from a trading partner to a Transfer Family server and add signing and sending an MDN response.**

In this scenario, you are still doing only inbound transfers, but now, in addition to receiving signed payloads, your trading partner wants to receive a signed MDN response.

1. Import your public and private signing keys (as a signing certificate to your profile).
2. Send the public signing key to your trading partner.

Outbound-only use cases

- **Transfer encrypted AS2 messages from a Transfer Family server to a trading partner.**

This case is similar to the inbound-only transfer use case, except that instead of adding an agreement to your AS2 server, you create a connector. In this case, you import your trading partner's public key to their profile.

- **Transfer encrypted AS2 messages from a Transfer Family server to a trading partner and add signing.**

You are still doing only outbound transfers, but now your trading partner wants you to sign the message that you send to them.

1. Import your signing private key (as a signing certificate added to your profile).
2. Send your trading partner your public key.

- **Transfer encrypted AS2 messages from a Transfer Family server to a trading partner and add signing and send an MDN response.**

You are still doing only outbound transfers, but now, in addition to sending signed payloads, you want to receive a signed MDN response from your trading partner.

1. Your trading partner sends you their public signing key.
2. Import your trading partner's public key (as a signing certificate added to your partner profile).

Inbound and outbound use cases

- **Transfer encrypted AS2 messages in both directions between a Transfer Family server and a trading partner.**

In this case, you do the following:

1. Create profiles for your trading partner and yourself.
2. Create a Transfer Family server that uses the AS2 protocol.
3. Create an agreement and add it to your server.
4. Create a connector.
5. Import a certificate with a private key and add it to your profile, and then import the public key to your partner profile for encryption.
6. Receive a public key from your trading partner and add it to their profile for encryption.
7. After you have these items, send the public key for your certificate to your trading partner.

Now you and your trading partner can exchange encrypted messages, and you can both decrypt them. You can store the messages that you receive in your Amazon S3 bucket, and your partner can decrypt and store the messages that you send to them.

- **Transfer encrypted AS2 messages in both directions between a Transfer Family server and a trading partner and add signing.**

Now you and your partner want signed messages.

1. Import your signing private key (as a signing certificate added to your profile).
2. Send your trading partner your public key.
3. Import your trading partner's signing public key and add it to their profile.

- **Transfer encrypted AS2 messages in both directions between a Transfer Family server and a trading partner and add signing and send an MDN response.**

Now, you want to exchange signed payloads, and both you and your trading partner want MDN responses.

1. Your trading partner sends you their public signing key.
2. Import your trading partner's public key (as a signing certificate to your partner profile).
3. Send your public key to your trading partner.

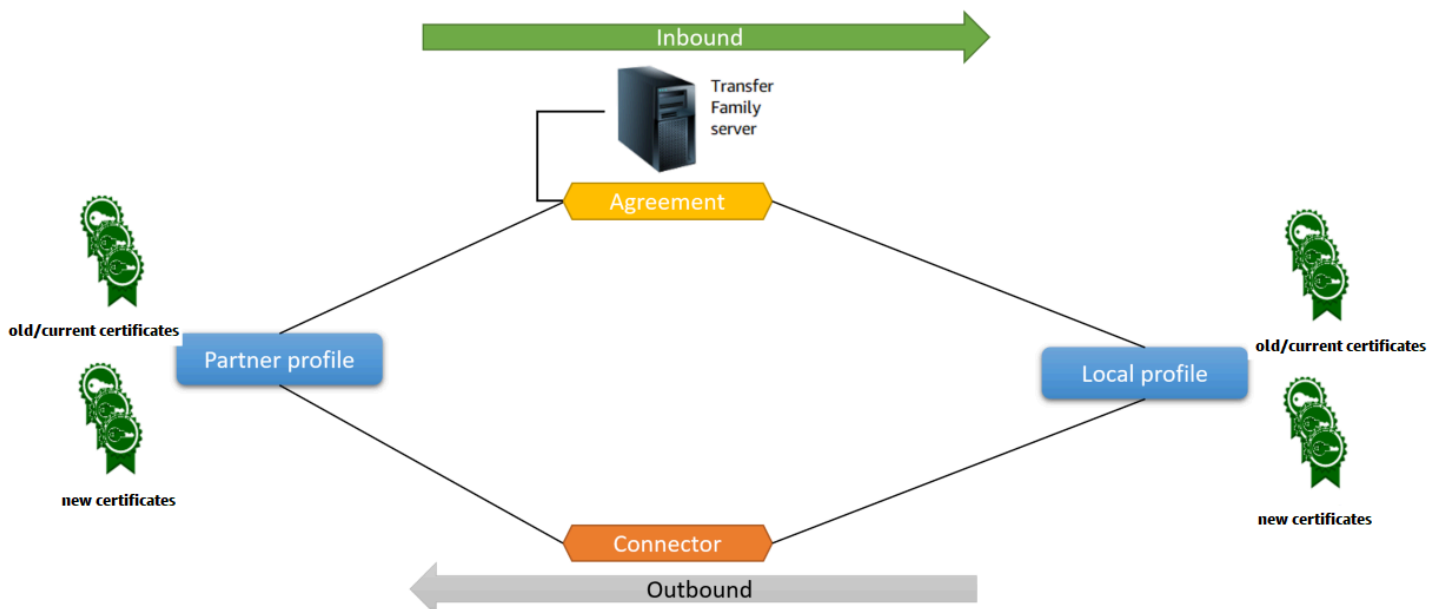
Configuring AS2

To create an AS2-enabled server, you must also specify the following components:

- **Agreements** – Bilateral trading partner *agreements*, or partnerships, define the relationship between the two parties that are exchanging messages (files). To define an agreement, Transfer Family combines server, local profile, partner profile, and certificate information. Transfer Family AS2-inbound processes use agreements.
- **Certificates** – *Public key (X.509) certificates* are used in AS2 communication for message encryption and verification. Certificates are also used for connector endpoints.
- **Local profiles and partner profiles** – A *local profile* defines the local (AS2-enabled Transfer Family server) organization or "party." Similarly, a *partner profile* defines the remote partner organization, external to Transfer Family.

While not required for all AS2-enabled servers, for outbound transfers, you need a **connector**. A connector captures the parameters for an outbound connection. The connector is required for sending files to a customer's external, non AWS server.

The following diagram shows the relationship between the AS2 objects involved in the inbound and outbound processes.



For an end-to-end example AS2 configuration, see [Setting up an AS2 configuration](#).

Topics

- [Create an AS2 server using the Transfer Family console](#)
- [Use a template to create a demo Transfer Family AS2 stack](#)
- [AS2 configurations](#)
- [AS2 quotas and limitations](#)
- [AS2 features and capabilities](#)

Create an AS2 server using the Transfer Family console

This procedure explains how to create an AS2-enabled server by using the Transfer Family console. If you want to use the AWS CLI instead, see [the section called “Step 2: Create a Transfer Family server that uses the AS2 protocol”](#).

Note

You can attach a file-processing workflow to a Transfer Family server that uses the AS2 protocol: however, AS2 messages don't execute workflows attached to the server.

To create an AS2-enabled server

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Servers**, and then choose **Create server**.
3. On the **Choose protocols** page, select **AS2 (Applicability Statement 2)**, and then choose **Next**.
4. On the **Choose an identity provider** page, choose **Next**.

Note

For AS2, you cannot choose an identity provider because basic authentication is not supported for the AS2 protocol. Instead, you control access through virtual private cloud (VPC) security groups.

5. On the **Choose an endpoint** page, do the following:

Choose an endpoint

Endpoint configuration [Info](#)

Endpoint type
Select whether the endpoint will be publicly accessible or hosted inside your VPC

Publicly accessible
Accessible over the internet

VPC hosted [Info](#)
Access controlled using Security Groups

Access [Info](#)

Internal

Internet Facing

VPC
Select a VPC ID

FIPS Enabled
Select whether the endpoint should comply with Federal Information Processing Standards (FIPS)

FIPS Enabled endpoint

- a. For **Endpoint type**, choose **VPC hosted** to host your server's endpoint. For information about setting up your VPC-hosted endpoint, see [Create a server in a virtual private cloud](#).

Note


Publicly accessible endpoints are not supported for the AS2 protocol. To make your VPC endpoint accessible over the internet, choose **Internet Facing** under **Access**, and then supply your Elastic IP addresses.

- b. For **Access**, choose one of the following options:

- **Internal** – Choose this option to provide access from within your VPC and VPC-connected environments, such as an on-premises data center over AWS Direct Connect or VPN.
- **Internet Facing** – Choose this option to provide access over the internet and from within your VPC and VPC-connected environments, such as an on-premises data center over AWS Direct Connect or VPN.

If you choose **Internet Facing**, supply your Elastic IP addresses when prompted.

- c. For **VPC**, either choose an existing VPC or choose **Create VPC** to create a new VPC.
- d. For **FIPS Enabled**, keep the **FIPS Enabled endpoint** check box cleared.


 **Note**

FIPS-enabled endpoints are not supported for the AS2 protocol.

- e. Choose **Next**.
6. On the **Choose a domain** page, choose **Amazon S3** to store and access your files as objects by using the selected protocol.

Choose **Next**.

7. On the **Configure additional details** page, choose the settings that you need.

 **Note**

If you are configuring any other protocols along with AS2, all of the additional detail settings apply. However, for the AS2 protocol, the only settings that apply are those in the **CloudWatch logging** and **Tags** sections.

Even though setting up a CloudWatch logging role is optional, we highly recommend setting it up so that you can see the status of your messages and troubleshoot configuration issues.

8. On the **Review and create** page, review your choices to make sure they are correct.
 - If you want to edit any of your settings, choose **Edit** next to the step that you want to change.

Note

If you edit a step, we recommend that you review each step after the step that you chose to edit.

- If you have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

It can take several minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations for your users.

Use a template to create a demo Transfer Family AS2 stack

We supply a self-contained, AWS CloudFormation template to quickly create an AS2-enabled Transfer Family server. The template configures the server with a public Amazon VPC endpoint, certificates, local and partner profiles, an agreement, and a connector.


Before using this template, note the following:

- If you create a stack from this template, you will be billed for the AWS resources that are used.
- The template creates multiple certificates and places them in AWS Secrets Manager to store them securely. You can delete these certificates from Secrets Manager if you want, because you're charged for using this service. Deleting these certificates in Secrets Manager doesn't delete them from the Transfer Family server. Therefore, the functionality of the demo stack isn't affected. However, for certificates that you're going to use with a production AS2 server, you might want to use Secrets Manager to manage and periodically rotate your stored certificates.
- We recommend that you use the template as a base only, and mainly for demonstration purposes. If you want to use this demo stack in production, we recommend that you modify the template's YAML code to create a more robust stack. For example, create production-level certificates, and create an AWS Lambda function that you can use in production.

To create an AS2-enabled Transfer Family server from a CloudFormation template


1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. In the left navigation pane, choose **Stacks**.
3. Choose **Create stack**, and then choose **With new resources (standard)**.

4. In the **Prerequisite - Prepare template** section, choose **Template is ready**.
5. Copy this link, [AS2 demo template](#), and paste it into the **Amazon S3 URL** field.
6. Choose **Next**.
7. On the **Specify stack details** page, name your stack, and then specify the following parameters:
 - Under **AS2**, enter values for **Local AS2 ID** and **Partner AS2 ID**, or accept the defaults, `local` and `partner`, respectively.
 - Under **Network**, enter a value for **Security group ingress CIDR IP**, or accept the default, `0.0.0.0/0`.
8. Choose **Next**. On the **Configure stack options** page, choose **Next** again.
9. Review the details for the stack that you're creating, and then choose **Create stack**.

 **Note**

This value, in CIDR format, specifies which IP addresses are allowed for incoming traffic to the AS2 server. The default value, `0.0.0.0/0`, allows all IP addresses.

- Under **General**, enter a value for **Prefix**, or accept the default, `transfer-as2`. This prefix is placed before any resource names that are created by the stack. For example, if you use the default prefix, your Amazon S3 bucket is named `transfer-as2-DOC-EXAMPLE-BUCKET`.

 **Note**

At the bottom of the page, under **Capabilities**, you must acknowledge that AWS CloudFormation might create AWS Identity and Access Management (IAM) resources.

After the stack is created, you can send a test AS2 message from the partner server to your local Transfer Family server by using the AWS Command Line Interface (AWS CLI). A sample AWS CLI command for sending a test message is created along with all of the other resources in the stack.

To use this sample command, go to the **Outputs** tab of your stack, and copy the **TransferExampleAs2Command**. You can then run the command by using the AWS CLI. If you haven't already installed the AWS CLI, see [Installing or updating the latest version of the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

The sample command has the following format:

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key test.txt && aws transfer start-file-transfer --region aws-region --connector-id TransferConnectorId --send-file-paths /DOC-EXAMPLE-BUCKET/test.txt
```

 **Note**

Your version of this command contains the actual values for the *DOC-EXAMPLE-BUCKET* and *TransferConnectorId* resources in your stack.

This sample command consists of two separate commands that are chained together by using the `&&` string.

The first command creates a new, empty text file in your bucket:

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key test.txt
```

Then, the second command uses the connector to send the file from the partner profile to the local profile. The Transfer Family server has an agreement set up that allows the local profile to accept messages from the partner profile.

```
aws transfer start-file-transfer --region aws-region --connector-id TransferConnectorId --send-file-paths /DOC-EXAMPLE-BUCKET/test.txt
```

After you run the command, you can go to your Amazon S3 bucket (*DOC-EXAMPLE-BUCKET*) and view the contents. If the command is successful, you should see the following objects in your bucket:

- *processed/* – This folder contains a JSON file that describes the transferred file and the MDN response.
- *processing/* – This folder temporarily contains files as they are being processed, but after a transfer is completed, this folder should be empty.
- *server-id/* – This folder is named based on your Transfer Family server ID. It contains *from-partner* (this folder is dynamically named, based on the partner's AS2 ID), which itself contains *failed/*, *processed/*, and *processing/* folders. The */server-id/*

from-*partner*/processed/ folder contains a copy of the transferred text file, and the corresponding JSON and MDN files.

- `test.txt` – This object is the (empty) file that was transferred.

AS2 configurations

This topic describes the supported configurations, features, and capabilities for transfers that use the Applicability Statement 2 (AS2) protocol, including the accepted ciphers and digests.

Signing, encryption, compression, MDN

For both inbound and outbound transfers, the following items are either required or optional:

- **Encryption** – Required (for HTTP transport, which is the only transport method currently supported). Unencrypted messages are only accepted if forwarded by a TLS-terminating proxy such as an Application Load Balancer (ALB) and the `X-Forwarded-Proto: https` header is present.
- **Signing** – Optional
- **Compression** – Optional (the only currently supported compression algorithm is ZLIB)
- **Message Disposition Notice (MDN)** – Optional

Ciphers

The following ciphers are supported for both inbound and outbound transfers:

- AES128_CBC
- AES192_CBC
- AES256_CBC
- 3DES (for backward compatibility only)

Digests

The following digests are supported:

- **Inbound signing and MDN** – SHA1, SHA256, SHA384, SHA512
- **Outbound signing and MDN** – SHA1, SHA256, SHA384, SHA512

MDN

For MDN responses, certain types are supported, as follows:

- **Inbound transfers** – Synchronous and asynchronous
- **Outbound transfers** – Synchronous only
- **Simple Mail Transfer Protocol (SMTP) (email MDN)** – Not supported

Transports

- **Inbound transfers** – HTTP is the only currently supported transport, and you must specify it explicitly.

Note

If you need to use HTTPS for inbound transfers, you can terminate TLS on an Application Load Balancer or a Network Load Balancer. This is described in [Receive AS2 messages over HTTPS](#).

- **Outbound transfers** – If you provide an HTTP URL, you must also specify an encryption algorithm. If you provide an HTTPS URL, you have the option of specifying **NONE** for your encryption algorithm.

AS2 quotas and limitations

This section discusses quotas and limitations for AS2

Topics

- [AS2 quotas](#)
- [Quotas for handling secrets](#)
- [Known limitations](#)

AS2 quotas

The following quotas are in place for AS2 file transfers. To request an increase for a quota that's adjustable, see [AWS service quotas](#) in the *AWS General Reference*.

AS2 quotas

Name	Default	Adjustable
Maximum number of inbound files received per second	100	No
Maximum number of outbound files sent per second	100	No
Maximum number of concurrent inbound files	400	No
Maximum number of concurrent outbound files	400	No
Maximum size of inbound file (uncompressed)	1 GB	No
Maximum size of outbound file (uncompressed)	1 GB	No
Maximum number of files per outbound request	10	No
Maximum number of outbound requests per second	100	No
Maximum number of inbound requests per second	100	No
Maximum outbound bandwidth per account (outbound SFTP and AS2 requests both contribute to this value)	50 MB per second	No

Name	Default	Adjustable
Maximum number of agreements per server	100	Yes
Maximum number of connectors per account (SFTP and AS2 connectors both contribute to this limit)	100	Yes
Maximum number of certificates per partner profile	10	No
Maximum number of certificates per account	1000	Yes
Maximum number of partner profiles per account	1000	Yes

Quotas for handling secrets

AWS Transfer Family makes calls to AWS Secrets Manager on behalf of AS2 customers that are using Basic authentication. Additionally Secrets Manager makes calls to AWS KMS.

Note

These quotas aren't specific to your use of secrets for Transfer Family: they're shared among all the services in your AWS account.

For Secrets Manager `GetSecretValue`, the quota that applies is **Combined rate of DescribeSecret and GetSecretValue API requests**, as described in [AWS Secrets Manager quotas](#).

Secrets Manager GetSecretValue

Name	Value	Description
Combined rate of DescribeSecret and GetSecretValue API requests	Each supported Region: 10,000 per second	The maximum transactions per second for DescribeSecret and GetSecretValue API requests combined.

For AWS KMS, the following quotas apply for Decrypt. For details, see [Request quotas for each AWS KMS API operation](#)

AWS KMS Decrypt

Quota name	Default value (requests per second)
Cryptographic operations (symmetric) request rate	<p>These shared quotas vary with the AWS Region and the type of AWS KMS key used in the request. Each quota is calculated separately.</p> <ul style="list-style-type: none"> • 5,500 (shared) • 10,000 (shared) in the following Regions: <ul style="list-style-type: none"> • US East (Ohio), us-east-2 • Asia Pacific (Singapore), ap-southeast-1 • Asia Pacific (Sydney), ap-southeast-2 • Asia Pacific (Tokyo), ap-northeast-1 • Europe (Frankfurt), eu-central-1 • Europe (London), eu-west-2 • 50,000 (shared) in the following Regions: <ul style="list-style-type: none"> • US East (N. Virginia), us-east-1 • US West (Oregon), us-west-2 • Europe (Ireland), eu-west-1
Custom key store request quotas	Custom key store request quotas are calculated separately for each custom key store.

Quota name	Default value (requests per second)
<p>Note</p> <p>This quota only applies if you are using an external key store.</p>	<ul style="list-style-type: none"> • 1,800 (shared) for each AWS CloudHSM key store • 1,800 (shared) for each external key store

Known limitations

- Server-side TCP keep-alive is not supported. The connection times out after 350 seconds of inactivity unless the client sends keep-alive packets.
- For an active agreement to be accepted by the service and appear in Amazon CloudWatch logs, messages must contain valid AS2 headers.
- The server that's receiving messages from AWS Transfer Family for AS2 must support the Cryptographic Message Syntax (CMS) algorithm protection attribute for validating message signatures, as defined in [RFC 6211](#). This attribute is not supported in some older IBM Sterling products.
- Duplicate message IDs result in a processed/Warning: duplicate-document message.
- The key length for AS2 certificates must be at least 2048 bits, and at most 4096.
- When sending AS2 messages or asynchronous MDNs to a trading partner's HTTPS endpoint, the messages or MDNs must use a valid SSL certificate that's signed by a publicly trusted certificate authority (CA). Self-signed certificates are currently supported for outbound transfers only.
- The endpoint must support the TLS version 1.2 protocol and a cryptographic algorithm that's permitted by the security policy (as described in [Security policies for AWS Transfer Family servers](#)).
- Multiple attachments and certificate exchange messaging (CEM) from AS2 version 1.2 is not currently supported.
- Basic authentication is currently supported for outbound messages only.
- You can attach a file-processing workflow to a Transfer Family server that uses the AS2 protocol: however, AS2 messages don't execute workflows attached to the server.

AS2 features and capabilities

The following tables list the features and capabilities available for Transfer Family resources that use AS2.

AS2 features

Transfer Family offers the following features for AS2.

Feature	Supported by AWS Transfer Family
Drummond certification	Yes
AWS CloudFormation support	Yes
Amazon CloudWatch metrics	Yes
SHA-2 cryptographic algorithms	Yes
Support for Amazon S3	Yes
Support for Amazon EFS	No
Scheduled Messages	Yes ¹
AWS Transfer Family Managed Workflows	No
Certificate Exchange Messaging (CEM)	No
Mutual TLS (mTLS)	No
Support for self-signed certificates	Yes

1. Outbound Scheduled Messages available by [scheduling AWS Lambda functions using Amazon EventBridge](#)

AS2 send and receive capabilities

The following table provides a list of AWS Transfer Family AS2 send and receive capabilities.

Capability	Inbound: Receiving with server	Outbound: Sending with connector
TLS Encrypted Transport (HTTPS)	Yes ¹	Yes
Non-TLS Transport (HTTP)	Yes	Yes ²
Synchronous MDN	Yes	Yes
Message Compression	Yes	Yes
Asynchronous MDN	Yes	No
Static IP Address	Yes	Yes
Bring Your Own IP Address	Yes	No
Multiple File Attachments	No	No
Basic Authentication	No	Yes
AS2 Restart	Not applicable	No
AS2 Reliability	No	No
Custom Subject per Message	Not applicable	No

1. Inbound TLS Encrypted Transport available with Network Load Balancer (NLB)

2. Outbound non-TLS Transport available only when encryption is enabled

Configure AS2 connectors

The purpose of a connector is to establish a relationship between trading partners for *outbound* transfers—sending AS2 files from a Transfer Family server to an external, partner-owned destination. For the connector, you specify the local party, the remote partner, and their certificates (by creating local and partner profiles).

After you have a connector in place, you can transfer information to your trading partners. Each AS2 server is assigned three static IP addresses. AS2 connectors use these IP addresses for sending asynchronous MDNs to your trading partners over AS2.

Note

The message size received by a trading partner will not match the object size in Amazon S3. This discrepancy occurs because the AS2 message wraps the file in an envelope prior to sending. So, the file size might increase, even if the file is sent with compression. Therefore, make sure that the trading partner's maximum file size is greater than the size of the file that you are sending.

Create an AS2 connector

This procedure explains how to create AS2 connectors by using the AWS Transfer Family console. If you want to use the AWS CLI instead, see [the section called “Step 6: Create a connector between you and your partner”](#).

To create an AS2 connector

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Connectors**, and then choose **Create connector**.
3. In the **Connector configuration** section, specify the following information:
 - **URL** – Enter the URL for outbound connections.
 - **Access role** – Choose the Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role to use. Make sure that this role provides read and write access to the parent directory of the file location that's used in the `StartFileTransfer` request. Additionally, make sure that the role provides read and write access to the parent directory of the files that you intend to send with `StartFileTransfer`.

Note

If you're using Basic authentication for your connector, the access role requires the `secretsmanager:GetSecretValue` permission for the secret. If the secret is encrypted by using a customer managed key instead of the AWS managed key in AWS Secrets Manager, then the role also needs the `kms:Decrypt` permission for

that key. If you name your secret with the prefix `aws/transfer/`, you can add the necessary permission with a wildcard character (*), as shown in [Example permission to create secrets](#).

- **Logging role** (optional) – Choose the IAM role for the connector to use to push events to your CloudWatch logs.
4. In the **AS2 configuration** section, choose the local and partner profiles, the encryption and signing algorithms, and whether to compress the transferred information. Note the following:
 - For the encryption algorithm, do not choose `DES_EDE3_CBC` unless you must support a legacy client that requires it, as it is a weak encryption algorithm.
 - The **Subject** is used as the subject HTTP header attribute in AS2 messages that are being sent with the connector.
 - If you choose to create a connector without an encryption algorithm, you must specify HTTPS as your protocol.
 5. In the **MDN configuration** section, specify the following information:
 - **Request MDN** – You have the option to require your trading partner to send you an MDN after they have successfully received your message over AS2.
 - **Signed MDN** – You have the option to require that MDNs be signed. This option is available only if you have selected **Request MDN**.
 6. In the **Basic authentication** section, specify the following information.
 - To send sign-on credentials along with outbound messages, select **Enable Basic authentication**. If you don't want to send any credentials with outbound messages, keep **Enable Basic authentication** cleared.
 - If you're using authentication, choose or create a secret.
 - To create a new secret, choose **Create a new secret** and then enter a username and password. These credentials must match the user that connects to the partner's endpoint.

Basic authentication [Info](#)

Enable Basic authentication - optional
Select this option to authenticate with your trading partner's host using username and password credentials.

Basic authentication credentials [Info](#)
Choose the username and password credentials that will be used to authenticate with your trading partner's host.

Create a new secret

Choose an existing secret

Username

Password

ⓘ Update the access role associated with your connector to provide AWS Transfer Family with permission to read the secret containing your Basic authentication credentials.

- To use an existing secret, choose **Choose an existing secret**, and then choose a secret from the dropdown menu. For the details of creating a correctly formatted secret in Secrets Manager, see [Enable Basic authentication for AS2 connectors](#).

Basic authentication Info

Enable Basic authentication - optional
Select this option to authenticate with your trading partner's host using username and password credentials.

Basic authentication credentials Info
Choose the username and password credentials that will be used to authenticate with your trading partner's host.

Create a new secret
 Choose an existing secret

Choose a secret ▲

- transfer/as2-test
- aws/transfer/c-9...
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-
- aws/transfer/c-

7. After you've confirmed all of your settings, choose **Create connector** to create the connector.

The **Connectors** page appears, with the ID of your new connector added to the list. To view the details for your connectors, see [View AS2 connector details](#).

AS2 connector algorithms

When you create an AS2 connector, the following security algorithms are attached to the connector.

Type	Algorithm
TLS Cipher	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

Type	Algorithm
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

Basic authentication for AS2 connectors

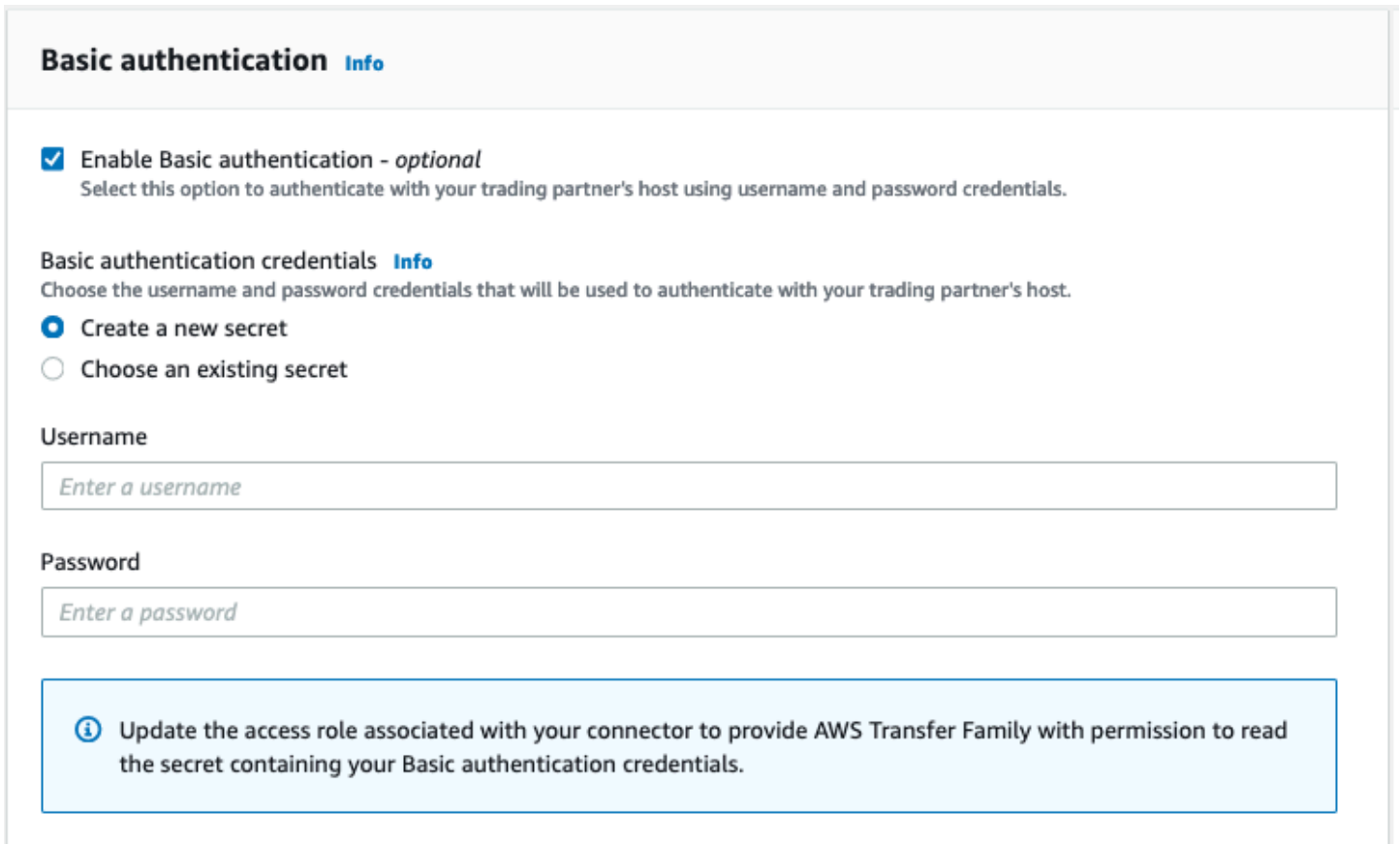
When you create or update a Transfer Family server that uses the AS2 protocol, you can add Basic authentication for outbound messages. You do this by adding authentication information to a connector.

Note

Basic authentication is available only if you're using HTTPS.

To use authentication for your connector, select **Enable Basic authentication** in the **Basic authentication** section. After you enable Basic authentication, you can choose to create a new secret, or use an existing one. In either case, the credentials in the secret are sent with outbound messages that use this connector. The credentials must match the user that is attempting to connect to the trading partner's remote endpoint.

The following screenshot shows **Enable Basic authentication** selected, and **Create a new secret** chosen. After making these choices, you can enter a username and password for the secret.



Basic authentication [Info](#)

Enable Basic authentication - optional
Select this option to authenticate with your trading partner's host using username and password credentials.

Basic authentication credentials [Info](#)
Choose the username and password credentials that will be used to authenticate with your trading partner's host.

Create a new secret

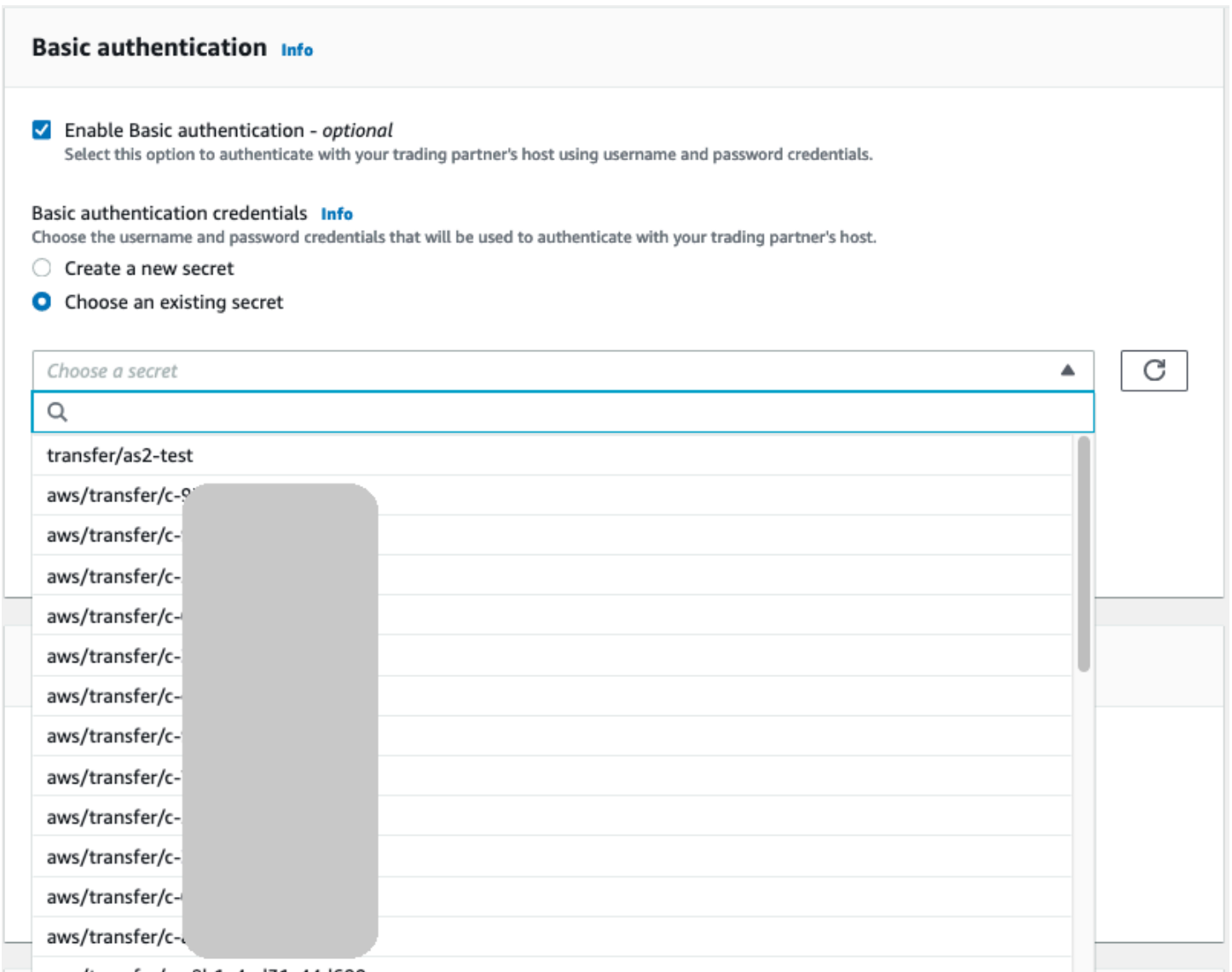
Choose an existing secret

Username

Password

i Update the access role associated with your connector to provide AWS Transfer Family with permission to read the secret containing your Basic authentication credentials.

The following screenshot shows **Enable Basic authentication** selected, and **Choose an existing secret** chosen. Your secret must be in the correct format, as described in [Enable Basic authentication for AS2 connectors](#).



Enable Basic authentication for AS2 connectors

When you enable Basic authentication for AS2 connectors, you can either create a new secret in the Transfer Family console, or you can use a secret that you create in AWS Secrets Manager. In either case, your secret is stored in Secrets Manager.

Topics

- [Create a new secret in the console](#)
- [Use an existing secret](#)
- [Create a secret in AWS Secrets Manager](#)

Create a new secret in the console

When you're creating a connector in the console, you can create a new secret.

To create a new secret, choose **Create a new secret** and then enter a username and password. These credentials must match the user that connects to the partner's endpoint.

Basic authentication [Info](#)

Enable Basic authentication - optional
Select this option to authenticate with your trading partner's host using username and password credentials.

Basic authentication credentials [Info](#)
Choose the username and password credentials that will be used to authenticate with your trading partner's host.

Create a new secret

Choose an existing secret

Username

Password

i Update the access role associated with your connector to provide AWS Transfer Family with permission to read the secret containing your Basic authentication credentials.

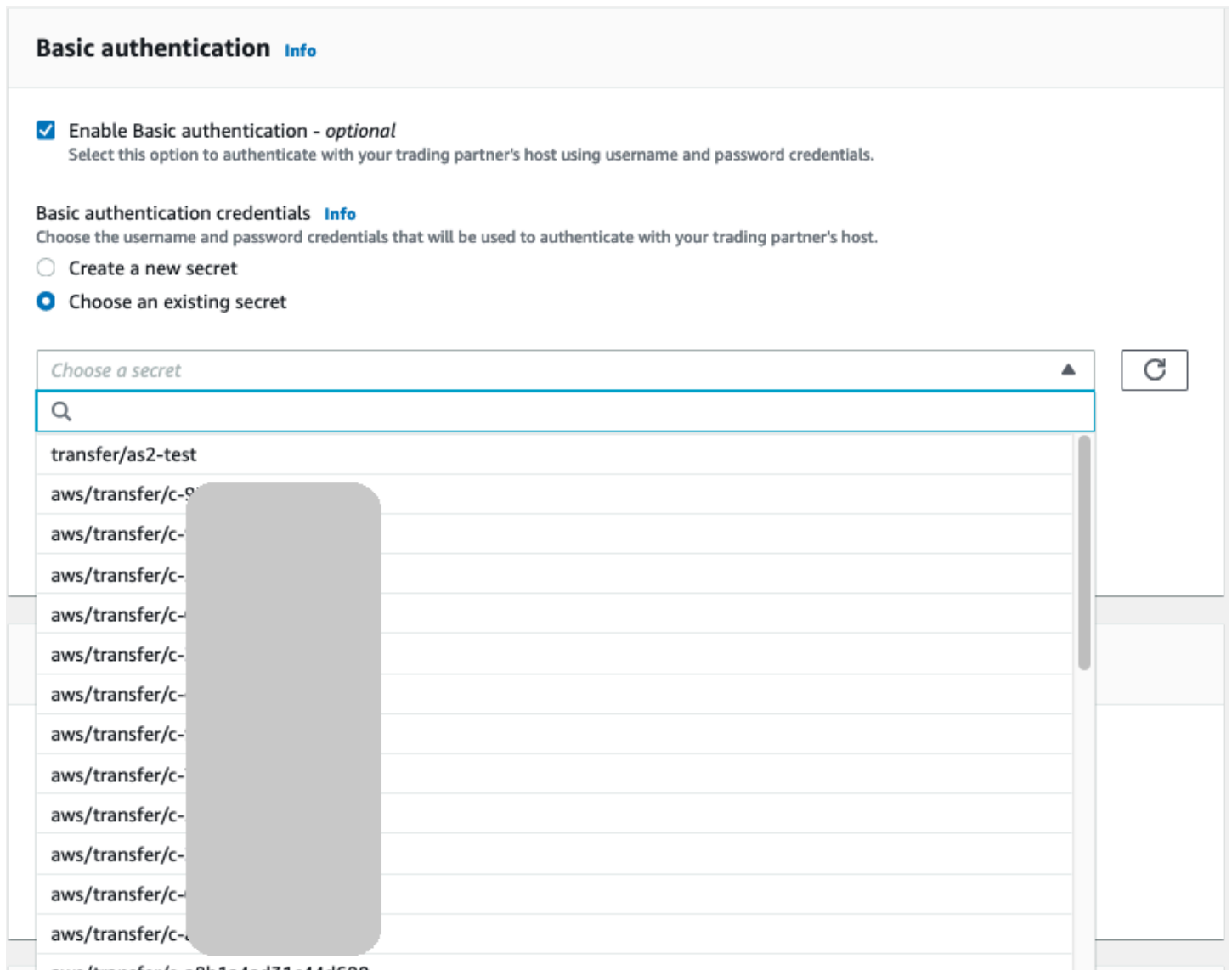
i Note

When you create a new secret in the console, the name of the secret follows this naming convention: `/aws/transfer/connector-id`, where *connector-id* is the ID of the connector that you're creating. Consider this when you are trying to locate the secret in AWS Secrets Manager.

Use an existing secret

When you're creating a connector in the console, you can specify an existing secret.

To use an existing secret, choose **Choose an existing secret**, and then choose a secret from the dropdown menu. For the details of creating a correctly formatted secret in Secrets Manager, see [Create a secret in AWS Secrets Manager](#).



Create a secret in AWS Secrets Manager

The following procedure describes how to create an appropriate secret for use with your AS2 connector.

Note
Basic authentication is available only if you're using HTTPS.

To store user credentials in Secrets Manager for AS2 Basic authentication

1. Sign in to the AWS Management Console and open the AWS Secrets Manager console at <https://console.aws.amazon.com/secretsmanager/>.
2. In the left navigation pane, choose **Secrets**.
3. On the **Secrets** page, choose **Store a new secret**.
4. On the **Choose secret type** page, for **Secret type**, choose **Other type of secret**.
5. In the **Key/value pairs** section, choose the **Key/value** tab.

- **Key** – Enter **Username**.
- **value** – Enter the name of the user that is authorized to connect to the partner' server.

6. If you want to provide a password, choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.

Choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.

- **Key** – Enter **Password**.
- **value** – Enter the password for the user.

7. If you want to provide a private key, choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.

- **Key** – Enter **PrivateKey**.
- **value** – Enter a private key for the user. This value must be stored in OpenSSH format, and must correspond to the public key that is stored for this user in the remote server.

8. Choose **Next**.
9. On the **Configure secret** page, enter a name and description for your secret. We recommend that you use a prefix of **aws/transfer/** for the name. For example, you could name your secret **aws/transfer/connector-1**.
10. Choose **Next**, and then accept the defaults on the **Configure rotation** page. Then choose **Next**.
11. On the **Review** page, choose **Store** to create and store the secret.

After you create the secret, you can choose it when you are creating a connector (see [Configure AS2 connectors](#)). In the step where you enable Basic authentication, choose the secret from the dropdown list of available secrets.

View AS2 connector details

You can find a list of details and properties for an AS2 AWS Transfer Family connector in the AWS Transfer Family console. An AS2 connector's properties include its URL, roles, profiles, MDNs, tags, and monitoring metrics.

This is the procedure for viewing connector details.

To view connector details

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Connectors**.
3. Choose the identifier in the **Connector ID** column to see the details page for the selected connector.

You can change the properties for the AS2 connector on the connector's details page by choosing **Edit**.

The screenshot displays the AWS Transfer Family console interface for an AS2 connector. The breadcrumb navigation shows 'Transfer Family > Connectors > [connector ID]'. The connector name is partially visible as 'C-'. A 'Delete' button is located in the top right corner. The main content is organized into four sections, each with an 'Edit' button:

- Connector configuration**: Includes fields for URL (http://...), Access role (...), and Logging role (...).
- Communication settings**: Includes AS2-From header (partner-test) and AS2-To header (local-test).
- AS2 configuration**: Includes Local profile (partner-test), Partner profile (local-test), Compression (Disabled), Message Subject (View), Encryption algorithm (AES256_CBC), and Signing algorithm (SHA256).
- MDN configuration**: Includes Request MDN (Enabled), Signed MDN (Default to message signing algorithm: SHA256), and Synchronization (Enabled).

Basic authentication Info
Edit

Basic authentication Secret

✔ Enabled aws/transfer-

Tags (3) Manage tags

Key	Value
aws:cloudformation:stack-name	
aws:cloudformation:logical-id	TransferConnector
aws:cloudformation:stack-id	arn:

AS2 Monitoring

OutboundMessages

2

● OutboundMessage

OutboundMessage

OutboundFailedMessage

--

● OutboundFailedMessage

OutboundFailedMessage

No data available. Try adjusting the dashboard time range.

i Note

You can get much of this information, albeit in a different format, by running the following AWS Command Line Interface (AWS CLI) command:

```
aws transfer describe-connector --connector-id your-connector-id
```

For more information, see [DescribeConnector](#) in the API reference.

Manage AS2 partners

This topic discusses how to manage AS2 certificates, profiles, and agreements.

Import AS2 certificates

The Transfer Family AS2 process uses certificate keys for both encryption and signing of transferred information. Partners can use the same key for both purposes, or a separate key for each. If you have common encryption keys kept in escrow by a trusted third-party so that data can be decrypted in the event of a disaster or security breach, we recommend having separate signing keys. By using separate signing keys (which you do not escrow), you don't compromise the non-repudiation features of your digital signatures.

Note

The key length for AS2 certificates must be at least 2048 bits, and at most 4096.

The following points detail how AS2 certificates are used during the process.

- Inbound AS2
 - The trading partner sends their public key for the signing certificate, and this key is imported to the partner profile.
 - The local party sends the public key for their encryption and signing certificates. The partner then imports the private key or keys. The local party can send separate certificate keys for signing and encryption, or can choose to use the same key for both purposes.
- Outbound AS2
 - The partner sends the public key for their encryption certificate, and this key is imported to the partner profile.
 - The local party sends the public key for the certificate for signing, and imports the private key of the certificate for signing.
 - If you are using HTTPS, you can import a self-signed Transport Layer Security (TLS) certificate.

For details on how to create certificates, see [the section called "Step 1: Create certificates for AS2"](#).

This procedure explains how to import certificates by using the Transfer Family console. If you want to use the AWS CLI instead, see [the section called "Step 3: Import certificates as Transfer Family certificate resources"](#).

To specify an AS2-enabled certificate

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, under **AS2 Trading Partners**, choose **Certificates**.
3. Choose **Import certificate**.
4. In the **Certificate description** section, enter an easily identifiable name for the certificate. Make sure that you can identify the certificate's purpose by its description. Additionally, choose the role for the certificate.
5. In the **Certificate contents** section, provide a public certificate from a trading partner, or the public and private keys for a local certificate.

6. In the **Certificate usage** section, choose the purpose for this certificate. It can be used for encryption, signing, or both.

Note

If you choose **Encryption and signing** for the usage, Transfer Family creates two identical certificates (each having their own ID): one with a usage value of ENCRYPTION and one with a usage value of SIGNING.

7. Fill in the **Certificate contents** section with the appropriate details.
 - If you choose **Self-signed certificate**, you do not provide the certificate chain.
 - Paste in the contents of the certificate.
 - If the certificate is not a self-signed certificate, provide the certificate chain.
 - If this certificate is a local certificate, paste in its private key.
8. Choose **Import certificate** to complete the process and save the details for the imported certificate.

Note

TLS certificates can only be imported as a partner's public certificate. If you select **Public certificate from a partner**, and then select **Transport Layer Security (TLS)** for the usage, you receive a warning. Also, TLS certificates must be self-signed (that is, you must select **Self Signed Certificate** to import a TLS certificate).

AS2 certificate rotation

Often, certificates are valid for a period of six months to a year. You might have set up profiles that you want to persist for a longer duration. To facilitate this, Transfer Family provides certificate rotation. You can specify multiple certificates for a profile, allowing you to keep using the profile for multiple years. Transfer Family uses certificates for signing (optional) and encryption (mandatory). You can specify a single certificate for both purposes, if you like.

Certificate rotation is the process of replacing an old expiring certificate with a newer certificate. The transition is a gradual one to avoid disrupting transfers where a partner in the agreement has yet to configure a new certificate for outbound transfers or might be sending payloads that are

signed or encrypted with an old certificate during a period when a newer certificate might also be in use. The intermediate period where both old and new certificates are valid is referred to as a *grace period*.

X.509 certificates have `Not Before` and `Not After` dates. However, these parameters might not provide enough control for administrators. Transfer Family provides `Active Date` and `Inactive Date` settings to control which certificate is used for outbound payloads and which is accepted for inbound payloads.

Outbound certificate selection uses the maximum value that is prior to the date of the transfer as an `Inactive Date`. Inbound processes accept certificates within the range of `Not Before` and `Not After` and within the range of `Active Date` and `Inactive Date`.

The following table describes one possible way to configure two certificates for a single profile.

Two certificates in rotation

Name	NOT BEFORE (controlled by certificate authority)	ACTIVE DATE (set by Transfer Family)	INACTIVE DATE (set by Transfer Family)	NOT AFTER (set by certificate authority)
Cert1 (older certificate)	2019-11-01	2020-01-01	2020-12-31	2024-01-01
Cert2 (newer certificate)	2020-11-01	2020-06-01	2021-06-01	2025-01-01

Note the following:

- When you specify an `Active Date` and `Inactive Date` for a certificate, the range must be inside the range between `Not Before` and `Not After`.
- We recommend that you configure several certificates for each profile, making sure that the active date range for all the certificates combined covers the amount of time for which you want to use the profile.
- We recommend that you specify some grace time between when your older certificate becomes inactive and when your newer certificate becomes active. In the preceding example, the first certificate does not become inactive until 2020-12-31, while the second certificate becomes

active on 2020-06-01, providing a 6-month grace period. During the period from 2020-06-01 until 2020-12-31, both certificates are active.

Create AS2 profiles

Use this procedure to create both local and partner profiles. This procedure explains how to create AS2 profiles by using the Transfer Family console. If you want to use the AWS CLI instead, see [the section called “Step 4: Create profiles for you and your trading partner”](#).

To create an AS2 profile

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, under **AS2 Trading Partners**, choose **Profiles**, then choose **Create profile**.
3. In the **Profile configuration** section, enter the AS2 ID for the profile. This value is used for the AS2 protocol-specific HTTP headers `as2-from` and `as2-to` to identify the trading partnership, which determines the certificates to use, and so on.
4. In the **Profile type** section, choose **Local profile** or **Partner profile**.
5. In the **Certificates** section, choose one or more certificates from the dropdown menu.

Note

If you want to import a certificate that is not listed in the dropdown menu, select **Import a new Certificate**. This opens a new browser window at the **Import certificate** screen. For the procedure about importing certificates see [Import AS2 certificates](#).

6. (Optional) In the **Tags** section, specify one or more key-value pairs to help identify this profile.
7. Choose **Create profile** to complete the process and save the new profile.

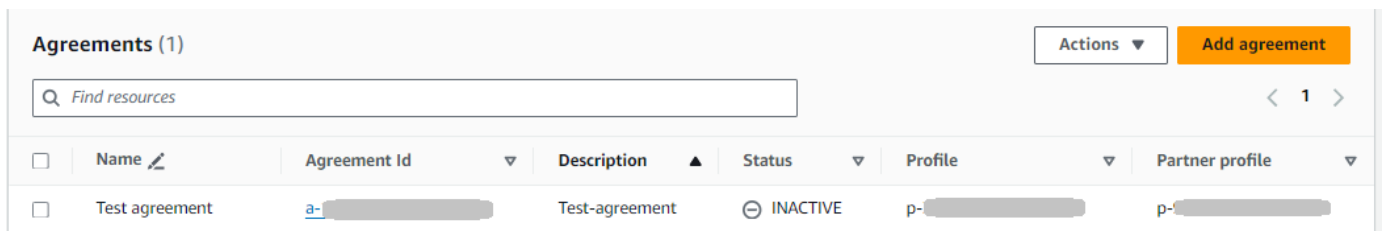
Create AS2 agreements

Agreements are associated with Transfer Family servers. They specify the details for trading partners that use the AS2 protocol to exchange messages or files by using Transfer Family, for *inbound* transfers—sending AS2 files from an external, partner-owned source to a Transfer Family server.

This procedure explains how to create AS2 agreements by using the Transfer Family console. If you want to use the AWS CLI instead, see [the section called “Step 5: Create an agreement between you and your partner”](#).

To create an agreement for a Transfer Family server

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Servers**, and then choose a server that uses the AS2 protocol.
3. On the server details page, scroll down to the **Agreements** section.



4. Choose **Add agreement**.
5. Fill in the agreement parameters, as follows:
 - a. In the **Agreement configuration** section, enter a descriptive name. Make sure that you can identify the agreement's purpose by its name. Also, set the **Status** for the agreement: either **Active** (selected by default) or **Inactive**.
 - b. In the **Communication configuration** section, choose a local profile and a partner profile.
 - c. In the **Inbox folder configuration** section, choose an Amazon S3 bucket to store incoming files and an IAM role that can access the bucket. Optionally, you can enter a prefix (folder) to use for storing files in the bucket.

For example, if you enter **DOC-EXAMPLE-BUCKET** for your bucket and **incoming** for your prefix, your incoming files are saved to the **/DOC-EXAMPLE-BUCKET/incoming** folder.
 - d. (Optional) Add tags in the **Tags** section.
 - e. After you have entered all the information for the agreement, choose **Create agreement**.

The new agreement appears in the **Agreements** section of the server details page.

Sending and receiving AS2 messages

This section describes the processes for sending and receiving AS2 messages. It also provide details on filenames and locations associated with AS2 messages.

The following table lists the available encryption algorithms for AS2 messages, and when you can use them.

Encryption algorithm	HTTP	HTTPS	Notes
AES128_CBC	Yes	Yes	
AES192_CBC	Yes	Yes	
AES256_CBC	Yes	Yes	
DES_EDE3_CBC	Yes	Yes	Only use this algorithm if you must support a legacy client that requires it, as it is a weak encryption algorithm.
NONE	No	Yes	If you are sending messages to a Transfer Family server, you can only select NONE if you are using an Application Load Balancer (ALB).

Topics

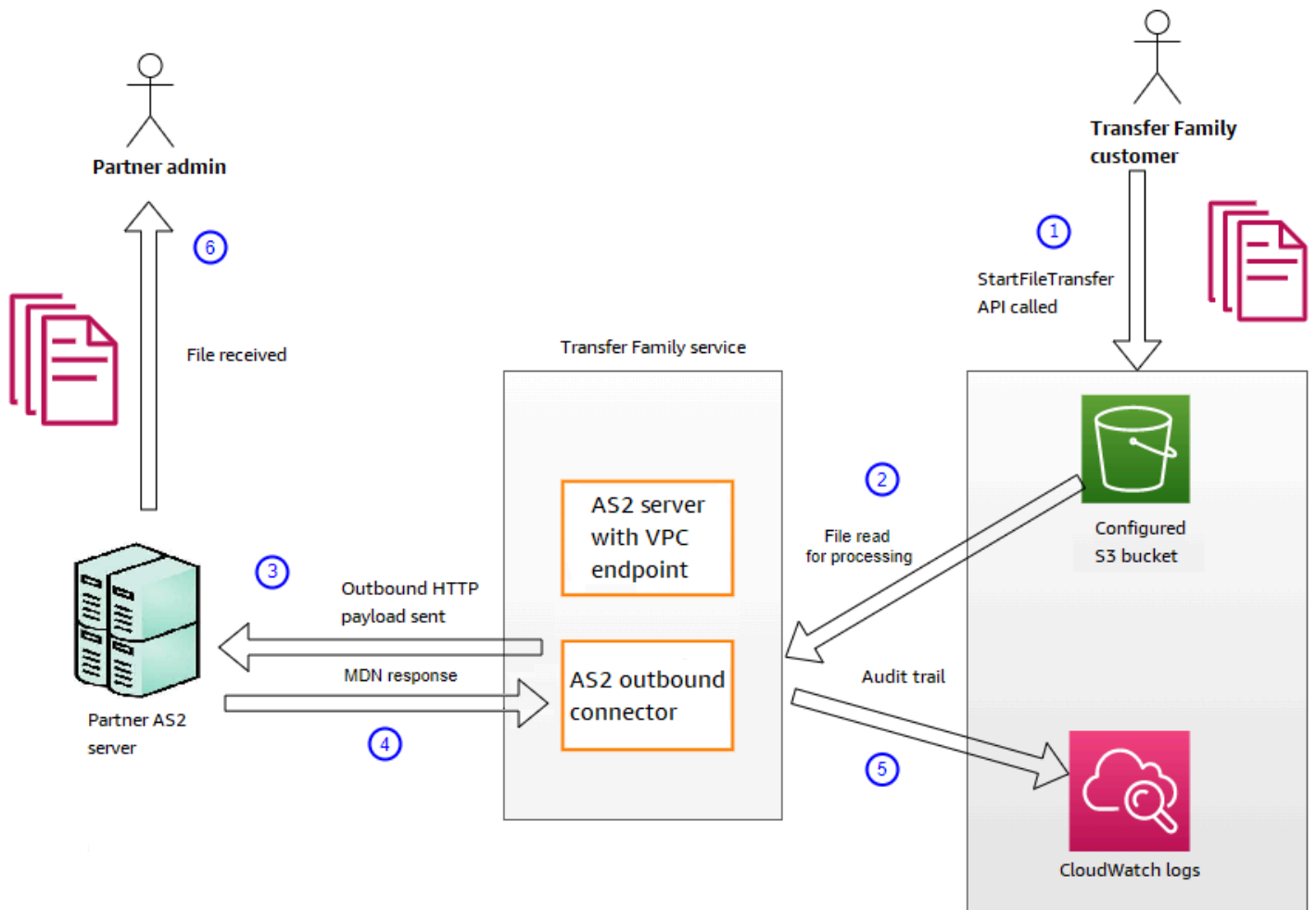
- [Send AS2 message process](#)
- [Receive AS2 message process](#)
- [Sending and receiving AS2 messages over HTTPS](#)
- [Transferring files by using an AS2 connector](#)

- [File names and locations](#)
- [Status codes](#)
- [Sample JSON files](#)

Send AS2 message process

The outbound process is defined as a message or file being sent from AWS to an external client or service. The sequence for outbound messages is as follows:

1. An admin calls the `start-file-transfer` AWS Command Line Interface (AWS CLI) command or the `StartFileTransfer` API operation. This operation references a connector configuration.
2. Transfer Family detects a new file request and locates the file. The file is compressed, signed, and encrypted.
3. A transfer HTTP client performs an HTTP POST request to transmit the payload to the partner's AS2 server.
4. The process returns the signed MDN response, inline with the HTTP response (synchronous MDN).
5. As the file moves between different stages of transmission, the process delivers the MDN response receipt and processing details to the customer.
6. The remote AS2 server makes the decrypted and verified file available to the partner admin.



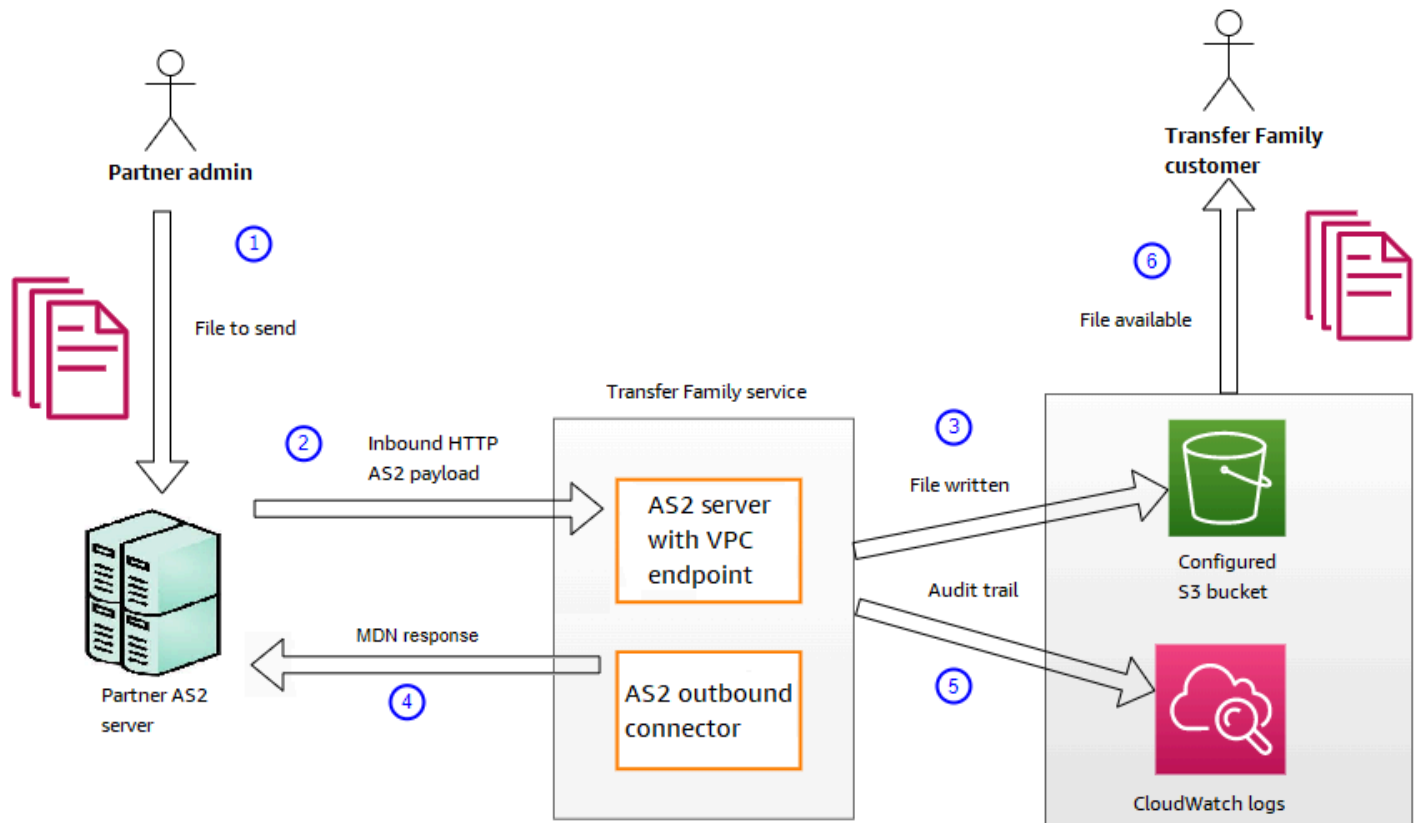
AS2 processing supports many of the RFC 4130 protocols, with a focus on common use cases and integration with existing AS2-enabled server implementations. For details of the supported configurations, see [???](#).

Receive AS2 message process

The inbound process is defined as a message or file that's being transferred to your AWS Transfer Family server. The sequence for inbound messages is as follows:

1. An admin or automated process starts an AS2 file transfer on the partner's remote AS2 server.
2. The partner's remote AS2 server signs and encrypts the file contents, then sends an HTTP POST request to an AS2 inbound endpoint hosted on Transfer Family.
3. Using the configured values for the server, partners, certificates, and agreement, Transfer Family decrypts and verifies the AS2 payload. The file contents are stored in the configured Amazon S3 file store.

4. The signed MDN response is returned either inline with the HTTP response, or asynchronously through a separate HTTP POST request back to the originating server.
5. An audit trail is written to Amazon CloudWatch with details about the exchange.
6. The decrypted file is available in a folder named `inbox/processed`.



Sending and receiving AS2 messages over HTTPS

This section describes how to configure a Transfer Family server that uses the AS2 protocol to send and receive messages over HTTPS.

Topics

- [Send AS2 messages over HTTPS](#)
- [Receive AS2 messages over HTTPS](#)

Send AS2 messages over HTTPS

To send AS2 messages using HTTPS, create a connector with the following information:

- For the URL, specify an HTTPS URL
- For the encryption algorithm, select any of the available algorithms.

Note

To send messages to a Transfer Family server while not using encryption (that is, you select NONE for the encryption algorithm), you must use an Application Load Balancer (ALB).

- Provide the remaining values for the connector as described in [Configure AS2 connectors](#).

Receive AS2 messages over HTTPS

AWS Transfer Family AS2 servers currently only provide HTTP transport over port 5080. However, you can terminate TLS on a network or application load balancer in front of your Transfer Family server VPC endpoint by using a port and certificate of your choosing. With this approach, you can have incoming AS2 messages use HTTPS.

Prerequisites

- The VPC must be in the same AWS Region as your Transfer Family server.
- The subnets of your VPC must be within the Availability Zones that you want to use your server in.

Note

Each Transfer Family server can support up to three Availability Zones.

- Allocate up to three Elastic IP addresses in the same Region as your server. Or, you can choose to bring your own IP address range (BYOIP).

Note

The number of Elastic IP addresses must match the number of Availability Zones that you use with your server endpoints.

You can configure either a Network Load Balance (NLB) or an Application Load Balancer (ALB). The following table lists the pros and cons for each approach.

The table below provides the differences in capabilities when you use an NLB versus an ALB to terminate TLS.

Feature	Network Load Balancer (NLB)	Application Load Balancer (ALB)
Latency	Lower latency as it operates at the network layer.	Higher latency as it operates at the application layer.
Static IP support	Can attach Elastic IP addresses that can be static.	Cannot attach Elastic IP addresses: provides a domain whose underlying IP addresses can change.
Advanced routing	Doesn't support advanced routing.	Supports advanced routing. Can inject X-Forwarded-Proto header required for AS2 without encryption. This header is described in X-Forwarded-Proto on the developer.mozilla.org website.
TLS/SSL termination	Supports TLS/SSL termination	Supports TLS/SSL termination
Mutual TLS (mTLS)	Transfer Family doesn't currently support using an NLB for mTLS	Support for mTLS

Configure NLB

This procedure describes how to set up an internet-facing Network Load Balancer (NLB) in your VPC.

To create a Network Load Balancer and define the VPC endpoint of the server as the load balancer's target

1. Open the Amazon Elastic Compute Cloud console at <https://console.aws.amazon.com/ec2/>.
2. From the navigation pane, choose **Load Balancers**, and then choose **Create load balancer**.
3. Under **Network Load Balancer**, choose **Create**.
4. In the **Basic configuration** section, enter the following information:
 - For **Name**, enter a descriptive name for the load balancer.
 - For **Scheme**, choose **Internet-facing**.
 - For **IP address type**, choose **IPv4**.
5. In the **Network mapping** section, enter the following information:
 - For **VPC**, choose the virtual private cloud (VPC) that you created.
 - Under **Mappings**, choose the Availability Zones associated with the public subnets that are available in the same VPC that you use with your server endpoints.
 - For the **IPv4 address** of each subnet, choose one of the Elastic IP addresses that you allocated.
6. In the **Listeners and routing** section, enter the following information:
 - For **Protocol**, choose **TLS**.
 - For **Port**, enter **5080**.
 - For **Default action**, choose **Create target group**. For the details of creating a new target group, see [To create a target group](#).

After you create a target group, enter its name in the **Default action** field.

7. In the **Secure listener settings** section, choose your certificate in the **Default SSL/TLS certificate** area.
8. Choose **Create load balancer** to create your NLB.
9. (Optional, but recommended) Turn on access logs for the Network Load Balancer to maintain a full audit trail, as described in [Access logs for your Network Load Balancer](#).

We recommend this step because the TLS connection is terminated at the NLB. Therefore, the source IP address that's reflected in your Transfer Family AS2 CloudWatch log groups is the NLB's private IP address, instead of your trading partner's external IP address.

Configure ALB

This procedure describes how to set up an Application Load Balancer (NLB) in your VPC.

To create an Application Load Balancer and define the VPC endpoint of the server as the load balancer's target

1. Open the Amazon Elastic Compute Cloud console at <https://console.aws.amazon.com/ec2/>.
2. From the navigation pane, choose **Load Balancers**, and then choose **Create load balancer**.
3. Under **Application Load Balancer**, choose **Create**.
4. In the ALB console, create a new HTTP listener on port 443 (HTTPS).
5. (Optional). If you want to set up mutual authentication (mTLS), configure security settings and a trust store.
 - a. Attach your SSL/TLS certificate to the listener.
 - b. Under **Client certificate handling**, select **Mutual authentication (mTLS)**.
 - c. Choose **Verify with trust store**.
 - d. Under **Advanced mTLS settings**, choose or create a trust store by uploading your CA certificates.
6. Create a new target group and add the private IP addresses of your Transfer Family AS2 server endpoints as targets on port 5080. For the details of creating a new target group, see [To create a target group](#).
7. Configure health checks for the target group to use the TCP protocol on port 5080.
8. Create a new rule to forward HTTPS traffic from the listener to the target group.
9. Configure the listener to use your SSL/TLS certificate.

After you set up the load balancer, clients communicate with the load balancer over the custom port listener. Then, the load balancer communicates with the server over port 5080.

To create a target group

1. After you choose **Create target group** in the previous procedure, you are taken to the **Specify group details** page for a new target group.
2. In the **Basic configuration** section, enter the following information.
 - For **Choose a target type**, choose **IP addresses**.
 - For **Target group name**, enter a name for the target group.
 - For **Protocol**, your selection is dependent upon whether you are using an ALB or an NLB.
 - For a Network Load Balancer (NLB), choose **TCP**
 - For an Application Load Balancer (ALB), choose **HTTP**
 - For **Port**, enter **5080**.
 - For **IP address type**, choose **IPv4**.
 - For **VPC**, choose the VPC that you created for your Transfer Family AS2 server.
3. In the **Health checks** section, choose **TCP** for the **Health check protocol**.
4. Choose **Next**.
5. On the **Register targets** page, enter the following information:
 - For **Network**, confirm that the VPC that you created for your Transfer Family AS2 server is specified.
 - For **IPv4 address**, enter the private IPv4 address of your Transfer Family AS2 server's endpoints.

If you have more than one endpoint for your server, choose **Add IPv4 address** to add another row for entering another IPv4 address. Repeat this process until you've entered the private IP addresses for all of your server's endpoints.
 - Make sure that **Ports** is set to **5080**.
 - Choose **Include as pending below** to add your entries to the **Review targets** section.
6. In the **Review targets** section, review your IP targets.
7. Choose **Create target group**, then go back to the previous procedure for creating your NLB and enter the new target group where indicated.

Test access to the server from an Elastic IP address

Connect to the server over the custom port by using an Elastic IP address or the DNS name of the Network Load Balancer.

Important

Manage access to your server from client IP addresses by using the [network access control lists \(network ACLs\)](#) for the subnets configured on the load balancer. Network ACL permissions are set at the subnet level, so the rules apply to all resources that are using the subnet. You can't control access from client IP addresses by using security groups, because the load balancer's target type is set to **IP addresses** instead of **Instances**. Therefore, the load balancer doesn't preserve source IP addresses. If the [Network Load Balancer's health checks](#) fail, this means that the load balancer can't connect to the server endpoint. To troubleshoot this issue, check the following:

- Confirm that the server [endpoint's associated security group](#) allows inbound connections from the subnets that are configured on the load balancer. The load balancer must be able to connect to the server endpoint over port 5080.
- Confirm that the server's **State** is **Online**.

Transferring files by using an AS2 connector

AS2 connectors establish a relationship between trading partners for transfers of AS2 messages from a Transfer Family server to an external, partner-owned destination.

You can use Transfer Family to send AS2 messages by referencing the connector ID and the paths to the files, as illustrated in the following `start-file-transfer` AWS Command Line Interface (AWS CLI) command:

```
aws transfer start-file-transfer --connector-id c-1234567890abcdef0 \  
--send-file-paths "/DOC-EXAMPLE-SOURCE-BUCKET/myfile1.txt" "/DOC-EXAMPLE-SOURCE-BUCKET/  
myfile2.txt"
```

To get the details for your connectors, run the following command:

```
aws transfer list-connectors
```

The `list-connectors` command returns the connector IDs, URLs, and Amazon Resource Names (ARNs) for your connectors.

To return the properties of a particular connector, run the following command with the ID that you want to use:

```
aws transfer describe-connector --connector-id your-connector-id
```

The `describe-connector` command returns all of the properties for the connector, including its URL, roles, profiles, Message Disposition Notices (MDNs), tags, and monitoring metrics.

You can confirm that the partner successfully received the files by viewing the JSON and MDN files. These files are named according to the conventions described in [File names and locations](#). If you configured a logging role when you created the connector, you can also check your CloudWatch logs for the status of AS2 messages.

To view AS2 connector details, see [View AS2 connector details](#). For more information about creating AS2 connectors, see [Configure AS2 connectors](#).

File names and locations

This section discusses the file-naming conventions for AS2 transfers.

For inbound file transfers, note the following:

- You specify the base directory in an agreement. The base directory is the Amazon S3 bucket name combined with a prefix, if any. For example, `/DOC-EXAMPLE-BUCKET/AS2-folder`.
- If an incoming file is processed successfully, the file (and the corresponding JSON file) is saved to the `/processed` folder. For example, `/DOC-EXAMPLE-BUCKET/AS2-folder/processed`.

The JSON file contains the following fields:

- `agreement-id`
- `as2-from`
- `as2-to`
- `as2-message-id`
- `transfer-id`
- `client-ip`
- `connector-id`
- `failure-message`
- `file-path`

- message-subject
 - mdn-message-id
 - mdn-subject
 - requester-file-name
 - requester-content-type
 - server-id
 - status-code
 - failure-code
 - transfer-size
- If an incoming file cannot be processed successfully, the file (and the corresponding JSON file) is saved to the /failed folder. For example, /DOC-EXAMPLE-BUCKET/AS2-folder/failed.
 - The transferred file is stored in the processed folder as *original_filename.messageId.original_extension*. That is, the message ID for the transfer is appended to the name of the file, before its original extension.
 - A JSON file is created and saved as *original_filename.messageId.original_extension.json*. In addition to the message ID being added, the string .json is appended to the transferred file's name.
 - A Message Disposition Notice (MDN) file is created and saved as *original_filename.messageId.original_extension.mdn*. In addition to the message ID being added, the string .mdn is appended to the transferred file's name.
 - If there is an inbound file named ExampleFileInS3Payload.dat, the following files are created:
 - **File** –
ExampleFileInS3Payload.c4d6b6c7-23ea-4b8c-9ada-0cb811dc8b35@44313c54b0a46a36.
 - **JSON** –
ExampleFileInS3Payload.c4d6b6c7-23ea-4b8c-9ada-0cb811dc8b35@44313c54b0a46a36.
 - **MDN** –
ExampleFileInS3Payload.c4d6b6c7-23ea-4b8c-9ada-0cb811dc8b35@44313c54b0a46a36.

For outbound transfers, the naming is similar, with the difference that there is no incoming message file, and also, the transfer ID for the transferred message is added to the file name. The

transfer ID is returned by the `StartFileTransfer` API operation (or when another process or script calls this operation).

- The `transfer-id` is an identifier that is associated with a file transfer. All requests that are part of a `StartFileTransfer` call share a `transfer-id`.
- The base directory is the same as the path that you use for the source file. That is, the base directory is the path that you specify in the `StartFileTransfer` API operation or `start-file-transfer` AWS CLI command. For example:

```
aws transfer start-file-transfer --send-file-paths /DOC-EXAMPLE-BUCKET/AS2-folder/
file-to-send.txt
```

If you run this command, MDN and JSON files are saved in `/DOC-EXAMPLE-BUCKET/AS2-folder/processed` (for successful transfers), or `/DOC-EXAMPLE-BUCKET/AS2-folder/failed` (for unsuccessful transfers).

- A JSON file is created and saved as `original_filename.transferId.messageId.original_extension.json`.
- An MDN file is created and saved as `original_filename.transferId.messageId.original_extension.mdn`.
- If there is an outbound file named `ExampleFileOutTestOutboundSyncMdn.dat`, the following files are created:
 - **JSON** – `ExampleFileOutTestOutboundSyncMdn.dedf4601-4e90-4043-b16b-579af35e0d83.fbe18db8-7361-42ff-8ab6-49ec1e435f34@c9c705f0baaaabaa.dat.json`
 - **MDN** – `ExampleFileOutTestOutboundSyncMdn.dedf4601-4e90-4043-b16b-579af35e0d83.fbe18db8-7361-42ff-8ab6-49ec1e435f34@c9c705f0baaaabaa.dat.mdn`

You can also check the CloudWatch logs to view the details of your transfers, including any that failed.

Status codes

The following table lists all of the status codes that can be logged to CloudWatch logs when you or your partner send an AS2 message. Different message processing steps apply to different message types and are intended for monitoring only. The `COMPLETED` and `FAILED` states represent the final step in processing, and are visible in JSON files.

Code	Description	Processing completed?
PROCESSING	The message is in the process of being converted to its final format. For example, decompression and decryption steps both have this status.	No
MDN_TRANSMIT	Message processing is sending an MDN response.	No
MDN_RECEIVE	Message processing is receiving an MDN response.	No
COMPLETED	Message processing has completed successfully. This state includes when an MDN is sent for an inbound message or for MDN verification of outbound messages.	Yes
FAILED	The message processing has failed. For a list of error codes, see AS2 error codes .	Yes

Sample JSON files

This section lists sample JSON files for both inbound and outbound transfers, including sample files for successful transfers and transfers that fail.

Sample outbound file that is successfully transferred:

```
{
  "requester-content-type": "application/octet-stream",
  "message-subject": "File xyzTest from MyCompany_OID to partner YourCompany",
  "requester-file-name": "TestOutboundSyncMdn-9lmCr79hV.dat",
  "as2-from": "MyCompany_OID",
  "connector-id": "c-c21c63ceaaf34d99b",
```

```

"status-code": "COMPLETED",
"disposition": "automatic-action/MDN-sent-automatically; processed",
"transfer-size": 3198,
"mdn-message-id": "OPENAS2-11072022063009+0000-df865189-1450-435b-9b8d-
d8bc0cee97fd@PartnerA_OID_MyCompany_OID",
"mdn-subject": "Message be18db8-7361-42ff-8ab6-49ec1e435f34@c9c705f0baaaabaa has been
accepted",
"as2-to": "PartnerA_OID",
"transfer-id": "dedf4601-4e90-4043-b16b-579af35e0d83",
"file-path": "/DOC-EXAMPLE-BUCKET/as2testcell10000/openAs2/
TestOutboundSyncMdn-9lmCr79hV.dat",
"as2-message-id": "fbe18db8-7361-42ff-8ab6-49ec1e435f34@c9c705f0baaaabaa",
"timestamp": "2022-07-11T06:30:10.791274Z"
}

```

Sample outbound file that is unsuccessfully transferred:

```

{
  "failure-code": "HTTP_ERROR_RESPONSE_FROM_PARTNER",
  "status-code": "FAILED",
  "requester-content-type": "application/octet-stream",
  "subject": "Test run from Id da86e74d6e57464aae1a55b8596bad0a to partner
9f8474d7714e476e8a46ce8c93a48c6c",
  "transfer-size": 3198,
  "requester-file-name": "openAs2TestOutboundWrongAs2Ids-necco-3VYn5n8wE.dat",
  "as2-message-id": "9a9cc9ab-7893-4cb6-992a-5ed8b90775ff@718de4cec1374598",
  "failure-message": "http://Test123456789.us-east-1.elb.amazonaws.com:10080 returned
status 500 for message with ID 9a9cc9ab-7893-4cb6-992a-5ed8b90775ff@718de4cec1374598",
  "transfer-id": "07bd3e07-a652-4cc6-9412-73ffdb97ab92",
  "connector-id": "c-056e15cc851f4b2e9",
  "file-path": "/DOC-EXAMPLE-BUCKET-4c1tq6ohjt9y/as2IntegCell10002/openAs2/
openAs2TestOutboundWrongAs2Ids-necco-3VYn5n8wE.dat",
  "timestamp": "2022-07-11T21:17:24.802378Z"
}

```

Sample inbound file that is successfully transferred:

```

{
  "requester-content-type": "application/EDI-X12",
  "subject": "File openAs2TestInboundAsyncMdn-necco-5Ab6bTfC0.dat sent from MyCompany
to PartnerA",
  "client-ip": "10.0.109.105",
  "requester-file-name": "openAs2TestInboundAsyncMdn-necco-5Ab6bTfC0.dat",

```

```

"as2-from": "MyCompany_OID",
"status-code": "COMPLETED",
"disposition": "automatic-action/MDN-sent-automatically; processed",
"transfer-size": 1050,
"mdn-subject": "Message Disposition Notification",
"as2-message-id": "OPENAS2-11072022233606+0000-5dab0452-0ca1-4f9b-b622-
fba84effff3c@MyCompany_OID_PartnerA_OID",
"as2-to": "PartnerA_OID",
"agreement-id": "a-f5c5cbea5f7741988",
"file-path": "processed/openAs2TestInboundAsyncMdn-
necco-5Ab6bTfC0.OPENAS2-11072022233606+0000-5dab0452-0ca1-4f9b-b622-
fba84effff3c@MyCompany_OID_PartnerA_OID.dat",
"server-id": "s-5f7422b04c2447ef9",
"timestamp": "2022-07-11T23:36:36.105030Z"
}

```

Sample inbound file that is unsuccessfully transferred:

```

{
  "failure-code": "INVALID_REQUEST",
  "status-code": "FAILED",
  "subject": "Sending a request from InboundHttpClientTests",
  "client-ip": "10.0.117.27",
  "as2-message-id": "testFailedLogs-TestRunConfig-Default-inbound-direct-
integ-0c97ee55-af56-4988-b7b4-a3e0576f8f9c@necco",
  "as2-to": "0beff6af56c548f28b0e78841dce44f9",
  "failure-message": "Unsupported date format: 2022/123/456T",
  "agreement-id": "a-0ceec8ca0a3348d6a",
  "as2-from": "ab91a398aed0422d9dd1362710213880",
  "file-path": "failed/01187f15-523c-43ac-9fd6-51b5ad2b08f3.testFailedLogs-
TestRunConfig-Default-inbound-direct-integ-0c97ee55-af56-4988-b7b4-a3e0576f8f9c@necco",
  "server-id": "s-0582af12e44540b9b",
  "timestamp": "2022-07-11T06:30:03.662939Z"
}

```

Monitoring AS2 usage

You can monitor AS2 activity using Amazon CloudWatch and AWS CloudTrail. To view other Transfer Family server metrics, see [Amazon CloudWatch logging for AWS Transfer Family](#)

AS2 metrics

Metric	Description
InboundMessage	<p>The total number of AS2 messages successfully received from a trading partner.</p> <p>Units: Count</p> <p>Period: 5 minutes</p>
InboundFailedMessage	<p>The total number of AS2 messages that were unsuccessfully received from a trading partner. That is, a trading partner sent a message, but the Transfer Family server was not able to successfully process it.</p> <p>Units: Count</p> <p>Period: 5 minutes</p>
OutboundMessage	<p>The total number of AS2 messages successfully sent from the Transfer Family server to a trading partner.</p> <p>Units: Count</p> <p>Period: 5 minute</p>
OutboundFailedMessage	<p>The total number of AS2 messages that were unsuccessfully sent to a trading partner. That is, they were sent from the Transfer Family server, but were not successfully received by the trading partner.</p> <p>Units: Count</p> <p>Period: 5 minutes</p>

AS2 Status codes

The following table lists all of the status codes that can be logged to CloudWatch logs when you or your partner send an AS2 message. Different message processing steps apply to different message types and are intended for monitoring only. The COMPLETED and FAILED states represent the final step in processing, and are visible in JSON files.

Code	Description	Processing completed?
PROCESSING	The message is in the process of being converted to its final format. For example, decompression and decryption steps both have this status.	No
MDN_TRANSMIT	Message processing is sending an MDN response.	No
MDN_RECEIVE	Message processing is receiving an MDN response.	No
COMPLETED	Message processing has completed successfully. This state includes when an MDN is sent for an inbound message or for MDN verification of outbound messages.	Yes
FAILED	The message processing has failed. For a list of error codes, see AS2 error codes .	Yes

AS2 error codes

The following table lists and describes error codes that you might receive from AS2 file transfers.

AS2 error codes

Code	Error	Description and resolution
ACCESS_DENIED	<ul style="list-style-type: none"> Access denied. Check if your access role has necessary permissions. Invalid file path <i>send-file-path</i> Failed to get credentials with ErrorCode: <i>error-code</i> 	<p>Occurs when handling a <code>StartFileTransfer</code> request where any of the <code>SendFilePaths</code> are not valid or malformed. That is, the path is missing the Amazon S3 bucket name, or the path includes characters that aren't valid. Also occurs if Transfer Family fails to assume the access role or logging role.</p> <p>Ensure that the path contains a valid Amazon S3 bucket name and key name.</p>
AGREEMENT_NOT_FOUND	Agreement was not found.	<p>Either the agreement was not found, or the agreement is associated with an inactive profile.</p> <p>Update the agreement within the Transfer Family server to include active profiles.</p>
CONNECTOR_NOT_FOUND	Connector or related configuration was not found.	<p>Either the connector was not found, or the connector is associated with an inactive profile.</p> <p>Update the connector to include active profiles.</p>

Code	Error	Description and resolution
CREDENTIALS_RETRIEVAL_FAILED	<ol style="list-style-type: none"> 1. Secret not found in Secrets Manager. 2. Cannot access Secrets Manager. 3. Failed to decrypt secret in Secrets Manager. 4. Cannot get secret value due to throttling. 	<p>For AS2 Basic authentication, the secret must be formatted correctly. The following resolutions correspond to the errors listed in the previous column.</p> <ol style="list-style-type: none"> 1. Ensure that the secret ID is correct. 2. Ensure that the access role has the appropriate permissions to read the secret. The access role must provide read and write access to the parent directory of the file location used in the <code>StartFileTransfer</code> request. Additionally, make sure that the role provides read and write access to the parent directory of the files that you intend to send with <code>StartFileTransfer</code>. 3. If a customer managed key is being used for the secret, ensure that the access role has permissions for the AWS Key Management Service (AWS KMS) key. 4. For the applicable quotas, see Quotas for handling secrets.

Code	Error	Description and resolution
DECOMPRESSION_FAILED	Failed to decompress message.	<p>Either the file sent is corrupt, or the compression algorithm is not valid.</p> <p>Resend the message and verify that ZLIB compression is used, or resend the message without compression enabled.</p>
DECRYPT_FAILED	Failed to decrypt message <i>message-ID</i> . Ensure that the partner has the correct public encryption key.	<p>Decryption failed.</p> <p>Confirm that the partner sent a payload by using a valid certificate and that encryption was performed by using a valid encryption algorithm.</p>
DECRYPT_FAILED_INVALID_SMIME_FORMAT	Unable to parse enveloped mimePart.	<p>MIME payload is either corrupt or in an unsupported SMIME format.</p> <p>The sender should make sure that the format they're using is supported, and then resend the payload.</p>

Code	Error	Description and resolution
DECRYPT_FAILED_NO_DECRYPTION_KEY_FOUND	No matching decryption key found.	<p>The partner profile did not have a certificate assigned that matched the message, or the certificates that matched the message are now expired or no longer valid.</p> <p>You must update the partner profile and ensure that it contains a valid certificate.</p>
DECRYPT_FAILED_UNSUPPORTED_ENCRYPTION_ALG	SMIME Payload Decryption requested using unsupported algorithm with ID: <i>encryption-ID</i> .	<p>The remote sender has sent an AS2 payload with an unsupported encryption algorithm.</p> <p>The sender must choose an encryption algorithm that's supported by AWS Transfer Family.</p>
DUPLICATE_MESSAGE	Duplicate or double processed step.	<p>The payload has a duplicate processing step. For example, there are two encryption steps.</p> <p>Resend the message with a single step for signing, compression, and encryption.</p>

Code	Error	Description and resolution
ENCRYPT_FAILED_NO_ENCRYPTION_KEY_FOUND	No valid public encryption certificates found in profile: <i>local-profile-ID</i>	<p>Transfer Family is attempting to encrypt an outbound message, but no encryption certificates are found for the local profile.</p> <p>Resolution options:</p> <ul style="list-style-type: none"> • Ensure that the local profile has a certificate and private key for encryption attached. • Ensure that the encryption certificate is currently active.
ENCRYPTION_FAILED	Failed to encrypt file <i>file-name</i> .	<p>The file to be sent is not available for encryption.</p> <p>Verify that the file is in its expected AS2 location and that AWS Transfer Family has permission to read the file.</p>
FILE_SIZE_TOO_LARGE	File size is too large.	This occurs when sending or receiving a file that exceeds the file size limit.
HTTP_ERROR_RESPONSE_FROM_PARTNER	<i>partner-URL</i> returned status 400 for message with ID= <i>message-ID</i> .	<p>Communicating with the partner's AS2 server returned an unexpected HTTP response code.</p> <p>The partner might be able to provide more diagnostics from their AS2 server logs.</p>

Code	Error	Description and resolution
INSUFFICIENT_MESSAGE_SECURITY_UNENCRYPTED	Encryption is required.	The partner sent an unencrypted message to Transfer Family, which is not supported. The sender must use an encrypted payload.
INVALID_ENDPOINT_PROTOCOL	Only HTTP and HTTPS are supported.	You must specify HTTP or HTTPS as the protocol in your AS2 connector configuration.

Code	Error	Description and resolution
INVALID_REQUEST	<ol style="list-style-type: none"> 1. There is a problem with a message header. 2. Could not parse secret JSON. Secret JSON did not match expected format. 3. Secret must be a JSON string. 4. Username must not contain a colon. Username must not contain control characters. Username must contain only ASCII characters. Password must not contain control characters. Password must contain only ASCII characters. 	<p>This error has several causes. The following resolutions correspond to the errors listed in the previous column.</p> <ol style="list-style-type: none"> 1. Check the <code>as2-from</code> and <code>as2-to</code> fields. Make sure that the original message ID is accurate for the MDN format. Also make sure that the message ID format is not missing any AS2 headers. 2. Ensure that the secret value matches the documented format, as described in Enable Basic authentication for AS2 connectors. 3. Ensure that the secret is provided as a string, and not as a binary. 4. Make the necessary correction to the username or password.

Code	Error	Description and resolution
INVALID_URL_FORMAT	Invalid URL format: <i>URL</i>	<p>This occurs when you are sending an outbound message using a connector configured with a malformed URL.</p> <p>Ensure that the connector is configured with a valid HTTP or HTTPS URL.</p>
MDN_RESPONSE_INDICATES_AUTHENTICATION_FAILED	Not applicable	<p>The receiver cannot authenticate the sender. The trading partner returns an MDN to Transfer Family with the disposition modifier Error: authentication-failed.</p>
MDN_RESPONSE_INDICATES_DECOMPRESSION_FAILED	Not applicable	<p>This occurs when the receiver cannot decompress the message contents. The trading partner returns an MDN to Transfer Family with the disposition modifier Error: decompression-failed.</p>
MDN_RESPONSE_INDICATES_DECRYPTION_FAILED	Not applicable	<p>The receiver cannot decrypt the message contents. The trading partner returns an MDN to Transfer Family with the disposition modifier Error: authentication-failed.</p>

Code	Error	Description and resolution
MDN_RESPONSE_INDICATES_INSUFFICIENT_MESSAGE_SECURITY	Not applicable	<p>The receiver expects the message to be signed or encrypted, but it isn't. The trading partner returns an MDN to Transfer Family with the disposition modifier Error: insufficient-message-security.</p> <p>Enable signing and/or encryption on the connector to match the trading partner's expectations.</p>
MDN_RESPONSE_INDICATES_INTEGRITY_CHECK_FAILED	Not applicable	<p>The receiver cannot verify content integrity. The trading partner returns an MDN to Transfer Family with the disposition modifier Error: integrity-check-failed.</p>
PATH_NOT_FOUND	Unable to create directory <i>file-path</i> . The parent path could not be found.	<p>Transfer Family is attempting to create a directory in the customer's Amazon S3 bucket, but the bucket is not found.</p> <p>Ensure that each path mentioned in the StartFile Transfer command contains the name of an existing bucket.</p>

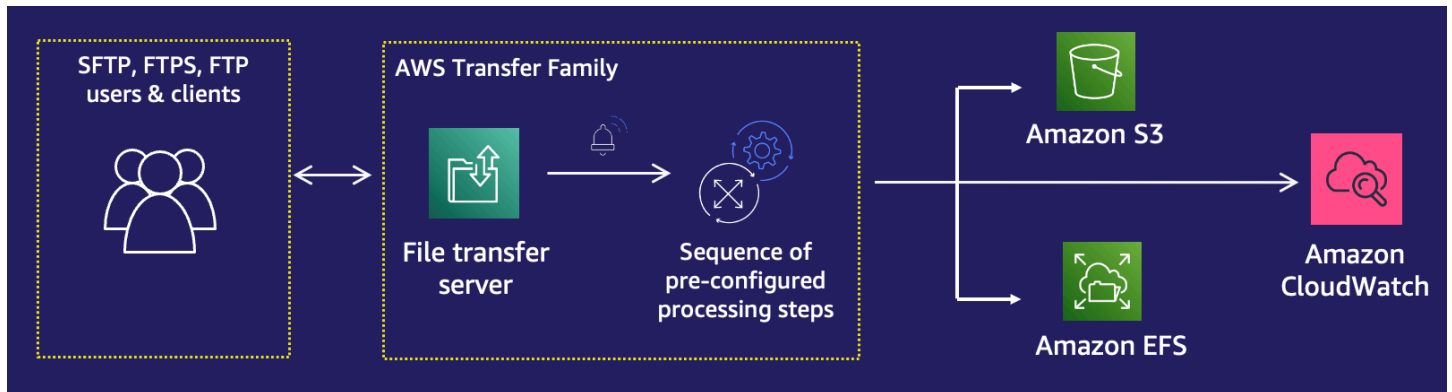
Code	Error	Description and resolution
SEND_FILE_NOT_FOUND	File path <i>file-path</i> not found.	<p>Transfer Family can't locate the file in the send file operation.</p> <p>Check that the configured home directory and path are valid and that Transfer Family has read permissions for the file.</p>
SERVER_NOT_FOUND	Server associated with the message cannot be found.	<p>Transfer Family could not find the server when receiving a message. This can happen if the server is deleted during the processing of an incoming message.</p>
SERVER_NOT_ONLINE	Server <i>server-ID</i> is not online.	<p>The Transfer Family server is offline.</p> <p>Start the server so that it can receive and process messages.</p>
SIGNING_FAILED	Failed to sign file.	<p>The file to be sent is not available for signing, or signing could not be performed.</p> <p>Verify that the file is in its expected AS2 location and that AWS Transfer Family has permission to read the file.</p>

Code	Error	Description and resolution
SIGNING_FAILED_NO_SIGNING_KEY_FOUND	No certificate found for profile: <i>local-profile-ID</i> .	<p>Attempting to sign an outbound message, but no signing certificates are found for the local profile.</p> <p>Resolution options:</p> <ul style="list-style-type: none"> • Ensure that the local profile has a certificate and private key for signing attached. • Ensure that the signing certificate is currently active.
UNABLE_RESOLVE_HOST_TO_IP_ADDRESS	Unable to resolve hostname to IP addresses.	<p>Transfer Family is unable to perform DNS to IP address resolution on the public DNS server that is configured in the AS2 connector.</p> <p>Update the connector to point to a valid partner URL.</p>
UNABLE_TO_CONNECT_TO_REMOTE_HOST_OR_IP	Connection to endpoint timed out.	<p>Transfer Family cannot establish a socket connection to the configured partner's AS2 server.</p> <p>Check that the partner's AS2 server is available at the configured IP address.</p>

Code	Error	Description and resolution
UNABLE_TO_RESOLVE_HOSTNAME	Unable to resolve hostname <i>hostname</i> .	<p>The Transfer Family server could not resolve the partner's hostname by using a public DNS server.</p> <p>Check that the configured host is registered and that the DNS record has had time to publish.</p>
VERIFICATION_FAILED	Signature verification failed for AS2 message <i>message-ID</i> or a MIC code did not match.	Check that the sender's signing certificate matches the signing certificates for the remote profile. Also check that the MIC algorithms are compatible with AWS Transfer Family.
VERIFICATION_FAILED_NO_MATCHING_KEY_FOUND	<ul style="list-style-type: none"> No public certificate matching message signature could be found in profile: <i>partner-profile-ID</i> . Cannot get certificates for non-existent profile: <i>partner-profile-ID</i> . No valid certificate was found in profile: <i>partner-profile-ID</i> . 	<p>AWS Transfer Family is attempting to verify the signature for a received message, but no matching signing certificate is found for the partner profile.</p> <p>Resolution options:</p> <ul style="list-style-type: none"> Ensure that the partner profile has a signing certificate attached. Ensure that the certificate is currently active. Ensure that the certificate is the correct signing certificate for the partner.

AWS Transfer Family managed workflows

AWS Transfer Family supports managed workflows for file processing. With managed workflows, you can kick off a workflow after a file has been transferred over SFTP, FTPS, or FTP. Using this feature, you can securely and cost effectively meet your compliance requirements for business-to-business (B2B) file exchanges by coordinating all the necessary steps required for file processing. In addition, you benefit from end-to-end auditing and visibility.



By orchestrating file-processing tasks, managed workflows help you preprocess data before it is consumed by your downstream applications. Such file-processing tasks might include:

- Moving files to user-specific folders.
- Decrypting files as part of a workflow.
- Tagging files.
- Performing custom processing by creating and attaching an AWS Lambda function to a workflow.
- Sending notifications when a file has been successfully transferred. (For a blog post that details this use case, see [Customize file delivery notifications using AWS Transfer Family managed workflows.](#))

To quickly replicate and standardize common post-upload file processing tasks spanning multiple business units in your organization, you can deploy workflows by using infrastructure as code (IaC). You can specify a managed workflow to be initiated on files that are uploaded in full. You can also specify a different managed workflow to be initiated on files that are only partially uploaded because of a premature session disconnect. Built-in exception handling helps you quickly react to file-processing outcomes, while offering you control over how to handle failures. In addition, each workflow step produces detailed logs, which you can audit to trace the data lineage.

To get started, perform the following tasks:

1. Set up your workflow to contain preprocessing actions, such as copying, tagging, and other steps based on your requirements. See [Create a workflow](#) for details.
2. Configure an execution role, which Transfer Family uses to run the workflow. See [IAM policies for workflows](#) for details.
3. Map the workflow to a server, so that on file arrival, the actions specified in this workflow are evaluated and initiated in real time. See [Configure and run a workflow](#) for details.

Related information

- To monitor your workflow executions, see [Using CloudWatch metrics for Transfer Family](#).
- For detailed execution logs and troubleshooting information, see [Troubleshoot workflow-related errors using Amazon CloudWatch](#).
- Transfer Family provides a blog post and a workshop that walk you through building a file transfer solution. This solution leverages AWS Transfer Family for managed SFTP/FTPS endpoints and Amazon Cognito and DynamoDB for user management.

The blog post is available at [Using Amazon Cognito as an identity provider with AWS Transfer Family and Amazon S3](#). You can view the details for the workshop [here](#).

- View [AWS Transfer Family Managed Workflows](#) for a brief introduction to Transfer Family workflows.

Topics

- [Create a workflow](#)
- [Use predefined steps](#)
- [Use custom file-processing steps](#)
- [IAM policies for workflows](#)
- [Exception handling for a workflow](#)
- [Monitor workflow execution](#)
- [Create a workflow from a template](#)
- [Remove a workflow from a Transfer Family server](#)
- [Managed workflows restrictions and limitations](#)

For more help getting started with managed workflows, see the following resources:

- [AWS Transfer Family managed workflows](#) demo video
- [Building a cloud-native file transfer platform using AWS Transfer Family workflows](#) blog post

Create a workflow

You can create a managed workflow by using the AWS Management Console, as described in this topic. To make the workflow creation process as easy as possible, contextual help panels are available for most of the sections in the console.

A workflow has two kinds of steps:

- **Nominal steps** – Nominal steps are file-processing steps that you want to apply to incoming files. If you select more than one nominal step, each step is processed in a linear sequence.
- **Exception-handling steps** – Exception handlers are file-processing steps that AWS Transfer Family executes in case any nominal steps fail or result in validation errors.

Create a workflow

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Workflows**.
3. On the **Workflows** page, choose **Create workflow**.
4. On the **Create workflow** page, enter a description. This description appears on the **Workflows** page.
5. In the **Nominal steps** section, choose **Add step**. Add one or more steps.
 - a. Choose a step type from the available options. For more information about the various step types, see [the section called "Use predefined steps"](#).
 - b. Choose **Next**, then configure parameters for the step.
 - c. Choose **Next**, then review the details for the step.
 - d. Choose **Create step** to add the step and continue.
 - e. Continue adding steps as needed. The maximum number of steps in a workflow is 8.
 - f. After you have added all of the necessary nominal steps, scroll down to the **Exception handlers – optional** section, and choose **Add step**.

Note

So that you are informed of failures in real time, we recommend that you set up exception handlers and steps to execute when your workflow fails.

6. To configure exception handlers, add steps in the same manner as described previously. If a file causes any step to throw an exception, your exception handlers are invoked one by one.
7. (Optional) Scroll down to the **Tags** section, and add tags for your workflow.
8. Review the configuration, and choose **Create workflow**.

Important

After you've created a workflow, you can't edit it, so make sure to review the configuration carefully.

Configure and run a workflow

Before you can run a workflow, you need to associate it with a Transfer Family server.

To configure Transfer Family to run a workflow on uploaded files

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Servers**.
 - To add the workflow to an existing server, choose the server that you want to use for your workflow.
 - Alternatively, create a new server and add the workflow to it. For more information, see [Configuring an SFTP, FTPS, or FTP server endpoint](#).
3. On the details page for the server, scroll down to the **Additional details** section, and then choose **Edit**.

Note

By default, servers do not have any associated workflows. You use the **Additional details** section to associate a workflow with the selected server.

4. On the **Edit additional details** page, in the **Managed workflows** section, select a workflow to be run on all uploads.

Note

If you do not already have a workflow, choose **Create a new Workflow** to create one.

- a. Choose the workflow ID to use.
- b. Choose an execution role. This is the role that Transfer Family assumes when executing the workflow's steps. For more information, see [IAM policies for workflows](#). Choose **Save**.

The screenshot shows the 'Managed workflows' configuration page. It has a title 'Managed workflows' with an 'Info' link. Below the title, there are three sections:

- Workflow for complete file uploads**: A sub-header with an 'Info' link. Below it is the instruction 'Select the workflow that AWS Transfer Family should run on all files that are uploaded in full via this server'. This section contains a dropdown menu with a placeholder 'w-...', a refresh button, and a 'Create a new Workflow' button with an external link icon.
- Workflow for partial file uploads**: A sub-header with an 'Info' link. Below it is the instruction 'Select the workflow that Transfer Family should run on all files that are only partially uploaded via this server'. This section also contains a dropdown menu with a placeholder 'w-...', a refresh button, and a 'Create a new Workflow' button with an external link icon.
- Managed workflows execution role**: A sub-header with an 'Info' link. Below it is the instruction 'Select the role that AWS Transfer Family should assume when executing a workflow'. This section contains a dropdown menu with a placeholder '...' and a refresh button.

Note

If you no longer want a workflow to be associated with the server, you can remove the association. For details, see [Remove a workflow from a Transfer Family server](#).

To execute a workflow

To execute a workflow, you upload a file to a Transfer Family server that you configured with an associated workflow.

Note

Anytime you remove a workflow from a server and replace it with a new one, or update server configuration (which impacts a workflow's execution role), you must wait approximately 10 minutes before executing the new workflow. The Transfer Family server caches the workflow details, and it takes 10 minutes for the server to refresh its cache. Additionally, you must log out of any active SFTP sessions, and then log back in after the 10-minute waiting period to see the changes.

Example

```
# Execute a workflow
> sftp bob@s-1234567890abcdef0.server.transfer.us-east-1.amazonaws.com

Connected to s-1234567890abcdef0.server.transfer.us-east-1.amazonaws.com.
sftp> put doc1.pdf
Uploading doc1.pdf to /DOC-EXAMPLE-BUCKET/home/users/bob/doc1.pdf
doc1.pdf                                     100% 5013KB
 601.0KB/s   00:08
sftp> exit
>
```

After your file has been uploaded, the action defined is performed on your file. For example, if your workflow contains a copy step, the file is copied to the location that you defined in that step. You can use Amazon CloudWatch Logs to track the steps that executed and their execution status.

View workflow details

You can view details about previously created workflows or to workflow executions. To view these details, you can use the console or the AWS Command Line Interface (AWS CLI).

Console**View workflow details**

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Workflows**.
3. On the **Workflows** page, choose a workflow.

The workflow details page opens.

The screenshot shows the AWS Transfer Family console interface. On the left is a navigation sidebar with options like Servers, Connectors, AS2 Trading Partners, Certificates, Profiles, Workflows, Feature Spotlight, What's New, and Documentation. The main content area displays the details for a specific workflow. At the top, there's a breadcrumb trail: Transfer Family > Workflows > w-1234567890abcdef0. Below this is a 'Delete' button. The 'Description' section shows the workflow name 'Test my workflow' and its description 'Test workflow A'. The 'Nominal steps (1)' section contains a table with one step: a 'tag_step' of type 'TAG'. The 'Exception handlers (1)' section contains a table with one handler: 'delete_if_exception' of type 'DELETE'. The 'In-flight executions (0)' section shows a search bar and a message 'No executions to display'. Finally, the 'Tags (1)' section has a 'Manage tags' button and a note about using tags to organize workflows.

CLI

To view the workflow details, use the `describe-workflow` CLI command, as shown in the following example. Replace the workflow ID `w-1234567890abcdef0` with your own value. For more information, see [describe-workflow](#) in the *AWS CLI Command Reference*.

```
# View Workflow details
> aws transfer describe-workflow --workflow-id w-1234567890abcdef0
{
  "Workflow": {
    "Arn": "arn:aws:transfer:us-east-1:111122223333:workflow/w-1234567890abcdef0",
    "WorkflowId": "w-1234567890abcdef0",
    "Name": "Copy file to shared_files",
    "Steps": [
```

```

    {
      "Type": "COPY",
      "CopyStepDetails": {
        "Name": "Copy to shared",
        "FileLocation": {
          "S3FileLocation": {
            "Bucket": "DOC-EXAMPLE-BUCKET",
            "Key": "home/shared_files/"
          }
        }
      }
    },
    "OnException": {}
  }
}

```

If your workflow was created as part of an AWS CloudFormation stack, you can manage the workflow using the AWS CloudFormation console (<https://console.aws.amazon.com/cloudformation>).

The screenshot shows the AWS Transfer Family console interface. At the top, there is a breadcrumb navigation: [Transfer Family](#) > [Workflows](#) > [w-...](#). Below this, the workflow name [W-...](#) is displayed with a [Delete](#) button. A blue information banner states: "This workflow belongs to the AWS CloudFormation stack **stack-transfer-notifications-blog**. [Manage this stack](#) on the CloudFormation console." Below the banner is a **Description** section with a table:

Name	Workflow description
Stack-managed	Workflow for blogpost

Below the description is a **Nominal steps (4) Info** section with a table:

Number	Description	Type	Configuration
1	Lambda	CUSTOM	Details
2	CopyFile	COPY	Details
3	TagFileNotified	TAG	Details
4	DeleteOriginalSourceFile	DELETE	Details

Use predefined steps

When you're creating a workflow, you can choose to add one of the following predefined steps discussed in this topic. You can also choose to add your own custom file-processing steps. For more information, see [the section called "Use custom file-processing steps"](#).

Topics

- [Copy file](#)
- [Decrypt file](#)
- [Tag file](#)
- [Delete file](#)
- [Named variables for workflows](#)
- [Example tag and delete workflow](#)

Copy file

A copy file step creates a copy of the uploaded file in a new Amazon S3 location. Currently, you can use a copy file step only with Amazon S3.

The following copy file step copies files into the test folder in the file-test destination bucket.

If the copy file step is not the first step of your workflow, you can specify the **File location**. By specifying the file location, you can copy either the file that was used in the previous step or the original file that was uploaded. You can use this feature to make multiple copies of the original file while keeping the source file intact for file archival and records retention. For an example, see [Example tag and delete workflow](#).

Configure copy parameters

Step name

File location

Select the file location to use as an input for this step

Copy the file created from previous step to a new location
Input file is selected from the previous step's output

Copy the original source file to a new location
Originally uploaded file

Destination bucket name

Destination key prefix

If you are copying files into a folder, specify / at the end of the prefix name. Use `${transfer:UserName}` or `${transfer:UploadDate}` to parametrize destination prefix by username or upload date respectively.

Overwrite existing

Provide the bucket and key details

You must provide the bucket name and a key for the destination of the copy file step. The key can be either a path name or a file name. Whether the key is treated as a path name or a file name is determined by whether you end the key with the forward slash (/) character.

If the final character is /, your file is copied to the folder, and its name does not change. If the final character is alphanumeric, your uploaded file is renamed to the key value. In this case, if a file with that name already exists, the behavior depends on the setting for the **Overwrite existing** field.

- If **Overwrite existing** is selected, the existing file is replaced with the file being processed.
- If **Overwrite existing** is not selected, nothing happens, and the workflow processing stops.

Tip

If concurrent writes are executed on the same file path, it may result in unexpected behavior when overwriting files.

For example, if your key value is `test/`, your uploaded files are copied to the `test` folder. If your key value is `test/today`, (and **Overwrite existing** is selected) every file you upload is copied to a file named `today` in the `test` folder, and each succeeding file overwrites the previous one.

Note

Amazon S3 supports buckets and objects, and there is no hierarchy. However, you can use prefixes and delimiters in object key names to imply a hierarchy and organize your data in a way similar to folders.

Use a named variable in a copy file step

In a copy file step, you can use a variable to dynamically copy your files into user-specific folders. Currently, you can use `${transfer:UserName}` or `${transfer:UploadDate}` as a variable to copy files to a destination location for the given user who's uploading files, or based on the current date.

In the following example, if the user `richard-roe` uploads a file, it gets copied into the `file-test2/richard-roe/processed/` folder. If the user `mary-major` uploads a file, it gets copied into the `file-test2/mary-major/processed/` folder.

Configure parameters

Configure copy parameters

Step name

Destination bucket name

Destination key prefix

If you are copying files into a folder, specify / at the end of the prefix name. Use `${transfer:UserName}` or `${transfer:UploadDate}` to parametrize destination prefix by username or upload date respectively.

Overwrite existing

Similarly, you can use `${transfer:UploadDate}` as a variable to copy files to a destination location named for the current date. In the following example, if you set the destination to `${transfer:UploadDate}/processed` on February 1, 2022, files uploaded are copied into the `file-test2/2022-02-01/processed/` folder.

Configure copy parameters

Step name

dynamic-copy-date

Destination bucket name

file-test2

Destination key prefix

If you are copying files into a folder, specify / at the end of the prefix name. Use `${transfer:UserName}` or `${transfer:UploadDate}` to parametrize destination prefix by username or upload date respectively.

`${transfer:UploadDate}/processed`

Overwrite existing

You can also use both of these variables together, combining their functionality. For example:

- You could set the **Destination key prefix** to **folder/\${transfer:UserName}/\${transfer:UploadDate}/**, which would create nested folders, for example `folder/marymajor/2023-01-05/`.
- You could set the **Destination key prefix** to **folder/\${transfer:UserName}-\${transfer:UploadDate}/**, to concatenate the two variables, for example `folder/marymajor-2023-01-05/`.

IAM permissions for copy step

To allow a copy step to succeed, make sure the execution role for your workflow contains the following permissions.

```
{
  "Sid": "ListBucket",
  "Effect": "Allow",
```

```
    "Action": "s3:ListBucket",
    "Resource": [
      "arn:aws:s3:::destination-bucket-name"
    ]
  },
  {
    "Sid": "HomeDirObjectAccess",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:DeleteObjectVersion",
      "s3:DeleteObject",
      "s3:GetObjectVersion"
    ],
    "Resource": "arn:aws:s3:::destination-bucket-name/*"
  }
}
```

Note

The `s3:ListBucket` permission is only necessary if you do not select **Overwrite existing**. This permission checks your bucket to see if a file with the same name already exists. If you have selected **Overwrite existing**, the workflow doesn't need to check for the file, and can just write it.

If your Amazon S3 files have tags, you need to add one or two permissions to your IAM policy.

- Add `s3:GetObjectTagging` for an Amazon S3 file that isn't versioned.
- Add `s3:GetObjectVersionTagging` for an Amazon S3 file that is versioned.

Decrypt file

The AWS storage blog has a post that describes how to simply decrypt files without writing any code using Transfer Family Managed workflows, [Encrypt and decrypt files with PGP and AWS Transfer Family](#).

Use PGP decryption in your workflow

Transfer Family has built-in support for Pretty Good Privacy (PGP) decryption. You can use PGP decryption on files that are uploaded over SFTP, FTPS, or FTP to Amazon Simple Storage Service (Amazon S3) or Amazon Elastic File System (Amazon EFS).

To use PGP decryption, you must create and store the PGP private keys that will be used for decryption of your files. Your users can then encrypt files by using corresponding PGP encryption keys before uploading the files to your Transfer Family server. After you receive the encrypted files, you can decrypt those files in your workflow. For a detailed tutorial, see [Setting up a managed workflow for decrypting a file](#).

To use PGP decryption in your workflow

1. Identify a Transfer Family server to host your workflow, or create a new one. You need to have the server ID before you can store your PGP keys in AWS Secrets Manager with the correct secret name.
2. Store your PGP key in AWS Secrets Manager under the required secret name. For details, see [Manage PGP keys](#). Workflows can automatically locate the correct PGP key to be used for decryption based on the secret name in Secrets Manager.

Note

When you store secrets in Secrets Manager, your AWS account incurs charges. For information about pricing, see [AWS Secrets Manager Pricing](#).

3. Encrypt a file by using your PGP key pair. (For a list of supported clients, see [Supported PGP clients](#).) If you are using the command line, run the following command. To use this command, replace *username@example.com* with the email address that you used to create the PGP key pair. Replace *testfile.txt* with the name of the file that you want to encrypt.

```
gpg -e -r username@example.com testfile.txt
```

4. Upload the encrypted file to your Transfer Family server.
5. Configure a decryption step in your workflow. For more information, see [Add a decryption step](#).

Add a decryption step

A decryption step decrypts an encrypted file that was uploaded to Amazon S3 or Amazon EFS as part of your workflow. For details about configuring decryption, see [Use PGP decryption in your workflow](#).

When you create your decryption step for a workflow, you must specify the destination for the decrypted files. You must also select whether to overwrite existing files if a file already exists at the destination location. You can monitor the decryption workflow results and get audit logs for each file in real time by using Amazon CloudWatch Logs.

After you choose the **Decrypt file** type for your step, the **Configure parameters** page appears. Fill in the values for the **Configure PGP decryption parameters** section.

The available options are as follows:

- **Step name** – Enter a descriptive name for the step.
- **File location** – By specifying the file location, you can decrypt either the file that was used in the previous step or the original file that was uploaded.

Note

This parameter is not available if this step is the first step of the workflow.

- **Destination for decrypted files** – Choose an Amazon S3 bucket or an Amazon EFS file system as the destination for the decrypted file.
 - If you choose Amazon S3, you must provide a destination bucket name and a destination key prefix. To parameterize the destination key prefix by username, enter `${transfer:UserName}` for **Destination key prefix**. Similarly, to parameterize the destination key prefix by upload date, enter `${Transfer:UploadDate}` for **Destination key prefix**.
 - If you choose Amazon EFS, you must provide a destination file system and path.

Note

The storage option that you choose here must match the storage system that's used by the Transfer Family server with which this workflow is associated. Otherwise, you will receive an error when you attempt to run this workflow.

- **Overwrite existing** – If you upload a file, and a file with the same filename already exists at the destination, the behavior depends on the setting for this parameter:
 - If **Overwrite existing** is selected, the existing file is replaced with the file being processed.
 - If **Overwrite existing** is not selected, nothing happens, and the workflow processing stops.

 **Tip**

If concurrent writes are executed on the same file path, it may result in unexpected behavior when overwriting files.

The following screenshot shows an example of the options that you might choose for your decrypt file step.

Step 1
[Choose step type](#)

Step 2
Configure parameters

Step 3
Review and create

Configure parameters

Configure PGP decryption parameters

Store your PGP private key(s) and passphrase(s) in AWS Secrets Manager. [Learn more](#)

Refer to the [AWS Transfer Family pricing page](#) for pricing details.

Step name

File location
Select the file location to use as an input for this step

Apply on the file created from the previous step
Input file is selected from the previous step's output

Apply on the original file
Originally uploaded file

Destination for decrypted files
Choose an S3 bucket or an EFS file system for storing decrypted files.

Amazon S3
Store your decrypted files as Amazon S3 objects

Amazon EFS
Store your decrypted files in an EFS file system

Destination bucket name

Destination key prefix
If you are decrypting files into a folder, specify / at the end of the prefix name. Use `${transfer:UserName}` or `${transfer:UploadDate}` to parametrize the destination prefix by username or upload date respectively.

Overwrite existing
Overwrite if a file with the same file name already exists at the destination.

IAM permissions for decrypt step

To allow a decrypt step to succeed, make sure the execution role for your workflow contains the following permissions.

```
{
    "Sid": "ListBucket",
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": [
        "arn:aws:s3::destination-bucket-name"
    ]
},
{
    "Sid": "HomeDirObjectAccess",
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:GetObjectVersion"
    ],
    "Resource": "arn:aws:s3::destination-bucket-name/*"
},
{
    "Sid": "Decrypt",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue",
    ],
    "Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/transfer/
**
}
```

Note

The `s3:ListBucket` permission is only necessary if you do not select **Overwrite existing**. This permission checks your bucket to see if a file with the same name already exists. If you have selected **Overwrite existing**, the workflow doesn't need to check for the file, and can just write it.

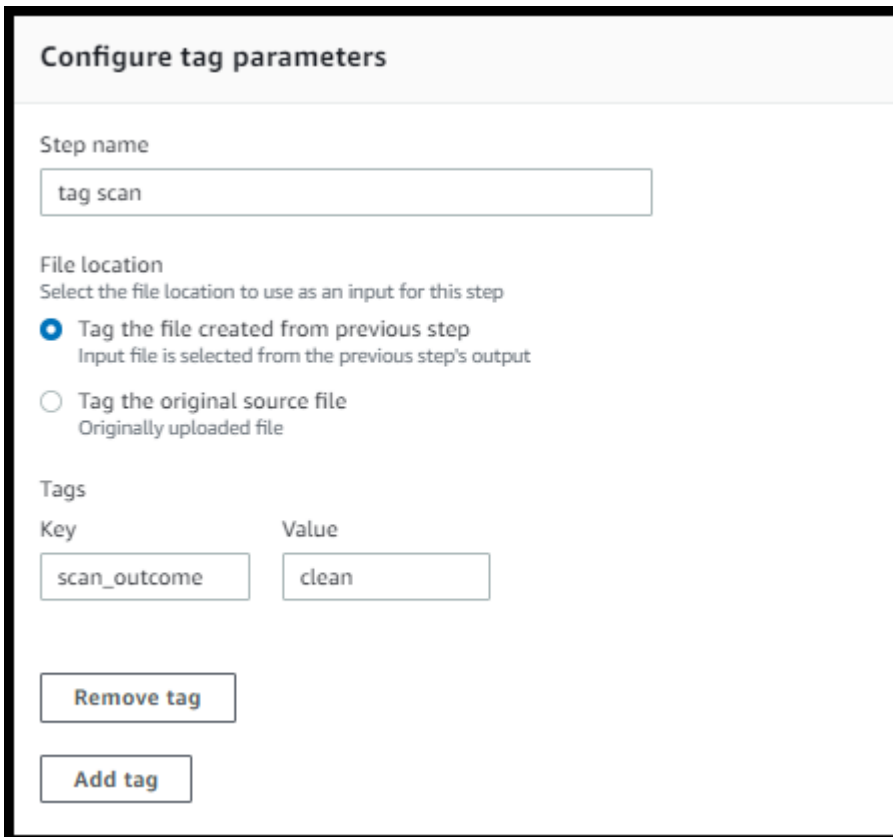
If your Amazon S3 files have tags, you need to add one or two permissions to your IAM policy.

- Add `s3:GetObjectTagging` for an Amazon S3 file that isn't versioned.
- Add `s3:GetObjectVersionTagging` for an Amazon S3 file that is versioned.

Tag file

To tag incoming files for further downstream processing, use a tag step. Enter the value of the tag that you would like to assign to the incoming files. Currently, the tag operation is supported only if you are using Amazon S3 for your Transfer Family server storage.

The following example tag step assigns `scan_outcome` and `clean` as the tag key and value, respectively.



Configure tag parameters

Step name
tag scan

File location
Select the file location to use as an input for this step

Tag the file created from previous step
Input file is selected from the previous step's output

Tag the original source file
Originally uploaded file

Tags

Key	Value
scan_outcome	clean

Remove tag

Add tag

To allow a tag step to succeed, make sure the execution role for your workflow contains the following permissions.

```
{
```



```

    "Sid": "Tag",
    "Effect": "Allow",
    "Action": [
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
    ],
    "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
    ]
}

```

Note

If your workflow contains a tag step that runs before either a copy or decrypt step, you need to add one or two permissions to your IAM policy.

- Add `s3:GetObjectTagging` for an Amazon S3 file that isn't versioned.
- Add `s3:GetObjectVersionTagging` for an Amazon S3 file that is versioned.

Delete file

To delete a processed file from a previous workflow step or to delete the originally uploaded file, use a delete file step.

Configure delete parameters

Step name

File location

Select the file location to use as an input for this step

Delete the file created from previous step
Input file is selected from the previous step's output

Delete the original source file
Originally uploaded file

To allow a delete step to succeed, make sure the execution role for your workflow contains the following permissions.

```
{
```

```
    "Sid": "Delete",
    "Effect": "Allow",
    "Action": [
        "s3:DeleteObjectVersion",
        "s3:DeleteObject"
    ],
    "Resource": "arn:aws:secretsmanager:region:account-ID:secret:aws/transfer/
* "
}
```

Named variables for workflows

For copy and decrypt steps, you can use a variable to dynamically perform actions. Currently, AWS Transfer Family supports the following named variables.

- Use `${transfer:UserName}` to copy or decrypt files to a destination based on the user who's uploading the files.
- Use `${transfer:UploadDate}` to copy or decrypt files to a destination location based on the current date.

Example tag and delete workflow

The following example illustrates a workflow that tags incoming files that need to be processed by a downstream application, such as a data analytics platform. After tagging the incoming file, the workflow then deletes the originally uploaded file to save on storage costs.

Console

Example tag and move workflow

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Workflows**.
3. On the **Workflows** page, choose **Create workflow**.
4. On the **Create workflow** page, enter a description. This description appears on the **Workflows** page.
5. Add the first step (copy).
 - a. In the **Nominal steps** section, choose **Add step**.

- b. Choose **Copy file**, then choose **Next**.
- c. Enter a step name, then select a destination bucket and a key prefix.

The screenshot shows the 'Configure parameters' step in the AWS Transfer Family console. On the left, a sidebar lists three steps: 'Step 1 Choose step type', 'Step 2 Configure parameters' (which is the active step), and 'Step 3 Review and create'. The main content area is titled 'Configure parameters' and contains a section for 'Configure copy parameters'. This section includes three input fields: 'Step name' with the value 'copy-step-first-step', 'Destination bucket name' with a dropdown menu showing 'example-bucket', and 'Destination key prefix' with the value 'test/'. Below these fields is a checkbox labeled 'Overwrite existing' which is currently unchecked. A note under the 'Destination key prefix' field explains that users can use placeholders like \${transfer:UserName} or \${transfer:UploadDate} to parametrize the prefix.

- d. Choose **Next**, then review the details for the step.
 - e. Choose **Create step** to add the step and continue.
6. Add the second step (tag).
- a. In the **Nominal steps** section, choose **Add step**.
 - b. Choose **Tag file**, then choose **Next**.
 - c. Enter a step name.
 - d. For **File location**, select **Tag the file created from previous step**.
 - e. Enter a **Key** and **Value**.

Configure tag parameters

Step name
tag scan

File location
Select the file location to use as an input for this step

- Tag the file created from previous step
Input file is selected from the previous step's output
- Tag the original source file
Originally uploaded file

Tags

Key	Value
scan_outcome	clean

Remove tag

Add tag

- f. Choose **Next**, then review the details for the step.
 - g. Choose **Create step** to add the step and continue.
7. Add the third step (delete).
- a. In the **Nominal steps** section, choose **Add step**.
 - b. Choose **Delete file**, then choose **Next**.

Configure delete parameters

Step name
delete original file

File location
Select the file location to use as an input for this step

- Delete the file created from previous step
Input file is selected from the previous step's output
- Delete the original source file
Originally uploaded file

- c. Enter a step name.

- d. For **File location**, select **Delete the original source file**.
 - e. Choose **Next**, then review the details for the step.
 - f. Choose **Create step** to add the step and continue.
8. Review the workflow configuration, and then choose **Create workflow**.

CLI

Example tag and move workflow

1. Save the following code into a file; for example, `tagAndMoveWorkflow.json`. Replace each *user input placeholder* with your own information.

```
[
  {
    "Type": "COPY",
    "CopyStepDetails": {
      "Name": "CopyStep",
      "DestinationFileLocation": {
        "S3FileLocation": {
          "Bucket": "DOC-EXAMPLE-BUCKET",
          "Key": "test/"
        }
      }
    }
  },
  {
    "Type": "TAG",
    "TagStepDetails": {
      "Name": "TagStep",
      "Tags": [
        {
          "Key": "name",
          "Value": "demo"
        }
      ],
      "SourceFileLocation": "${previous.file}"
    }
  },
  {
    "Type": "DELETE",
    "DeleteStepDetails":{
```

```

        "Name": "DeleteStep",
        "SourceFileLocation": "${original.file}"
    }
}
]

```

The first step copies the uploaded file to a new Amazon S3 location. The second step adds a tag (key-value pair) to the file (previous .file) that was copied to the new location. And, finally, the third step deletes the original file (original.file).

2. Create a workflow from the saved file. Replace each *user input placeholder* with your own information.

```

aws transfer create-workflow --description "short-description" --steps
file://path-to-file --region region-ID

```

For example:

```

aws transfer create-workflow --description "copy-tag-delete workflow" --steps
file://tagAndMoveWorkflow.json --region us-east-1

```

Note

For more details about using files to load parameters, see [How to load parameters from a file](#).

3. Update an existing server.

Note

This step assumes you already have a Transfer Family server and you want to associate a workflow with it. If not, see [Configuring an SFTP, FTPS, or FTP server endpoint](#). Replace each *user input placeholder* with your own information.

```

aws transfer update-server --server-id server-ID --region region-ID
--workflow-details '{"OnUpload": [{ "WorkflowId": "workflow-ID", "ExecutionRole": "execution-role-ARN" } ]}'

```

For example:

```
aws transfer update-server --server-id s-1234567890abcdef0 --region us-east-2
  --workflow-details '{"OnUpload":[{"WorkflowId": "w-
  abcdef01234567890","ExecutionRole": "arn:aws:iam::111111111111:role/nikki-wolf-
  execution-role"}]}'
```

Use custom file-processing steps

By using a custom file-processing step, you can Bring Your Own file-processing logic using AWS Lambda. Upon file arrival, a Transfer Family server invokes a Lambda function that contains custom file-processing logic, such as encrypting files, scanning for malware, or checking for incorrect file types. In the following example, the target AWS Lambda function is used to process the output file from the previous step.

Configure custom parameters

Step name

File location

Select the file location to use as an input for this step

Apply custom processing to the file created from previous step
Input file is selected from the previous step's output

Apply custom processing to the original source file
Originally uploaded file

Target

Timeout (seconds)

Note

For an example Lambda function, see [Example Lambda function for a custom workflow step](#). For example events (including the location for files passed into the Lambda), see [Example events sent to AWS Lambda upon file upload](#).

With a custom workflow step, you must configure the Lambda function to call the [SendWorkflowStepState](#) API operation. `SendWorkflowStepState` notifies the workflow execution that the step was completed with either a success or a failure status. The status of the `SendWorkflowStepState` API operation invokes an exception handler step or a nominal step in the linear sequence, based on the outcome of the Lambda function.

If the Lambda function fails or times out, the step fails, and you see `StepErrored` in your CloudWatch logs. If the Lambda function is part of the nominal step and the function responds to `SendWorkflowStepState` with `Status="FAILURE"` or times out, the flow continues with the exception handler steps. In this case, the workflow does not continue to execute the remaining (if any) nominal steps. For more details, see [Exception handling for a workflow](#).

When you call the `SendWorkflowStepState` API operation, you must send the following parameters:

```
{
  "ExecutionId": "string",
  "Status": "string",
  "Token": "string",
  "WorkflowId": "string"
}
```

You can extract the `ExecutionId`, `Token`, and `WorkflowId` from the input event that is passed when the Lambda function executes (examples are shown in the following sections). The `Status` value can be either `SUCCESS` or `FAILURE`.

To be able to call the `SendWorkflowStepState` API operation from your Lambda function, you must use a version of the AWS SDK that was published after [Managed Workflows were introduced](#).

Using multiple Lambda functions consecutively

When you use multiple custom steps one after the other, the **File location** option works differently than if you use only a single custom step. Transfer Family doesn't support passing the Lambda-

processed file back to use as the next step's input. So, if you have multiple custom steps all configured to use the `previous.file` option, they all use the same file location (the input file location for the first custom step).

Note

The `previous.file` setting also works differently if you have a predefined step (tag, copy, decrypt, or delete) after a custom step. If the predefined step is configured to use the `previous.file` setting, the predefined step uses the same input file that's used by the custom step. The processed file from the custom step is not passed to the predefined step.

Accessing a file after custom processing

If you're using Amazon S3 as your storage, and if your workflow includes a custom step that performs actions on the originally uploaded file, subsequent steps cannot access that processed file. That is, any step after the custom step cannot reference the updated file from the custom step output.

For example, suppose that you have the following three steps in your workflow.

- **Step 1** – Upload a file named `example-file.txt`.
- **Step 2** – Invoke a Lambda function that changes `example-file.txt` in some way.
- **Step 3** – Attempt to perform further processing on the updated version of `example-file.txt`.

If you configure the `sourceFileLocation` for Step 3 to be `${original.file}`, Step 3 uses the original file location from when the server uploaded the file to storage in Step 1. If you're using `${previous.file}` for Step 3, Step 3 reuses the file location that Step 2 used as input.

Therefore, Step 3 causes an error. For example, if step 3 attempts to copy the updated `example-file.txt`, you receive the following error:

```
{
  "type": "StepErrored",
  "details": {
    "errorType": "NOT_FOUND",
    "errorMessage": "ETag constraint not met (Service: null; Status Code: 412; Error Code: null; Request ID: null; S3 Extended Request ID: null; Proxy: null)",
    "stepType": "COPY",
```

```
    "stepName": "CopyFile"
  },
```

This error occurs because the custom step modifies the entity tag (ETag) for `example-file.txt` so that it doesn't match the original file.

Note

This behavior doesn't occur if you're using Amazon EFS because Amazon EFS doesn't use entity tags to identify files.

Example events sent to AWS Lambda upon file upload

The following examples show the events that are sent to AWS Lambda when a file upload is complete. One example uses a Transfer Family server where the domain is configured with Amazon S3. The other example uses a Transfer Family server where the domain uses Amazon EFS.

Custom step that uses an Amazon S3 domain

```
{
  "token": "MzI0Nzc4ZDktMGRmMi00MjFhLTgxMjUtYWZmZmRmODNkYjc0",
  "serviceMetadata": {
    "executionDetails": {
      "workflowId": "w-1234567890example",
      "executionId": "abcd1234-aa11-bb22-cc33-abcdef123456"
    },
    "transferDetails": {
      "sessionId": "36688ff5d2deda8c",
      "userName": "myuser",
      "serverId": "s-example1234567890"
    }
  },
  "fileLocation": {
    "domain": "S3",
    "bucket": "DOC-EXAMPLE-BUCKET",
    "key": "path/to/mykey",
    "eTag": "d8e8fca2dc0f896fd7cb4cb0031ba249",
    "versionId": null
  }
}
```

Custom step that uses an Amazon EFS domain

```
{
  "token": "MTg0N2Y3N2UtNWI5Ny00ZmZlLTk5YTgtZTU3YzViYjllNmZm",
  "serviceMetadata": {
    "executionDetails": {
      "workflowId": "w-1234567890example",
      "executionId": "abcd1234-aa11-bb22-cc33-abcdef123456"
    },
    "transferDetails": {
      "sessionId": "36688ff5d2deda8c",
      "userName": "myuser",
      "serverId": "s-example1234567890"
    }
  },
  "fileLocation": {
    "domain": "EFS",
    "fileSystemId": "fs-1234567",
    "path": "/path/to/myfile"
  }
}
```

Example Lambda function for a custom workflow step

The following Lambda function extracts the information regarding the execution status, and then calls the [SendWorkflowStepState](#) API operation to return the status to the workflow for the step—either `SUCCESS` or `FAILURE`. Before your function calls the `SendWorkflowStepState` API operation, you can configure Lambda to take an action based on your workflow logic.

```
import json
import boto3

transfer = boto3.client('transfer')

def lambda_handler(event, context):
    print(json.dumps(event))

    # call the SendWorkflowStepState API to notify the workflow about the step's
    # SUCCESS or FAILURE status
    response = transfer.send_workflow_step_state(
        WorkflowId=event['serviceMetadata']['executionDetails']['workflowId'],
```

```

        ExecutionId=event['serviceMetadata']['executionDetails']['executionId'],
        Token=event['token'],
        Status='SUCCESS|FAILURE'
    )

    print(json.dumps(response))

    return {
        'statusCode': 200,
        'body': json.dumps(response)
    }

```

IAM permissions for a custom step

To allow a step that calls a Lambda to succeed, make sure the execution role for your workflow contains the following permissions.

```

{
    "Sid": "Custom",
    "Effect": "Allow",
    "Action": [
        "lambda:InvokeFunction"
    ],
    "Resource": [
        "arn:aws:lambda:region:account-id:function:function-name"
    ]
}

```

IAM policies for workflows

When you add a workflow to a server, you must select an execution role. The server uses this role when it executes the workflow. If the role does not have the proper permissions, AWS Transfer Family cannot run the workflow.

This section describes one possible set of AWS Identity and Access Management (IAM) permissions that you can use to execute a workflow. Other examples are described later in this topic.

Note

If your Amazon S3 files have tags, you need to add one or two permissions to your IAM policy.

- Add `s3:GetObjectTagging` for an Amazon S3 file that isn't versioned.
- Add `s3:GetObjectVersionTagging` for an Amazon S3 file that is versioned.

To create an execution role for your workflow

1. Create a new IAM role, and add the AWS managed policy `AWSTransferFullAccess` to the role. For more information about creating a new IAM role, see [the section called "Create an IAM role and policy"](#).
2. Create another policy with the following permissions, and attach it to your role. Replace each *user input placeholder* with your own information.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConsoleAccess",
      "Effect": "Allow",
      "Action": "s3:GetBucketLocation",
      "Resource": "*"
    },
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
      ]
    },
    {
      "Sid": "AllObjectActions",
      "Effect": "Allow",
      "Action": "s3:*Object",
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ]
    },
    {
      "Sid": "GetObjectVersion",
      "Effect": "Allow",
      "Action": "s3:GetObjectVersion",
```

```

        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        ]
    },
    {
        "Sid": "Custom",
        "Effect": "Allow",
        "Action": [
            "lambda:InvokeFunction"
        ],
        "Resource": [
            "arn:aws:lambda:region:account-id:function:function-name"
        ]
    },
    {
        "Sid": "Tag",
        "Effect": "Allow",
        "Action": [
            "s3:PutObjectTagging",
            "s3:PutObjectVersionTagging"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        ]
    }
]
}

```

3. Save this role and specify it as the execution role when you add a workflow to a server.

Note

When you're constructing IAM roles, AWS recommends that you restrict access to your resources as much as is possible for your workflow.

Workflow trust relationships

Workflow execution roles also require a trust relationship with `transfer.amazonaws.com`. To establish a trust relationship for AWS Transfer Family, see [To establish a trust relationship](#).

While you're establishing your trust relationship, you can also take steps to avoid the *confused deputy* problem. For a description of this problem, as well as examples of how to avoid it, see [the section called "Cross-service confused deputy prevention"](#).

Example execution role: Decrypt, copy, and tag

If you have workflows that include tagging, copying, and decrypt steps, you can use the following IAM policy. Replace each *user input placeholder* with your own information.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CopyRead",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::source-bucket-name/*"
    },
    {
      "Sid": "CopyWrite",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectTagging"
      ],
      "Resource": "arn:aws:s3:::destination-bucket-name/*"
    },
    {
      "Sid": "CopyList",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": [
        "arn:aws:s3:::source-bucket-name",
        "arn:aws:s3:::destination-bucket-name"
      ]
    },
    {
      "Sid": "Tag",
      "Effect": "Allow",
```

```

    "Action": [
      "s3:PutObjectTagging",
      "s3:PutObjectVersionTagging"
    ],
    "Resource": "arn:aws:s3:::destination-bucket-name/*",
    "Condition": {
      "StringEquals": {
        "s3:RequestObjectTag/Archive": "yes"
      }
    }
  },
  {
    "Sid": "ListBucket",
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": [
      "arn:aws:s3:::destination-bucket-name"
    ]
  },
  {
    "Sid": "HomeDirObjectAccess",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:DeleteObjectVersion",
      "s3:DeleteObject",
      "s3:GetObjectVersion"
    ],
    "Resource": "arn:aws:s3:::destination-bucket-name/*"
  },
  {
    "Sid": "Decrypt",
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": "arn:aws:secretsmanager:region:account-ID:secret:aws/transfer/
*"
  }
]
}

```


Example execution role: Run function and delete

In this example, you have a workflow that invokes an AWS Lambda function. If the workflow deletes the uploaded file and has an exception handler step to act upon a failed workflow execution in the previous step, use the following IAM policy. Replace each *user input placeholder* with your own information.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Delete",
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject",
        "s3:DeleteObjectVersion"
      ],
      "Resource": "arn:aws:s3:::bucket-name"
    },
    {
      "Sid": "Custom",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "arn:aws:lambda:region:account-id:function:function-name"
      ]
    }
  ]
}
```

Exception handling for a workflow

If any errors occur during a workflow's execution, the exception-handling steps that you specified are executed. You specify the error-handling steps for a workflow in the same manner as you specify the nominal steps for the workflow. For example, suppose that you've configured custom processing in nominal steps to validate incoming files. If the file validation fails, an exception-handling step can send an email to the administrator.

The following example workflow contains two steps:

- One nominal step that checks whether the uploaded file is in CSV format
- An exception-handling step that sends an email in case the uploaded file is not in CSV format, and the nominal step fails

To initiate the exception-handling step, the AWS Lambda function in the nominal step must respond with `Status="FAILURE"`. For more information about error handling in workflows, see [the section called "Use custom file-processing steps"](#).

w-1234567890abcdef0
Delete

Description

Name Delete after upload	Workflow description test-my-workflow
-----------------------------	--

Nominal steps (1) [Info](#)

Number	Description	Type	Configuration
1	is-CSV	CUSTOM	Details

Exception handlers (1) [Info](#)

Number	Description	Type	Configuration
1	send-email	CUSTOM	Details

Monitor workflow execution

Amazon CloudWatch monitors your AWS resources and the applications that you run in the AWS Cloud in real time. You can use Amazon CloudWatch to collect and track metrics, which are variables that you can measure for your workflows. You can view workflow metrics and consolidated logs by using Amazon CloudWatch.

CloudWatch logging for a workflow

CloudWatch provides consolidated auditing and logging for workflow progress and results.

View Amazon CloudWatch logs for workflows

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the left navigation pane, choose **Logs**, then choose **Log groups**.
3. On the **Log groups** page, on the navigation bar, choose the correct Region for your AWS Transfer Family server.
4. Choose the log group that corresponds to your server.

For example, if your server ID is `s-1234567890abcdef0`, your log group is `/aws/transfer/s-1234567890abcdef0`.

5. On the log group details page for your server, the most recent log streams are displayed. There are two log streams for the user that you are exploring:
 - One for each Secure Shell (SSH) File Transfer Protocol (SFTP) session.
 - One for the workflow that is being executed for your server. The format for the log stream for the workflow is `username.workflowID.uniqueStreamSuffix`.

For example, if your user is `mary-major`, you have the following log streams:

```
mary-major-east.1234567890abcdef0  
mary.w-abcdef01234567890.021345abcdef6789
```

Note

The 16-digit alphanumeric identifiers listed in this example are fictitious. The values that you see in Amazon CloudWatch are different.

The **Log events** page for `mary-major-usa-east.1234567890abcdef0` displays the details for each user session, and the `mary.w-abcdef01234567890.021345abcdef6789` log stream contains the details for the workflow.

The following is a sample log stream for `mary.w-abcdef01234567890.021345abcdef6789`, based on a workflow (`w-abcdef01234567890`) that contains a copy step.

```
{  
  "type": "ExecutionStarted",
```

```

"details": {
  "input": {
    "initialFileLocation": {
      "bucket": "DOC-EXAMPLE-BUCKET",
      "key": "mary/workflowSteps2.json",
      "versionId": "version-id",
      "etag": "etag-id"
    }
  }
},
"workflowId": "w-abcdef01234567890",
"executionId": "execution-id",
"transferDetails": {
  "serverId": "s-server-id",
  "username": "mary",
  "sessionId": "session-id"
},
{
  "type": "StepStarted",
  "details": {
    "input": {
      "fileLocation": {
        "backingStore": "S3",
        "bucket": "DOC-EXAMPLE-BUCKET",
        "key": "mary/workflowSteps2.json",
        "versionId": "version-id",
        "etag": "etag-id"
      }
    },
    "stepType": "COPY",
    "stepName": "copyToShared"
  },
  "workflowId": "w-abcdef01234567890",
  "executionId": "execution-id",
  "transferDetails": {
    "serverId": "s-server-id",
    "username": "mary",
    "sessionId": "session-id"
  }
},
{
  "type": "StepCompleted",
  "details": {

```

```

    "output": {},
    "stepType": "COPY",
    "stepName": "copyToShared"
  },
  "workflowId": "w-abcdef01234567890",
  "executionId": "execution-id",
  "transferDetails": {
    "serverId": "server-id",
    "username": "mary",
    "sessionId": "session-id"
  }
},
{
  "type": "ExecutionCompleted",
  "details": {},
  "workflowId": "w-abcdef01234567890",
  "executionId": "execution-id",
  "transferDetails": {
    "serverId": "s-server-id",
    "username": "mary",
    "sessionId": "session-id"
  }
}
}

```

CloudWatch metrics for workflows

AWS Transfer Family provides several metrics for workflows. You can view metrics for how many workflows executions started, completed successfully, and failed in the previous minute. All of the CloudWatch metrics for Transfer Family are described in [Using CloudWatch metrics for Transfer Family](#).

Create a workflow from a template

You can deploy an AWS CloudFormation stack that creates a workflow and a server from a template. This procedure contains an example that you can use to quickly deploy a workflow.

To create an AWS CloudFormation stack that creates an AWS Transfer Family workflow and server

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. Save the following code to a file.

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Resources:
  SFTPServer:
    Type: 'AWS::Transfer::Server'
    Properties:
      WorkflowDetails:
        OnUpload:
          - ExecutionRole: workflow-execution-role-arn
            WorkflowId: !GetAtt
              - TransferWorkflow
              - WorkflowId
  TransferWorkflow:
    Type: AWS::Transfer::Workflow
    Properties:
      Description: Transfer Family Workflows Blog
      Steps:
        - Type: COPY
          CopyStepDetails:
            Name: copyToUserKey
            DestinationFileLocation:
              S3FileLocation:
                Bucket: archived-records
                Key: ${transfer:UserName}/
                OverwriteExisting: 'TRUE'
        - Type: TAG
          TagStepDetails:
            Name: tagFileForArchive
            Tags:
              - Key: Archive
                Value: yes
        - Type: CUSTOM
          CustomStepDetails:
            Name: transferExtract
            Target: arn:aws:lambda:region:account-id:function:function-name
            TimeoutSeconds: 60
        - Type: DELETE
          DeleteStepDetails:
            Name: DeleteInputFile
            SourceFileLocation: '${original.file}'
      Tags:
        - Key: Name

```

Value: TransferFamilyWorkflows

JSON

```

{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "SFTPServer": {
      "Type": "AWS::Transfer::Server",
      "Properties": {
        "WorkflowDetails": {
          "OnUpload": [
            {
              "ExecutionRole": "workflow-execution-role-arn",
              "WorkflowId": {
                "Fn::GetAtt": [
                  "TransferWorkflow",
                  "WorkflowId"
                ]
              }
            ]
          ]
        }
      }
    },
    "TransferWorkflow": {
      "Type": "AWS::Transfer::Workflow",
      "Properties": {
        "Description": "Transfer Family Workflows Blog",
        "Steps": [
          {
            "Type": "COPY",
            "CopyStepDetails": {
              "Name": "copyToUserKey",
              "DestinationFileLocation": {
                "S3FileLocation": {
                  "Bucket": "archived-records",
                  "Key": "${transfer:UserName}/"
                }
              },
              "OverwriteExisting": "TRUE"
            }
          ]
        }
      }
    }
  }
}

```

```
    {
      "Type": "TAG",
      "TagStepDetails": {
        "Name": "tagFileForArchive",
        "Tags": [
          {
            "Key": "Archive",
            "Value": "yes"
          }
        ]
      }
    },
    {
      "Type": "CUSTOM",
      "CustomStepDetails": {
        "Name": "transferExtract",
        "Target": "arn:aws:lambda:region:account-  
id:function:function-name",
        "TimeoutSeconds": 60
      }
    },
    {
      "Type": "DELETE",
      "DeleteStepDetails": {
        "Name": "DeleteInputFile",
        "SourceFileLocation": "${original.file}"
      }
    }
  ],
  "Tags": [
    {
      "Key": "Name",
      "Value": "TransferFamilyWorkflows"
    }
  ]
}
}
```

3. Replace the following items with your actual values.

- Replace *workflow-execution-role-arn* with the ARN for an actual workflow execution role. For example, `arn:aws:transfer:us-east-2:111122223333:workflow/w-1234567890abcdef0`
 - Replace `arn:aws:lambda:region:account-id:function:function-name` with the ARN for your Lambda function. For example, `arn:aws:lambda:us-east-2:123456789012:function:example-lambda-idp`.
4. Follow the instructions for deploying an AWS CloudFormation stack from an existing template in [Selecting a stack template](#) in the *AWS CloudFormation User Guide*.

After the stack has been deployed, you can view details about it in the **Outputs** tab in the CloudFormation console. The template creates a new AWS Transfer Family SFTP server that uses service-managed users, and a new workflow, and associates the workflow with the new server.

Remove a workflow from a Transfer Family server

If you have associated a workflow with a Transfer Family server, and you now want to remove that association, you can do so by using the console or programmatically.

Console

To remove a workflow from a Transfer Family server

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Servers**.
3. Choose the identifier for the server in the **Server ID** column.
4. On the details page for the server, scroll down to the **Additional details** section, and then choose **Edit**.
5. On the **Edit additional details** page, in the **Managed workflows** section, clear the information for all settings:
 - Select the dash (-) from the list of workflows for the **Workflow for complete file uploads**.
 - If not already cleared, select the dash (-) from the list of workflows for the **Workflow for partial file uploads**.
 - Select the dash (-) from the list of roles for the **Managed workflows execution role**.

If you don't see the dash, scroll up until you see it, as it is the first value in each menu.

The screen should look like the following.

The screenshot shows the 'Managed workflows' configuration page in the AWS Transfer Family console. It is titled 'Managed workflows Info'. There are three main sections:

- Workflow for complete file uploads:** Select the workflow that AWS Transfer Family should run on all files that are uploaded in full via this server. It features a dropdown menu with 'Select a workflow', a refresh button, and a 'Create a new Workflow' button with an external link icon.
- Workflow for partial file uploads:** Select the workflow that AWS Transfer Family should run on all files that are only partially uploaded via this server. It also features a dropdown menu with 'Select a workflow', a refresh button, and a 'Create a new Workflow' button with an external link icon.
- Managed workflows execution role:** Select the role that AWS Transfer Family should assume when executing a workflow. It features a dropdown menu with a dash '-' and a refresh button.

6. Scroll down and choose **Save** to save your changes.

CLI

You use the `update-server` (or `UpdateServer` for API) call, and provide empty arguments for the `OnUpload` and `OnPartialUpload` parameters.

From the AWS CLI, run the following command:

```
aws transfer update-server --server-id your-server-id --workflow-details
'{"OnPartialUpload": [], "OnUpload": []}'
```

Replace *your-server-id* with the ID for your server. For example, if your server ID is, `s-01234567890abcdef`, the command is as follows:

```
aws transfer update-server --server-id s-01234567890abcdef --workflow-details
'{"OnPartialUpload": [], "OnUpload": []}'
```

Managed workflows restrictions and limitations

Restrictions

The following restrictions currently apply to post-upload processing workflows for AWS Transfer Family.

- Cross-account and cross-Region AWS Lambda functions are not supported. You can, however, copy across accounts, provided that your AWS Identity and Access Management (IAM) policies are correctly configured.
- For all workflow steps, any Amazon S3 buckets accessed by the workflow must be in the same region as the workflow itself.
- For a decryption step, the decryption destination must match the source for Region and backing store (for example, if the file to be decrypted is stored in Amazon S3, then the specified destination must also be in Amazon S3).
- Only asynchronous custom steps are supported.
- Custom step timeouts are approximate. That is, it might take slightly longer to time out than specified. Additionally, the workflow is dependent upon the Lambda function. Therefore, if the function is delayed during execution, the workflow is not aware of the delay.
- If you exceed your throttling limit, Transfer Family doesn't add workflow operations to the queue.
- Workflows are not initiated for files that have a size of 0. Files with a size greater than 0 do initiate the associated workflow.
- You can attach a file-processing workflow to a Transfer Family server that uses the AS2 protocol; however, AS2 messages don't execute workflows attached to the server.

Limitations

Additionally, the following functional limits apply to workflows for Transfer Family:

- The number of workflows per Region, per account, is limited to 10.
- The maximum timeout for custom steps is 30 minutes.
- The maximum number of steps in a workflow is 8.
- The maximum number of tags per workflow is 50.
- The maximum number of concurrent executions that contain a decrypt step is 250 per workflow.
- You can store a maximum of 3 PGP private keys, per Transfer Family server, per user.
- The maximum size for a decrypted file is 10 GB.

- We throttle the new execution rate using a [token bucket](#) system with a burst capacity of 100 and a refill rate of 1.
- Anytime you remove a workflow from a server and replace it with a new one, or update server configuration (which impacts a workflow's execution role), you must wait approximately 10 minutes before executing the new workflow. The Transfer Family server caches the workflow details, and it takes 10 minutes for the server to refresh its cache.

Additionally, you must log out of any active SFTP sessions, and then log back in after the 10-minute waiting period to see the changes.

Managing servers

In this section, you can find information on how to view a list of your servers, how to view your server details, how to edit your server details, and how to change the host key for your SFTP-enabled server.

Topics

- [View a list of servers](#)
- [Delete a server](#)
- [View SFTP, FTPS, and FTP server details](#)
- [View AS2 server details](#)
- [Edit server details](#)
- [Manage host keys for your SFTP-enabled server](#)
- [Monitoring usage in the console](#)

View a list of servers

On the AWS Transfer Family console, you can find a list of all your servers that are located in the AWS Region that you chose.

To find a list of your servers that exist in an AWS Region

- Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.

If you have one or more servers in the current AWS Region, the console opens to show a list of your servers. If you don't see a list of servers, make sure that you are in the correct Region. You can also choose **Servers** from the navigation pane.

For more information about viewing your server details, see [View SFTP, FTPS, and FTP server details](#).

Delete a server

This procedure explains how to delete a Transfer Family server by using the AWS Transfer Family console or AWS CLI.

⚠ Important

You are billed, for each of the protocols enabled to access your endpoint, until you delete the server.

⚠ Warning

Deleting a server results in all its users being deleted. Data in the bucket that was accessed by using the server is not deleted, and remains accessible to AWS users that have privileges to those Amazon S3 buckets.

Console

To delete a server by using the console

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Servers**.
3. Select the check box of the server that you want to delete.
4. For **Actions**, choose **Delete**.
5. In the confirmation dialog box that appears, enter the word **delete**, and then choose **Delete** to confirm that you want to delete the server.

The server is deleted from the **Servers** page and you are no longer billed for it.

AWS CLI

To delete a server by using the CLI

1. (Optional) Run the following command to view the details for the server that you want to delete permanently.

```
aws transfer describe-server --server-id your-server-id
```

This `describe-server` command returns all of the details for your server.

2. Run the following command to delete the server.

```
aws transfer delete-server --server-id your-server-id
```

If successful, the command deletes the server and does not return any information.

View SFTP, FTPS, and FTP server details

You can find a list of details and properties for an individual AWS Transfer Family server. Server properties include protocols, identity provider, status, endpoint type, custom hostname, endpoint, users, logging role, server host key, and tags.

To view server details

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the navigation pane, choose **Servers**.
3. Choose the identifier in the **Server ID** column to see the **Server details** page, shown following.

You can change the server's properties on this page by choosing **Edit**. For more information about editing server details, see [Edit server details](#). The details page for AS2 servers differs slightly. For AS2 servers, see [View AS2 server details](#).

Protocols Edit	Identity provider Edit
Protocols over which clients can connect to your server's endpoint <ul style="list-style-type: none">• SFTP	Identity provider type Info Custom - AWS Lambda AWS Lambda function test-UserAuthenticationLambda ↗

Note

The server host key **Description** and **Date imported** values are new as of September 2022. These values were introduced to support the multiple host keys feature.

This feature required migration of any single host keys that were in use before the introduction of multiple host keys.

The **Date imported** value for a migrated server host key is set to the last modified date for the server. That is, the date that you see for your migrated host key corresponds

to the date that you last modified the server in any way, before the server host key migration.

The only key that was migrated is your oldest or only server host key. Any additional keys have their actual date from when you imported them. Additionally, the migrated key has a description that makes it easy to identify it as having been migrated.

The migration occurred between September 2 and September 13. The actual migration date within this range depends on the Region of your server.

Additional details Edit

<p>Log group /aws/transfer/s- [redacted] </p> <p>Logging role Info AWSTransferLoggingAccess </p> <p>Server host key Info SHA256: [redacted]</p> <p>Security Policy Info TransferSecurityPolicy-2020-06</p>	<p>Domain Amazon S3</p> <p>Workflow for complete uploads w-[redacted]</p> <p>Workflow for partial uploads -</p> <p>Managed workflows execution role transfer-workflows-[redacted] </p>	<p>Login display banner View the display message</p> <p>SetStat option Ignore</p> <p>TLS session resumption -</p> <p>Passive IP -</p>
--	--	---

View AS2 server details

You can find a list of details and properties for an individual AWS Transfer Family server. Server properties include protocols, status, and more. For AS2 servers, you can also view the AS2 asynchronous MDN egress IP addresses.

Protocols Edit

Protocols over which clients can connect to your server's endpoint

- AS2

Identity provider Edit

AS2 Auth
Basic authentication is not supported for AS2. Access can be controlled through VPC security groups.

Each AS2 server is assigned three static IP addresses. Use these IP addresses for sending asynchronous MDNs to your trading partners over AS2.

AS2 asynchronous MDN egress IP details

Below are the service managed static IP addresses used for sending your asynchronous MDNs to trading partners over AS2

- IP address: [redacted]
- IP address: [redacted]
- IP address: [redacted]

The bottom portion of the AS2 server details page contains details for any attached workflow and monitoring and tagging information.

Workflows

[Edit](#)

Workflow for complete uploads w- [redacted] 0	Workflow for partial uploads -	Managed workflows execution role [redacted] ↗
--	-----------------------------------	--

Monitoring

1h 3h 12h 1d 3d 1w UTC timezone [↻](#) [⌵](#)

BytesIn	BytesOut	FilesIn	FilesOut
No data available. Try adjusting the dashboard time range.	No data available. Try adjusting the dashboard time range.	No data available. Try adjusting the dashboard time range.	No data available. Try adjusting the dashboard time range.

AS2 Monitoring

1h 3h 12h 1d 3d 1w UTC timezone [↻](#) [⌵](#)

InboundMessage	InboundMessage	[redacted] sage	[redacted] sage
InboundMessage	No data available. Try adjusting the dashboard time range.	sage	No data available. Try adjusting the dashboard time range.

Edit server details

After you create an AWS Transfer Family server, you can edit the server configuration.

Topics

- [Edit the file transfer protocols](#)
- [Edit custom identity provider parameters](#)
- [Edit the server endpoint](#)
- [Edit your logging configuration](#)
- [Edit the security policy](#)
- [Change the managed workflow for your server](#)
- [Change the display banners for your server](#)
- [Put your server online or offline](#)

To edit a server's configuration

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Servers**.
3. Choose the identifier in the **Server ID** column to see the **Server details** page, shown following.

You can change the server's properties on this page by choosing **Edit**:


- To change the protocols, see [Edit the file transfer protocols](#).
- For the identity provider, note that you can't change a server's identity provider type after you create the server. To change the identity provider, delete the server and create a new one with the identity provider that you want.

Note

If your server uses a custom identity provider, you can edit some properties. For details, see [Edit custom identity provider parameters](#).

- To change the endpoint type or custom hostname, see [Edit the server endpoint](#).
- To add an agreement, you need to first add AS2 as a protocol to your server. For details, see [Edit the file transfer protocols](#).

- To manage host keys for your server, see [Manage host keys for your SFTP-enabled server](#).
- Under **Additional details**, you can edit the following information:
 - To change the logging role, see [Edit your logging configuration](#).
 - To change the security policy, see [Edit the security policy](#).
 - To change the server host key, see [Manage host keys for your SFTP-enabled server](#).
 - To change the managed workflow for your server, see [Change the managed workflow for your server](#).
 - To edit the display banners for your server, see [Change the display banners for your server](#).
- Under Additional configuration, you can edit the following information:
 - **SetStat option:** enable this option to ignore the error that is generated when a client attempts to use SETSTAT on a file you are uploading to an Amazon S3 bucket. For additional details, see the SetStatOption documentation in the [ProtocolDetails](#) topic.
 - **TLS session resumption:** provides a mechanism to resume or share a negotiated secret key between the control and data connection for an FTPS session. For additional details, see the TlsSessionResumptionMode documentation in the [ProtocolDetails](#) topic.
 - **Passive IP:** indicates passive mode, for FTP and FTPS protocols. Enter a single IPv4 address, such as the public IP address of a firewall, router, or load balancer. For additional details, see the PassiveIp documentation in the [ProtocolDetails](#) topic.
- To start or stop your server, see [Put your server online or offline](#).
- To delete a server, see [Delete a server](#).
- To edit a user's properties, see [Managing access controls](#).

Protocols Edit	Identity provider Edit
Protocols over which clients can connect to your server's endpoint <ul style="list-style-type: none"> • SFTP 	Identity provider type Info Custom - AWS Lambda AWS Lambda function test-UserAuthenticationLambda 

Note

The server host key **Description** and **Date imported** values are new as of September 2022. These values were introduced to support the multiple host keys feature.

This feature required migration of any single host keys that were in use before the introduction of multiple host keys.

The **Date imported** value for a migrated server host key is set to the last modified date for the server. That is, the date that you see for your migrated host key corresponds to the date that you last modified the server in any way, before the server host key migration.

The only key that was migrated is your oldest or only server host key. Any additional keys have their actual date from when you imported them. Additionally, the migrated key has a description that makes it easy to identify it as having been migrated.

The migration occurred between September 2 and September 13. The actual migration date within this range depends on the Region of your server.

Additional details

Edit

<p>Log group /aws/transfer/s- [redacted]</p> <p>Logging role Info AWSTransferLoggingAccess</p> <p>Server host key Info SHA256: [redacted]</p> <p>Security Policy Info TransferSecurityPolicy-2020-06</p>	<p>Domain Amazon S3</p> <p>Workflow for complete uploads w-[redacted]</p> <p>Workflow for partial uploads -</p> <p>Managed workflows execution role transfer-workflows [redacted]</p>	<p>Login display banner View the display message</p> <p>SetStat option Ignore</p> <p>TLS session resumption -</p> <p>Passive IP -</p>
--	---	---

Edit the file transfer protocols

On the AWS Transfer Family console, you can edit the file transfer protocol. The file transfer protocol connects the client to your server's endpoint.

To edit the protocols

1. On the **Server details** page, choose **Edit** next to **Protocols**.
2. On the **Edit protocols** page, select or clear the protocol check box or check boxes to add or remove the following file transfer protocols:

- Secure Shell (SSH) File Transfer Protocol (SFTP) – file transfer over SSH

For more information about SFTP, see [Create an SFTP-enabled server](#).

- File Transfer Protocol Secure (FTPS) – file transfer with TLS encryption

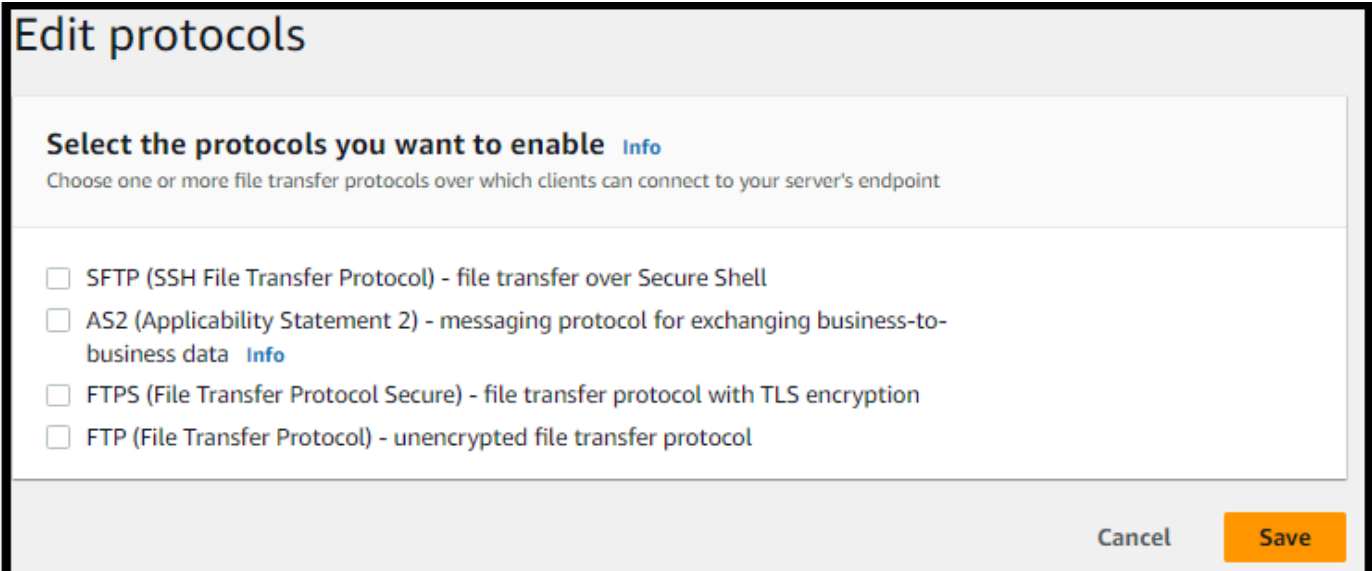
For more information about FTP, see [Create an FTPS-enabled server](#).

- File Transfer Protocol (FTP) – unencrypted file transfer

For more information about FTPS, see [Create an FTP-enabled server](#).

Note

If you have an existing server enabled only for SFTP, and you want to add FTPS and FTP, you must ensure that you have the right identity provider and endpoint type settings that are compatible with FTPS and FTP.



Edit protocols

Select the protocols you want to enable [Info](#)

Choose one or more file transfer protocols over which clients can connect to your server's endpoint

- SFTP (SSH File Transfer Protocol) - file transfer over Secure Shell
- AS2 (Applicability Statement 2) - messaging protocol for exchanging business-to-business data [Info](#)
- FTPS (File Transfer Protocol Secure) - file transfer protocol with TLS encryption
- FTP (File Transfer Protocol) - unencrypted file transfer protocol

Cancel Save

If you select **FTPS**, you must choose a certificate stored in AWS Certificate Manager (ACM) which will be used to identify your server when clients connect to it over FTPS.


To request a new public certificate, see [Request a public certificate](#) in the *AWS Certificate Manager User Guide*.

To import an existing certificate into ACM, see [Importing certificates into ACM](#) in the *AWS Certificate Manager User Guide*.

To request a private certificate to use FTPS through private IP addresses, see [Requesting a private certificate](#) in the *AWS Certificate Manager User Guide*.

Certificates with the following cryptographic algorithms and key sizes are supported:

- 2048-bit RSA (RSA_2048)
- 4096-bit RSA (RSA_4096)
- Elliptic Prime Curve 256 bit (EC_prime256v1)
- Elliptic Prime Curve 384 bit (EC_secp384r1)
- Elliptic Prime Curve 521 bit (EC_secp521r1)

 **Note**

The certificate must be a valid SSL/TLS X.509 version 3 certificate with FQDN or IP address specified and contain information about the issuer.

3. Choose **Save**. You are returned to the **Server details** page.

Edit custom identity provider parameters

On the AWS Transfer Family console, for custom identity providers, you can change some of the settings, depending on whether you are using a Lambda function or an API Gateway. In either case, if your server uses the SFTP protocol, you can edit your authentication method.

- If you are using a Lambda as your identity provider, you can change the underlying Lambda function.

Transfer Family > Servers > s- [redacted] > Edit identity provider

Edit identity provider

Identity Provider for SFTP, FTPS, or FTP

Identity provider type
An identity provider manages user access for authentication and authorization

Service managed
Create and manage users within the service

AWS Directory Service [Info](#)
Enable users in AWS Managed AD or use your own self-managed AD in your on-premises environment or in AWS

Custom Identity Provider [Info](#)
Manage users by integrating an identity provider of your choice

Use AWS Lambda to connect your identity provider [Info](#)
Invoke an AWS Lambda function to call your identity provider's API for user authentication and authorization

Use Amazon API Gateway to connect your identity provider [Info](#)
Use a RESTful API method to call your identity provider's API for user authentication and authorization

AWS Lambda function

[redacted] ▼ G

Authentication methods
Choose which authentication methods are required for users to connect to your server

Password OR public key

Password ONLY

Public Key ONLY

Password AND public key

i Either a valid password or valid private key will be required during user authentication

Cancel Save

- If you are using an API Gateway as your identity provider, you can update the Gateway URL or the invocation role, or both.

Transfer Family > Servers > s-[redacted] > Edit identity provider

Edit identity provider

Identity Provider for SFTP, FTPS, or FTP

Identity provider type
An identity provider manages user access for authentication and authorization

Service managed
Create and manage users within the service

AWS Directory Service [Info](#)
Enable users in AWS Managed AD or use your own self-managed AD in your on-premises environment or in AWS

Custom Identity Provider [Info](#)
Manage users by integrating an identity provider of your choice

Use AWS Lambda to connect your identity provider [Info](#)
Invoke an AWS Lambda function to call your identity provider's API for user authentication and authorization

Use Amazon API Gateway to connect your identity provider [Info](#)
Use a RESTful API method to call your identity provider's API for user authentication and authorization

Provide an Amazon API Gateway URL

Invocation role
IAM role for the service to invoke your Amazon API Gateway URL

[Refresh](#)

Authentication methods
Choose which authentication methods are required for users to connect to your server

Password OR public key

Password ONLY

Public Key ONLY

Password AND public key

[Info](#) Either a valid password or valid private key will be required during user authentication

Cancel Save

Edit the server endpoint

On the AWS Transfer Family console, you can modify the server endpoint type and custom hostname. Additionally, for VPC endpoints, you can edit the availability zone information.

To edit the server endpoint details

1. On the **Server details** page, choose **Edit** next to **Endpoint details**.
2. Before you can edit the **Endpoint type**, you must first stop the server. Then, on the **Edit endpoint configuration** page, for **Endpoint type**, you can choose either of the following values:
 - **Public** – This option makes your server accessible over the internet.
 - **VPC** – This option makes your server accessible in your virtual private cloud (VPC). For information about VPC, see [Create a server in a virtual private cloud](#).
3. For **Custom hostname**, choose one of the following:
 - **None** – If you don't want to use a custom domain, choose **None**.

You get a server hostname provided by AWS Transfer Family. The server hostname takes the form `serverId.server.transfer.regionId.amazonaws.com`.

- **Amazon Route 53 DNS alias** – To use a DNS alias automatically created for you in Route 53, choose this option.
- **Other DNS** – To use a hostname that you already own in an external DNS service choose **Other DNS**.

Choosing **Amazon Route 53 DNS alias** or **Other DNS** specifies the name resolution method to associate with your server's endpoint.

For example, your custom domain might be `sftp.inbox.example.com`. A custom hostname uses a DNS name that you provide and that a DNS service can resolve. You can use Route 53 as your DNS resolver, or use your own DNS service provider. To learn how AWS Transfer Family uses Route 53 to route traffic from your custom domain to the server endpoint, see [Working with custom hostnames](#).

4. For VPC endpoints, you can change the information in the **Availability Zones** pane.
5. Choose **Save**. You are returned to the **Server details** page.

Edit your logging configuration

On the AWS Transfer Family console, you can change your logging configuration.

Note

If Transfer Family created a CloudWatch logging IAM role for you when you created a server, the IAM role is called `AWSTransferLoggingAccess`. You can use it for all your Transfer Family servers.

To edit your logging configuration

1. On the **Server details** page, choose **Edit** next to **Additional details**.
2. Based on your configuration, choose between a logging role, structured JSON logging, or both. For more information, see [Updating logging for a server](#).

Edit the security policy

This procedure explains how to change a Transfer Family server's security policy by using the AWS Transfer Family console or AWS CLI.

Note

If your endpoint is FIPS-enabled, you can't change the FIPS security policy to a non-FIPS security policy.

Console

To edit the security policy by using the console

1. On the **Server details** page, choose **Edit** next to **Additional details**.
2. In the **Cryptographic algorithm options** section, choose a security policy that contains the cryptographic algorithms enabled for use by your server.

For more information about security policies, see [Security policies for AWS Transfer Family servers](#).

3. Choose **Save**.

You are returned to the **Server details** page where you can see the updated security policy.

AWS CLI

To edit the security policy by using the CLI

1. Run the following command to view the current security policy that is attached to your server.

```
aws transfer describe-server --server-id your-server-id
```

This `describe-server` command returns all of the details for your server, including the following line:

```
"SecurityPolicyName": "TransferSecurityPolicy-2018-11"
```

In this case, the security policy for the server is `TransferSecurityPolicy-2018-11`.

2. Make sure to provide the exact name of the security policy to the command.
For example, run the following command to update the server to `TransferSecurityPolicy-2023-05`.

```
aws transfer update-server --server-id your-server-id --security-policy-name  
"TransferSecurityPolicy-2023-05"
```

Note

The names of the available security policies are listed in [Security policies for AWS Transfer Family servers](#).

If successful, the command returns the following code, and updates your server's security policy.

```
{  
  "ServerId": "your-server-id"  
}
```

Change the managed workflow for your server

On the AWS Transfer Family console, you can change the managed workflow associated with the server.

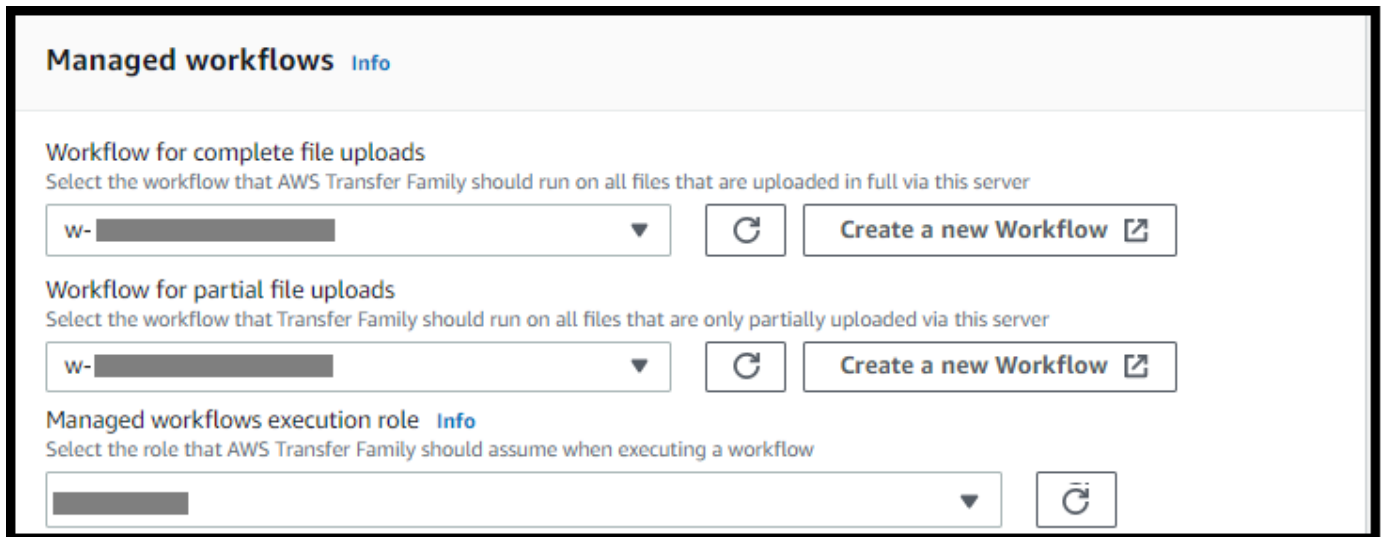
To change the managed workflow

1. On the **Server details** page, choose **Edit** next to **Additional details**.
2. On the **Edit additional details** page, in the **Managed workflows** section, select a workflow to be run on all uploads.

Note

If you do not already have a workflow, choose **Create a new workflow** to create one.

- a. Select the workflow ID to use.
- b. Choose an execution role. This is the role that Transfer Family assumes when executing the workflow's steps. For more information, see [IAM policies for workflows](#). Choose **Save**.



Managed workflows [Info](#)

Workflow for complete file uploads
Select the workflow that AWS Transfer Family should run on all files that are uploaded in full via this server

w- [redacted] ▼ [↗](#)

Workflow for partial file uploads
Select the workflow that Transfer Family should run on all files that are only partially uploaded via this server

w- [redacted] ▼ [↗](#)

Managed workflows execution role [Info](#)
Select the role that AWS Transfer Family should assume when executing a workflow

[redacted] ▼

3. Choose **Save**. You are returned to the **Server details** page.

Change the display banners for your server

On the AWS Transfer Family console, you can change the display banners associated with the server.

To change the display banners

1. On the **Server details** page, choose **Edit** next to **Additional details**.
2. On the **Edit additional details** page, in the **Display banners** section, enter text for the available display banners.
3. Choose **Save**. You are returned to the **Server details** page.

Put your server online or offline

On the AWS Transfer Family console, you can bring your server online or take it offline.

To bring your server online

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the navigation pane, choose **Servers**.
3. Select the check box of the server that is offline.

4. For **Actions**, choose **Start**.

It can take a couple of minutes for a server to switch from offline to online.

Note

When you stop a server to take it offline, currently you are still accruing service charges for that server. To eliminate additional server-based charges, delete that server.

To take your server offline

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the navigation pane, choose **Servers**.
3. Select the check box of the server that is online.
4. For **Actions**, choose **Stop**.

While a server is starting up or shutting down, servers aren't available for file operations. The console doesn't show the starting and stopping states.

If you find the error condition `START_FAILED` or `STOP_FAILED`, contact AWS Support to help resolve your issues.

Manage host keys for your SFTP-enabled server

Important

If you aren't planning to migrate existing users from an existing SFTP-enabled server to a new SFTP-enabled server, ignore this section.

Accidentally changing a server's host key can be disruptive. Depending on how your SFTP client is configured, it can fail immediately, with the message that no trusted host key exists, or present threatening prompts. If there are scripts for automating connections, they most likely would fail as well.

By default, AWS Transfer Family provides a host key for your SFTP-enabled server. You can replace the default host key with a host key from another server. Do so only if you plan to move existing users from an existing SFTP-enabled server to your new SFTP-enabled server.

To prevent your users from being prompted to verify the authenticity of your SFTP-enabled server again, import the host key for your on-premises server to the SFTP-enabled server. Doing this also prevents your users from getting a warning about a potential man-in-the-middle attack.

You can also rotate host keys periodically, as an additional security measure.

Note

Although the Transfer Family console allows you to specify and add server host keys for all servers, these keys are only useful for servers that use the SFTP protocol.

Topics

- [Add an additional server host key](#)
- [Delete a server host key](#)
- [Rotate the server host keys](#)
- [Additional server host key information](#)

Add an additional server host key

On the AWS Transfer Family console, you can add additional server host keys. Adding additional host keys of differing formats can be useful for identifying a server when clients connect to it, as well as improving your security profile. For example, if your original key is an RSA key, you could add an additional ECDSA key.

Note

The SFTP client connects using the first public key it has that can match one of the active server keys.

To add an additional server host key

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.

- In the left navigation pane, choose **Servers**, and then choose a server that uses the SFTP protocol.
- On the server details page, scroll down to the **Server host keys** section.

Server host keys (2) Actions ▾ Add host key

< 1 >

<input type="checkbox"/>	Name ↗	Host key ID ▾	Fingerprint Info	Description ▾	Key type ▾	Date imported ▾
<input type="checkbox"/>	Default host key	hostkey-██████████	SHA256:██████████	Default	ssh-rsa	2023-06-30
<input type="checkbox"/>	ecdsa-521	hostkey-██████████	SHA256:██████████	ECDSA host key	ecdsa-sha2-nistp521	2024-06-13

- Choose **Add host key**.

The **Add server host key** page displays.

- In the **Server Host Key** section, enter an RSA, ECDSA, or ED25519 private key that is used to identify your server when clients connect to it over the SFTP-enabled server.

Note

When you create a server host key, make sure to specify `-N ""` (no passphrase). See [Creating SSH keys on macOS, Linux, or Unix](#) for details on how to generate key pairs.

- (Optional) Add a description to differentiate among multiple server host keys. You can also add tags for your key.
- Choose **Add key**. You are returned to the **Server details** page.

To add a host key by using the AWS Command Line Interface (AWS CLI), use the [the section called "ImportHostKey"](#) API operation and provide the new host key. If you create a new SFTP-enabled server, you provide your host key as a parameter in the [the section called "CreateServer"](#) API operation. You can also use the AWS CLI to update the description for an existing host key.

The following example `import-host-key` AWS CLI command imports a host key for the specified SFTP-enabled server.

```
aws transfer import-host-key --description key-description --server-id your-server-id
--host-key-body file://my-host-key
```

Delete a server host key

On the AWS Transfer Family console, you can delete a server host key.

To delete a server host key

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. In the left navigation pane, choose **Servers**, and then choose a server that uses the SFTP protocol.
3. On the server details page, scroll down to the **Server host keys** section.

Server host keys (2)							
Find resources							
	Name	Host key ID	Fingerprint	Description	Key type	Date imported	
<input type="checkbox"/>	Default host key	hostkey- [REDACTED]	SHA256: [REDACTED]	Default	ssh-rsa	2023-06-30	
<input type="checkbox"/>	ecdsa-521	hostkey- [REDACTED]	SHA256: [REDACTED]	ECDSA host key	ecdsa-sha2-nistp521	2024-06-13	

4. In the **Server Host Keys** section, select a key, and then under **Actions**, choose **Delete**.
5. In the confirmation dialog box that appears, enter the word **delete**, and then choose **Delete** to confirm that you want to delete the host key.

The host key is deleted from the **Servers** page.

To delete the host key by using the AWS CLI, use the [the section called "DeleteHostKey"](#) API operation and provide the server ID and host key ID.

The following example `delete-host-key` AWS CLI command deletes a host key for the specified SFTP-enabled server.

```
aws transfer delete-host-key --server-id your-server-id --host-key-id your-host-key-id
```

Rotate the server host keys

Periodically, you can rotate your server host key.

How the client chooses a server host key

The way that Transfer Family chooses which server key to apply depends on conditions for the SFTP client, as explained here. The assumption is that there is one older key and one newer key.

- An SFTP client has no prior public host key for the server. The first time the client connects to the server, either of the following occurs:
 - The client fails the connection, if it is configured to do so.

- Or, the client chooses the first key that matches the possible available algorithms and asks the user if that key can be trusted. If so, the client auto-updates the `known_hosts` file (or whatever local configuration file or resource the client uses to record trust decisions) and enters that key.
- An SFTP client has an older key in its `known_hosts` file. The client prefers to use this key, even if a newer key exists, either for this key's algorithm or another algorithm. This is because the client has a higher level of trust for the key that is in its `known_hosts` file.
- An SFTP client has the new key (in any of the available algorithms) in its `known_hosts` keys file. The client ignores older keys because they are not trusted and uses the new key.
- An SFTP client has both keys in its `known_hosts` file. The client chooses the first key by index that matches the list of available keys offered by the server.

Transfer Family prefers that the SFTP client has all of the keys in its `known_hosts` file, since this allows the most flexibility when connecting to a Transfer Family server. Key rotation is based on the fact that multiple entries can exist in the `known_hosts` file for the same Transfer Family server.

Rotate the server host key procedure

As an example, assume that you have added the following set of server host keys to your Transfer Family server.

Server host keys

Host key type	Date added to the server
RSA	April 1, 2020
ECDSA	February 1, 2020
ED25519	December 1, 2019
RSA	October 1, 2019
ECDSA	June 1, 2019
ED25519	March 1, 2019

To rotate the server host key

1. Add a new server host key. This procedure is described in [Add an additional server host key](#).
2. Delete one or more of the host keys of the same type that you had added previously. This procedure is described in [Delete a server host key](#).
3. All keys are visible, and can be active, subject to the behavior described previously in [How the client chooses a server host key](#).

Additional server host key information

You can select a host key to display details for that key.

The screenshot shows the AWS console interface for a specific host key. The breadcrumb navigation is: Transfer Family > Servers > s-3fe215d89f074ed2a > Hostkey: hostkey-a6fed60bcb704ea9b. The main heading is "hostkey-a6fed60bcb704ea9b" with "Delete" and "Edit" buttons. Below is the "Host key configuration" section with the following details:

Name	ecdsa-521	Key type	ecdsa-sha2-nistp521
Fingerprint	SHA256: [REDACTED]	Date imported	Thu, 13 Jun 2024 17:08:59 GMT
Description	ECDSA host key	Amazon Resource Name (ARN)	arn:aws:transfer:us-east-1:[REDACTED]:host-key/s-[REDACTED]/hostkey-a-[REDACTED]

You can delete a host key, or edit its description from the **Actions** menu on the Server details screen. Select the host key, then choose the appropriate action from the menu.

The screenshot shows the "Server host keys (2)" page in the AWS console. It includes a search bar with "Find resources" and an "Add host key" button. A table lists the host keys, and an "Actions" menu is highlighted with a red box, showing options for "Edit" and "Delete".

	Name	Host key ID	Fingerprint	Description	Key type	Date imported
<input type="checkbox"/>	Default host key	hostkey-[REDACTED]	SHA256:[REDACTED]	Default	ssh-rsa	2023-06-30
<input checked="" type="checkbox"/>	ecdsa-521	hostkey-[REDACTED]	SHA256:[REDACTED]	ECDSA host key	ecdsa-sha2-nistp521	2024-06-13

Monitoring usage in the console

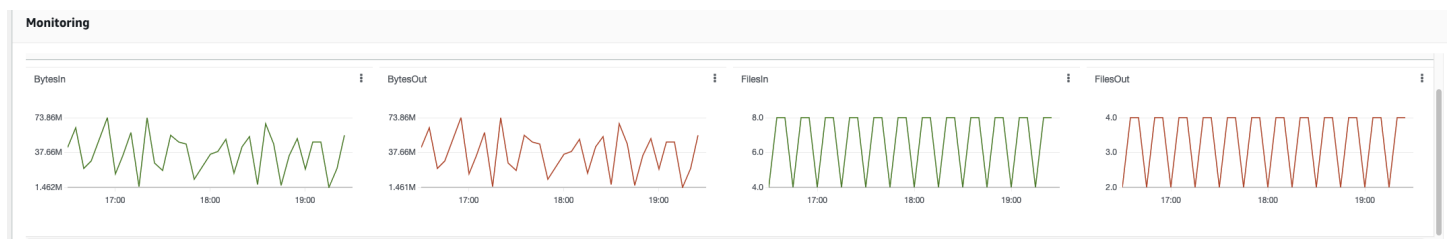
You can get information about your server's metrics on its **Server details** page. This provides you with a single place to monitor your file-transfers workloads. You can track how many files you have exchanged with your partners and closely track their usage using a centralized dashboard.

For details, see [View SFTP, FTPS, and FTP server details](#). The following table describes the metrics available for Transfer Family.

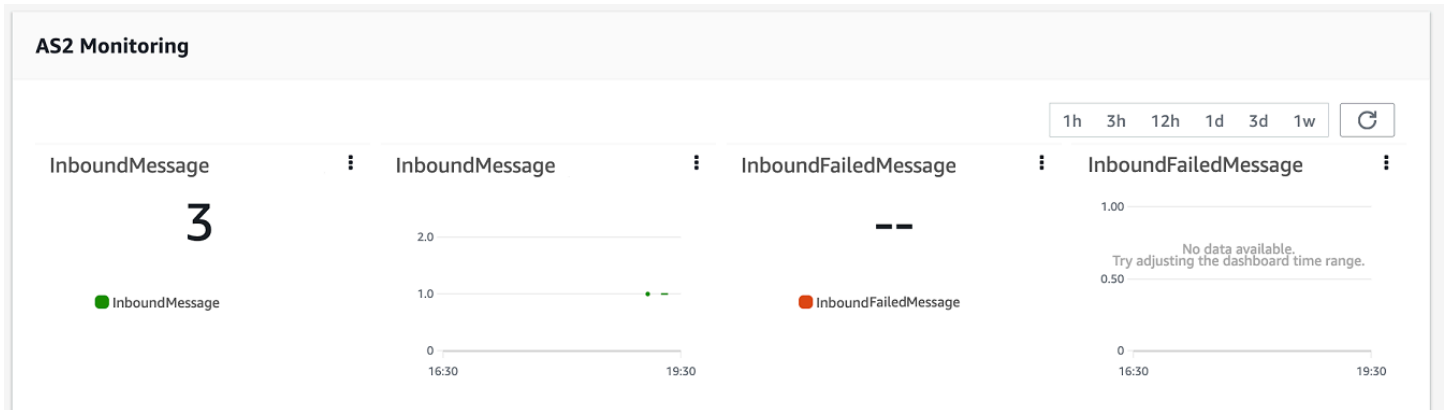
Namespace	Metric	Description
AWS/Transfer	BytesIn	The total number of bytes transferred into the server. Units: Count Period: 5 minutes
	BytesOut	The total number of bytes transferred out of the server. Unit: Count Period: 5 minutes
	FilesIn	The total number of files transferred into the server. For servers using the AS2 protocol, this metric represents the number of messages received. Units: Count Period: 5 minutes
	FilesOut	The total number of files transferred out of the server. Units: Count Period: 5 minutes
	InboundMessage	The total number of AS2 messages successfully received from a trading partner. Units: Count Period: 5 minutes
	InboundFailedMessage	The total number of AS2 messages that were unsuccessfully received from a trading partner. That is, a trading

Namespace	Metric	Description
		<p>partner sent a message, but the Transfer Family server was not able to successfully process it.</p> <p>Units: Count</p> <p>Period: 5 minutes</p>
	OnUploadExecutionsStarted	<p>The total number of workflow executions started on the server.</p> <p>Units: Count</p> <p>Period: 1 minute</p>
	OnUploadExecutionsSuccess	<p>The total number of successful workflow executions on the server.</p> <p>Units: Count</p> <p>Period: 1 minute</p>
	OnUploadExecutionsFailed	<p>The total number of unsuccessful workflow executions on the server.</p> <p>Units: Count</p> <p>Period: 1 minute</p>

The **Monitoring** section contains four, individual graphs. These graphs show the bytes in, bytes out, files in, and files out.



For servers that have the AS2 protocol enabled, there is an **AS2 Monitoring** section below the **Monitoring** information. This section contains details for the number of inbound messages, both successful and failed.



To open the selected graph in its own window, choose the expand icon

().

You can also click a graph's vertical ellipsis icon

()

to open a dropdown menu with the following items:

- **Enlarge** – Opens the selected graph in its own window.
- **Refresh** – Reloads the graph with the most recent data.
- **View in metrics** – Opens the corresponding metrics details in Amazon CloudWatch.
- **View logs** – Opens the corresponding log group in CloudWatch.

Managing access controls

You can control a user's access to AWS Transfer Family resources by using an AWS Identity and Access Management (IAM) policy. An IAM policy is a statement, typically in JSON format, that allows a certain level of access to a resource. You use an IAM policy to define what file operations that you want to allow your users to perform and not perform. You can also use an IAM policy to define what Amazon S3 bucket or buckets that you want to give your users access to. To specify these policies for users, you create an IAM role for AWS Transfer Family that has the IAM policy and trust relationship associated with it.

Each user is assigned an IAM role. The type of IAM role that AWS Transfer Family uses is called a *service role*. When a user logs in to your server, AWS Transfer Family assumes the IAM role mapped to the user. To learn about creating an IAM role that provides a user access to an Amazon S3 bucket, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

You can grant write-only access to Amazon S3 objects by using certain permissions within an IAM policy. For details, see [Grant ability to only write and list files](#).

The AWS Storage Blog contains a post detailing how to set up least privilege access. For details, see [Implementing least privilege access in an AWS Transfer Family workflow](#).

Note

If your Amazon S3 bucket is encrypted using AWS Key Management Service (AWS KMS), you must specify additional permissions in your policy. For details, see [Data encryption in Amazon S3](#). Additionally, you can see more information about [session policies](#) in the *IAM User Guide*.

Topics

- [Allowing read and write access to an Amazon S3 bucket](#)
- [Creating a session policy for an Amazon S3 bucket](#)
- [Preventing users from running mkdir in an S3 bucket](#)

Allowing read and write access to an Amazon S3 bucket

This section describes how to create an IAM policy that allows read and write access to a specific Amazon S3 bucket. Assigning an IAM role that has this IAM policy to your user gives that user read/write access to the specified Amazon S3 bucket.

The following policy provides programmatic read, write, and tagging access to an Amazon S3 bucket. The `GetObjectACL` and `PutObjectACL` statements are only required if you need to enable Cross Account Access. That is, your Transfer Family server needs to access a bucket in a different account.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadWriteS3",
      "Action": [
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetObjectTagging",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionTagging",
        "s3:GetObjectACL",
        "s3:PutObjectACL"
      ],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"]
    }
  ]
}
```


The `ListBucket` action requires permission to the bucket itself. The `PUT`, `GET`, and `DELETE` actions require object permissions. Because these are different resources, they are specified using different Amazon Resource Names (ARNs).

To further restrict your users' access to only the home prefix of the specified Amazon S3 bucket, see [Creating a session policy for an Amazon S3 bucket](#).

Creating a session policy for an Amazon S3 bucket

A *session policy* is an AWS Identity and Access Management (IAM) policy that restricts users to certain portions of an Amazon S3 bucket. It does so by evaluating access in real time.

Note

Session policies are only used with Amazon S3. For Amazon EFS, you use POSIX file permissions to limit access.

You can use a session policy when you need to give the same access to a group of users to a particular portion of your Amazon S3 bucket. For example, a group of users might need access to only the home directory. That group of users share the same IAM role.

Note

The maximum length of a session policy is 2048 characters. For more details, see the [Policy request parameter](#) for the `CreateUser` action in the *API reference*.

To create a session policy, use the following policy variables in your IAM policy:

- `${transfer:HomeBucket}`
- `${transfer:HomeDirectory}`
- `${transfer:HomeFolder}`
- `${transfer:UserName}`

⚠ Important

You can't use the preceding variables in Managed Policies. Nor can you use them as policy variables in an IAM role definition. You create these variables in an IAM policy and supply them directly when setting up your user. Also, you can't use the `${aws:Username}` variable in this session policy. This variable refers to an IAM user name and not the username required by AWS Transfer Family.

The following code shows an example session policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListingOfUserFolder",
      "Action": [
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::${transfer:HomeBucket}"
      ],
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "${transfer:HomeFolder}/*",
            "${transfer:HomeFolder}"
          ]
        }
      }
    },
    {
      "Sid": "HomeDirObjectAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:GetObjectVersion",
        "s3:GetObjectACL",

```

```
        "s3:PutObjectACL"
      ],
      "Resource": "arn:aws:s3:::${transfer:HomeDirectory}/*"
    }
  ]
}
```

Note

The preceding policy example assumes that users have their home directories set to include a trailing slash, to signify that it is a directory. If, on the other hand, you set a user's `HomeDirectory` without the trailing slash, then you should include it as part of your policy.

In the previous example policy, note the use of the `transfer:HomeFolder`, `transfer:HomeBucket`, and `transfer:HomeDirectory` policy parameters. These parameters are set for the `HomeDirectory` that is configured for the user, as described in [HomeDirectory](#) and [Implementing your API Gateway method](#). These parameters have the following definitions:

- The `transfer:HomeBucket` parameter is replaced with the first component of `HomeDirectory`.
- The `transfer:HomeFolder` parameter is replaced with the remaining portions of the `HomeDirectory` parameter.
- The `transfer:HomeDirectory` parameter has the leading forward slash (/) removed so that it can be used as part of an S3 Amazon Resource Name (ARN) in a `Resource` statement.

Note

If you are using logical directories—that is, the user's `homeDirectoryType` is `LOGICAL`—these policy parameters (`HomeBucket`, `HomeDirectory`, and `HomeFolder`) are not supported.

For example, assume that the `HomeDirectory` parameter that is configured for the Transfer Family user is `/home/bob/amazon/stuff/`.

- `transfer:HomeBucket` is set to `/home`.
- `transfer:HomeFolder` is set to `/bob/amazon/stuff/`.
- `transfer:HomeDirectory` becomes `home/bob/amazon/stuff/`.

The first "Sid" allows the user to list all directories starting from `/home/bob/amazon/stuff/`.

The second "Sid" limits the user's put and get access to that same path, `/home/bob/amazon/stuff/`.

With the preceding policy in place, when a user logs in, they can access only objects in their home directory. At connection time, AWS Transfer Family replaces these variables with the appropriate values for the user. Doing this makes it easier to apply the same policy documents to multiple users. This approach reduces the overhead of IAM role and policy management for managing your users' access to your Amazon S3 bucket.

You can also use a session policy to customize access for each of your users based on your business requirements. For more information, see [Permissions for AssumeRole, AssumeRoleWithSAML, and AssumeRoleWithWebIdentity](#) in the *IAM User Guide*.

Note

AWS Transfer Family stores the policy JSON, instead of the Amazon Resource Name (ARN) of the policy. So, when you change the policy in the IAM console, you need to return to AWS Transfer Family console and update your users with the latest policy contents. You can update the user on the **Policy Info** tab in the **User configuration** section.

If you are using the AWS CLI, you can use the following command to update the policy.

```
aws transfer update-user --server-id server --user-name user --policy \
    "$(aws iam get-policy-version --policy-arn policy --version-id version --
    output json)"
```

Preventing users from running `mkdir` in an S3 bucket

You can limit users' ability to create a directory in an Amazon S3 bucket. To do so, you create an IAM policy that allows the `s3:PutObject` action but also denies it when the key ends with a `/` (forward slash). The following example policy allows users to upload files to an Amazon S3 bucket but denies the `mkdir` command in the Amazon S3 bucket.

```
{
  "Sid": "DenyMkdir",
  "Action": [
    "s3:PutObject"
  ],
  "Effect": "Deny",
  "Resource": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*/",
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*/*"
  ]
}
```

Note

The second resource line makes it impossible for users to create sub-folders by running a command such as `put my-file DOC-EXAMPLE-BUCKET/new-folder/my-file`.

AWS CloudTrail logging for AWS Transfer Family

AWS Transfer Family integrates with both AWS CloudTrail and Amazon CloudWatch. CloudTrail and CloudWatch serve different but complementary purposes.

- This topic covers integration with CloudTrail , an AWS service that creates a record of actions taken within your AWS account. It continuously monitors and records API calls for activities like console sign-ins, AWS Command Line Interface commands, and SDK/API calls. This allows you to keep a log of who took what action, when, and from where. CloudTrail helps with auditing, access management, and regulatory compliance by providing a history of all activity in your AWS environment. For details, see the [AWS CloudTrail User Guide](#).
- [Amazon CloudWatch logging for AWS Transfer Family](#) covers integration with CloudWatch, a monitoring service for AWS resources and applications. It collects metrics and logs to provide visibility into resource utilization, application performance, and overall system health. CloudWatch helps with operational tasks like troubleshooting issues, setting alarms and autoscaling. For details, see the [Amazon CloudWatch User Guide](#).

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

For an ongoing record of events in your AWS account, including events for AWS Transfer Family, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for creating a trail](#)
- [CloudTrail supported services and integrations](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

All AWS Transfer Family actions are logged by CloudTrail and are documented in the [Actions API reference](#). For example, calls to the `CreateServer`, `ListUsers` and `StopServer` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity element](#).

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Transfer Family. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**.

Using the information collected by CloudTrail, you can determine the request that was made to AWS Transfer Family, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

Topics

- [Enable AWS CloudTrail logging](#)
- [Example log entry for creating a server](#)

Enable AWS CloudTrail logging

You can monitor AWS Transfer Family API calls using AWS CloudTrail. By monitoring API calls, you can get useful security and operational information. If you have [Amazon S3 object level logging enabled](#), `RoleSessionName` is contained in the `Requester` field as `[AWS:Role Unique Identifier]/username.sessionid@server-id`. For more information about AWS Identity and Access Management (IAM) role unique identifiers, see [Unique identifiers](#) in the *AWS Identity and Access Management User Guide*.

⚠ Important

The maximum length of the RoleSessionName is 64 characters. If the RoleSessionName is longer, the server-id gets truncated.

Example log entry for creating a server

The following example shows a CloudTrail log entry (in JSON format) that demonstrates the CreateServer action.

```
{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AAAA4FFF5HHHHH6NNWWW:user1",
    "arn": "arn:aws:sts::123456789102:assumed-role/Admin/user1",
    "accountId": "123456789102",
    "accessKeyId": "AAAA52C2WWWWW3BB4Z",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2018-12-18T20:03:57Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AAAA4FFF5HHHHH6NNWWW",
        "arn": "arn:aws:iam::123456789102:role/Admin",
        "accountId": "123456789102",
        "userName": "Admin"
      }
    }
  },
  "eventTime": "2024-02-05T19:18:53Z",
  "eventSource": "transfer.amazonaws.com",
  "eventName": "CreateServer",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "11.22.1.2",
  "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36",
  "requestParameters": {
    "domain": "S3",
```



```
    "hostKey": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "protocols": [
      "SFTP"
    ],
    "protocolDetails": {
      "passiveIp": "AUTO",
      "tlsSessionResumptionMode": "ENFORCED",
      "setStatOption": "DEFAULT"
    },
    "securityPolicyName": "TransferSecurityPolicy-2020-06",
    "s3StorageOptions": {
      "directoryListingOptimization": "ENABLED"
    }
  },
  "responseElements": {
    "serverId": "s-1234abcd5678efghi"
  },
  "requestID": "6fe7e9b1-72fc-45b0-a7f9-5840268aeadf",
  "eventID": "4781364f-7c1e-464e-9598-52d06aa9e63a",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789102",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "transfer.us-east-1.amazonaws.com"
  },
  "sessionCredentialFromConsole": "true"
}
```

Amazon CloudWatch logging for AWS Transfer Family

Amazon CloudWatch monitors your AWS Transfer Family resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

The CloudWatch home page automatically displays metrics about Transfer Family and every other AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications, and display custom collections of metrics that you choose.

You can create alarms that watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. For example, you can monitor the files being transferred into a Transfer Family server and use that data to determine whether you need to deploy additional servers to handle increased load. You can also use this data to stop or delete under-used instances to save money.

Types of CloudWatch logging for Transfer Family

Transfer Family provides two ways to log events to CloudWatch:

- JSON structured logging
- Logging via a logging role

For Transfer Family servers, you can choose the logging mechanism that you prefer. For connectors and workflows, only logging roles are supported.

JSON structured logging

For logging server events, we recommend using JSON structured logging. This provides a more comprehensive logging format that enables CloudWatch log querying. For this type of logging, the IAM policy for the user that creates the server (or edits the server's logging configuration) must contain the following permissions:

- `logs:CreateLogDelivery`
- `logs>DeleteLogDelivery`
- `logs:DescribeLogGroups`
- `logs:DescribeResourcePolicies`

- logs:GetLogDelivery
- logs:ListLogDeliveries
- logs:PutResourcePolicy
- logs:UpdateLogDelivery

The following is an example policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs>DeleteLogDelivery",
        "logs:ListLogDeliveries",
        "logs:PutResourcePolicy",
        "logs:DescribeResourcePolicies",
        "logs:DescribeLogGroups"
      ],
      "Resource": "arn:aws:logs:region-id:AWS account:log-group:/aws/transfer/*"
    }
  ]
}
```

For details on setting up JSON structured logging, see [Creating, updating, and viewing logging for servers](#).

Logging role

To log events for a managed workflow that is attached to a server, as well as for connectors, you need to specify a logging role. To set access, you create a resource-based IAM policy and an IAM role that provides that access information. The following is an example policy for an AWS account that can log server events.

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

```
"Statement": [
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogStream",
      "logs:DescribeLogStreams",
      "logs:CreateLogGroup",
      "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*:*:log-group:/aws/transfer/*"
  }
]
```

For details on configuring a logging role to log workflow events see [Managing logging for workflows](#).

Creating, updating, and viewing logging for servers

For all AWS Transfer Family servers, you can choose between two options for logging: `LoggingRole` (used for logging workflows that are attached to the server) or `StructuredLogDestinations`. Benefits of using `StructuredLogDestinations` include the following:

- Receive logs in a structured JSON format.
- Query your logs with Amazon CloudWatch Logs Insights, which automatically discovers JSON formatted fields.
- Share log groups across AWS Transfer Family resources allows you to combine log streams from multiple servers into a single log group, making it easier to manage your monitoring configurations and log retention settings.
- Create aggregated metrics and visualizations that can be added to CloudWatch dashboards.
- Track usage and performance data by using log groups to create consolidated log metrics, visualizations, and dashboards.

The options for `LoggingRole` or `StructuredLogDestinations` are configured and controlled separately. For each server, you can set up one or both methods of logging, or configure your server to have no logging whatsoever (though this is not recommended).

If you create a new server by using the Transfer Family console, logging is enabled by default. After you create the server, you can use the `UpdateServer` API call to change your logging configuration. For details, see [StructuredLogDestinations](#).

Currently, for workflows, if you want logging enabled, you must specify a logging role:

- If you associate a workflow with a server, using either the `CreateServer` or `UpdateServer` API call, the system does not automatically create a logging role. If you want to log your workflow events, you need to explicitly attach a logging role to the server.
- If you create a server using the Transfer Family console and you attach a workflow, logs are sent to a log group that contains the server ID in the name. The format is `/aws/transfer/server-id`, for example, `/aws/transfer/s-1111aaaa2222bbbb3`. The server logs can be sent to this same log group or a different one.

Logging considerations for creating and editing servers in the console

- New servers created through the console only support structured JSON logging, unless a workflow is attached to the server.
- *No logging* is not an option for new servers that you create in the console.
- Existing servers can enable structured JSON logging through the console at any time.
- Enabling structured JSON logging through the console disables the existing logging method, so as to not double charge customers. The exception is if a workflow is attached to the server.
- If you enable structured JSON logging, you cannot later disable it through the console.
- If you enable structured JSON logging, you can change the log group destination through the console at any time.
- If you enable structured JSON logging, you cannot edit the logging role through the console if you have enabled both logging types through the API. The exception is if your server has a workflow attached. However, the logging role does continue to appear in **Additional details**.

Logging considerations for creating and editing servers using the API or SDK

- If you create a new server through the API, you can configure either or both types of logging, or choose no logging.
- For existing servers, enable and disable structured JSON logging at any time.
- You can change the log group through the API at any time.

- You can change the logging role through the API at any time.

To enable structured logging, you must be logged into an account with the following permissions

- `logs:CreateLogDelivery`
- `logs>DeleteLogDelivery`
- `logs:DescribeLogGroups`
- `logs:DescribeResourcePolicies`
- `logs:GetLogDelivery`
- `logs>ListLogDeliveries`
- `logs:PutResourcePolicy`
- `logs:UpdateLogDelivery`

An example policy is available in the section [Configure CloudWatch logging role](#).

Topics

- [Creating logging for servers](#)
- [Updating logging for a server](#)
- [Viewing the server configuration](#)

Creating logging for servers

When you create a new server, on the **Configure additional details** page, you can specify an existing log group, or create a new one.

Transfer Family > Servers > Create server

Step 1
Choose protocols

Step 2
Choose an identity provider

Step 3
Choose an endpoint

Step 4
Choose a domain

Step 5
Configure additional details

Step 6
Review and create

Configure additional details

Logging Info

Log group Info
Choose the CloudWatch log group where your events will be delivered in a structured JSON format

Create a new log group Choose an existing log group

Logging role Info
Choose the IAM role that will be used to deliver events to your CloudWatch logs

Create a new role Choose an existing role

Note Logging role is only required when selecting a workflow in the Managed workflows section below.

If you choose **Create log group**, the CloudWatch console (<https://console.aws.amazon.com/cloudwatch/>) opens to the **Create log group** page. For details, see [Create a log group in CloudWatch Logs](#).

Updating logging for a server

The details for logging depend on the scenario for your update.

Note

When you opt into structured JSON logging, there can be a delay, in rare cases, where Transfer Family stops logging in the old format, but takes some time to start logging in the new JSON format. This can result in events that don't get logged. There won't be any service disruptions, but you should be careful transferring files during the first hour after changing your logging method, as logs could be dropped.

If you are editing an existing server, your options depend on the state of the server.

- The server already has a logging role enabled, but does not have Structured JSON logging enabled.

Edit additional details

Logging [Info](#)

Log group [Info](#)

Choose an existing log group from the dropdown or create a new log group in Amazon CloudWatch

Enable structured JSON logging

/aws/transfer/scooter ▼



Create log group [↗](#)

i Enabling the structured JSON log format will override your existing logging configuration. Potential changes include new log format and log group.

Logging Role [Info](#)

Select an existing role from your account

AWSTransferLoggingAccess ▼



i Workflows events will be delivered to a log group labelled with the server ID.

- The server does not have any logging enabled.

Edit additional details

Logging [Info](#)

Log group [Info](#)

Choose an existing log group from the dropdown or create a new log group in Amazon CloudWatch

Enable structured JSON logging

Choose an existing log group ▼



Create log group ↗

Logging Role [Info](#)

Select an existing role from your account

Choose a role ▼



Logging role is only required when selecting a workflow in the Managed workflows section below.

- The server already has Structured JSON logging enabled, but does not have a logging role specified.

Edit additional details

Logging [Info](#)

Log group [Info](#)

Choose an existing log group from the dropdown or create a new log group in Amazon CloudWatch

Enable structured JSON logging

/aws/transfer/ [redacted] ▼



Create log group ↗

Logging Role [Info](#)

Select an existing role from your account

Choose a role ▼



Logging role is only required when selecting a workflow in the Managed workflows section below.

- The server already has Structured JSON logging enabled, and also has a logging role specified.

Edit additional details

Logging [Info](#)

Log group [Info](#)
Choose an existing log group from the dropdown or create a new log group in Amazon CloudWatch

Enable structured JSON logging

Logging Role [Info](#)
Select an existing role from your account

Workflows events will be delivered to a log group labelled with the server ID.

Viewing the server configuration

The details for the server configuration page depend on your scenario:

Depending on your scenario, the server configuration page might look like one of the following examples:

- No logging is enabled.

Additional details

Edit

Log group -	Domain Amazon S3	Login display banner View the display message
Logging role Info -	Workflow for complete uploads -	SetStat option Ignore
Server host key Info SHA256: [REDACTED]	Workflow for partial uploads -	TLS session resumption -
Security Policy Info TransferSecurityPolicy-2018-11	Managed workflows execution role -	Passive IP -

- Structured JSON logging is enabled.

Additional details

Edit

Log group /aws/transfer/s[REDACTED]	Domain Amazon S3	Login display banner View the display message
Logging role Info -	Workflow for complete uploads -	SetStat option Ignore
Server host key Info SHA256: [REDACTED]	Workflow for partial uploads -	TLS session resumption -
Security Policy Info TransferSecurityPolicy-2020-06	Managed workflows execution role -	Passive IP -

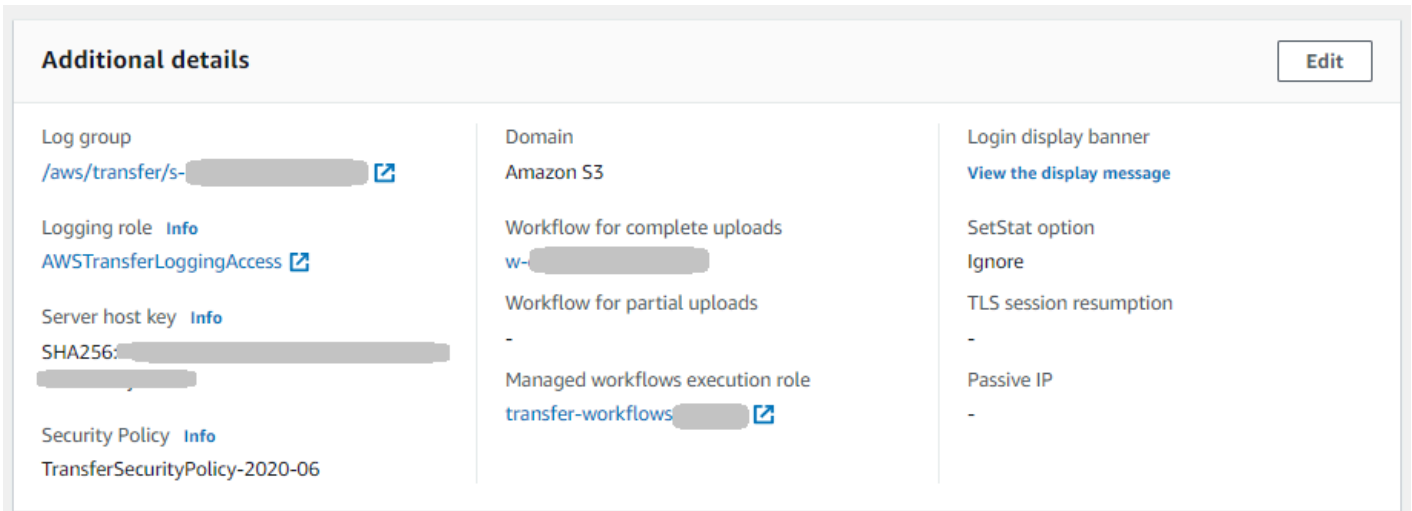
- Logging role is enabled, but structured JSON logging is not enabled.

Additional details

Edit

Log group -	Domain Amazon S3	Login display banner View the display message
Logging role Info AWSTransferLoggingAccess	Workflow for complete uploads w-[REDACTED]	SetStat option Ignore
Server host key Info SHA256:lx39/[REDACTED]	Workflow for partial uploads -	TLS session resumption -
Security Policy Info TransferSecurityPolicy-2018-11	Managed workflows execution role [REDACTED]execution-role[REDACTED]	Passive IP -

- Both types of logging (logging role and structured JSON logging) are enabled.



Managing logging for workflows

CloudWatch provides consolidated auditing and logging for workflow progress and results. Additionally, AWS Transfer Family provides several metrics for workflows. You can view metrics for how many workflows executions started, completed successfully, and failed in the previous minute. All of the CloudWatch metrics for Transfer Family are described in [Using CloudWatch metrics for Transfer Family](#).

View Amazon CloudWatch logs for workflows

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the left navigation pane, choose **Logs**, then choose **Log groups**.
3. On the **Log groups** page, on the navigation bar, choose the correct Region for your AWS Transfer Family server.
4. Choose the log group that corresponds to your server.

For example, if your server ID is s-1234567890abcdef0, your log group is /aws/transfer/s-1234567890abcdef0.

5. On the log group details page for your server, the most recent log streams are displayed. There are two log streams for the user that you are exploring:
 - One for each Secure Shell (SSH) File Transfer Protocol (SFTP) session.

- One for the workflow that is being executed for your server. The format for the log stream for the workflow is `username.workflowID.uniqueStreamSuffix`.

For example, if your user is `mary-major`, you have the following log streams:

```
mary-major-east.1234567890abcdef0  
mary.w-abcdef01234567890.021345abcdef6789
```

 **Note**

The 16-digit alphanumeric identifiers listed in this example are fictitious. The values that you see in Amazon CloudWatch are different.

The **Log events** page for `mary-major-usa-east.1234567890abcdef0` displays the details for each user session, and the `mary.w-abcdef01234567890.021345abcdef6789` log stream contains the details for the workflow.

The following is a sample log stream for `mary.w-abcdef01234567890.021345abcdef6789`, based on a workflow (`w-abcdef01234567890`) that contains a copy step.

```
{  
  "type": "ExecutionStarted",  
  "details": {  
    "input": {  
      "initialFileLocation": {  
        "bucket": "DOC-EXAMPLE-BUCKET",  
        "key": "mary/workflowSteps2.json",  
        "versionId": "version-id",  
        "etag": "etag-id"  
      }  
    }  
  },  
  "workflowId": "w-abcdef01234567890",  
  "executionId": "execution-id",  
  "transferDetails": {  
    "serverId": "s-server-id",  
    "username": "mary",  
    "sessionId": "session-id"  
  }  
}
```

```
},
{
  "type": "StepStarted",
  "details": {
    "input": {
      "fileLocation": {
        "backingStore": "S3",
        "bucket": "DOC-EXAMPLE-BUCKET",
        "key": "mary/workflowSteps2.json",
        "versionId": "version-id",
        "etag": "etag-id"
      }
    },
    "stepType": "COPY",
    "stepName": "copyToShared"
  },
  "workflowId": "w-abcdef01234567890",
  "executionId": "execution-id",
  "transferDetails": {
    "serverId": "s-server-id",
    "username": "mary",
    "sessionId": "session-id"
  }
},
{
  "type": "StepCompleted",
  "details": {
    "output": {},
    "stepType": "COPY",
    "stepName": "copyToShared"
  },
  "workflowId": "w-abcdef01234567890",
  "executionId": "execution-id",
  "transferDetails": {
    "serverId": "server-id",
    "username": "mary",
    "sessionId": "session-id"
  }
},
{
  "type": "ExecutionCompleted",
  "details": {},
  "workflowId": "w-abcdef01234567890",
  "executionId": "execution-id",
```

```

    "transferDetails":{
      "serverId":"s-server-id",
      "username":"mary",
      "sessionId":"session-id"
    }
  }
}

```

Configure CloudWatch logging role

To set access, you create a resource-based IAM policy and an IAM role that provides that access information.

To enable Amazon CloudWatch logging, you start by creating an IAM policy that enables CloudWatch logging. You then create an IAM role and attach the policy to it. You can do this when you are [creating a server](#) or by [editing an existing server](#). For more information about CloudWatch, see [What is Amazon CloudWatch?](#) and [What is Amazon CloudWatch logs?](#) in the *Amazon CloudWatch User Guide*.

Use the following example IAM policies to allow CloudWatch logging.

Use a logging role

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:CreateLogGroup",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/transfer/*"
    }
  ]
}

```

Use structured logging

```

{

```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogDelivery",
      "logs:GetLogDelivery",
      "logs:UpdateLogDelivery",
      "logs>DeleteLogDelivery",
      "logs>ListLogDeliveries",
      "logs:PutResourcePolicy",
      "logs:DescribeResourcePolicies",
      "logs:DescribeLogGroups"
    ],
    "Resource": "arn:aws:logs:region-id:AWS account:log-group:/aws/transfer/
*"
  }
]
}

```

In the preceding example policy, for the **Resource**, replace the *region-id* and *AWS account* with your values. For example, **"Resource": "arn:aws::logs:us-east-1:111122223333:log-group:/aws/transfer/*"**

You then create a role and attach the CloudWatch Logs policy that you created.

To create an IAM role and attach a policy

1. In the navigation pane, choose **Roles**, and then choose **Create role**.

On the **Create role** page, make sure that **AWS service** is chosen.

2. Choose **Transfer** from the service list, and then choose **Next: Permissions**. This establishes a trust relationship between AWS Transfer Family and the IAM role. Additionally, add `aws:SourceAccount` and `aws:SourceArn` condition keys to protect yourself against the *confused deputy* problem. See the following documentation for more details:
 - Procedure for establishing a trust relationship with AWS Transfer Family: [To establish a trust relationship](#)
 - Description for confused deputy problem: [the confused deputy problem](#)

3. In the **Attach permissions policies** section, locate and choose the CloudWatch Logs policy that you just created, and choose **Next: Tags**.
4. (Optional) Enter a key and value for a tag, and choose **Next: Review**.
5. On the **Review** page, enter a name and description for your new role, and then choose **Create role**.
6. To view the logs, choose the **Server ID** to open the server configuration page, and choose **View logs**. You are redirected to the CloudWatch console where you can see your log streams.

On the CloudWatch page for your server, you can see records of user authentication (success and failure), data uploads (PUT operations), and data downloads (GET operations).

Viewing Transfer Family log streams

To view your Transfer Family server logs

1. Navigate to the details page for a server.
2. Choose **View logs**. This opens Amazon CloudWatch.
3. The log group for your selected server is displayed.

The screenshot displays the AWS CloudWatch console interface for a log group. The left sidebar shows navigation options like Dashboards, Alarms, Logs, Metrics, X-Ray traces, Events, Application monitoring, and Insights. The main content area shows the log group details and a list of log streams.

Log group details:

- ARN: `arn:aws:logs:us-east-2:5:log-group:/aws/transfer/s-...:*`
- Metric filters: 0
- Subscription filters: 0
- Contributor Insights rules: -
- Creation time: 2 years ago
- Retention: Never expire
- Stored bytes: 39.39 MB
- Data protection - new: Inactive
- Sensitive data found - new: -
- KMS key ID: -

Log streams (10):

Log stream	Last event
<input type="checkbox"/> ERRORS	2023-
<input type="checkbox"/> scooterstack4-...	2023-
<input type="checkbox"/> scooterstack4-...	2023-
<input type="checkbox"/> scooterstack4-...	2023-

4. You can select a log stream to display details and individual entries for the stream.
 - If there is a listing for **ERRORS**, you can choose it to view details for the latest errors for the server.

CloudWatch > Log groups > /aws/transfer/s- > ERRORS

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Timestamp	Message
There are older events to load. Load more.	
2023-03-23T16:08:29.281-04:00	ERRORS AUTH_FAILURE Method=password User=ubuntu Message= SourceIP=
2023-03-23T16:08:30.979-04:00	ERRORS AUTH_FAILURE Method=password User=ubuntu Message= SourceIP=
2023-03-23T16:08:32.647-04:00	ERRORS AUTH_FAILURE Method=password User=ubuntu Message= SourceIP=
2023-03-23T16:08:34.306-04:00	ERRORS AUTH_FAILURE Method=password User=ubuntu Message= SourceIP=
2023-03-23T16:08:36.010-04:00	ERRORS AUTH_FAILURE Method=password User=ubuntu Message= SourceIP=
2023-03-23T16:08:37.659-04:00	ERRORS AUTH_FAILURE Method=password User=ubuntu Message= SourceIP=
2023-03-23T16:12:33.307-04:00	ERRORS AUTH_FAILURE Method=password User=scooterstack4 Message="Missing POSIX profile" Source...
2023-03-23T16:12:34.943-04:00	ERRORS AUTH_FAILURE Method=password User=scooterstack4 Message="Missing POSIX profile" Source... ERRORS AUTH_FAILURE Method=password User=scooterstack4 Message="Missing POSIX profile" SourceIP=
2023-03-23T16:12:56.857-04:00	ERRORS AUTH_FAILURE Method=password User=debian Message= SourceIP= ERRORS AUTH_FAILURE Method=password User=debian Message= SourceIP=
2023-03-23T16:12:58.430-04:00	ERRORS AUTH_FAILURE Method=password User=debian Message= SourceIP= ERRORS AUTH_FAILURE Method=password User=debian Message= SourceIP=
2023-03-23T16:13:00.106-04:00	ERRORS AUTH_FAILURE Method=password User=debian Message= SourceIP=

- Choose any other entry to see an example log stream.

CloudWatch > Log groups > /aws/transfer/s- > scooterstack4.

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Timestamp	Message
No older events at this moment. Retry	
2023-03-23T16:19:43.747-04:00	scooterstack4. CONNECTED SourceIP= User=scooterstack4 HomeDir=/fs- scooterstack4. CONNECTED SourceIP= User=scooterstack4 HomeDir=/fs- Client=SSH-2.0- OpenSSH_7.4 Role=arn:aws:iam:: :role/ Kex=
2023-03-23T16:19:47.030-04:00	scooterstack4. DISCONNECTED scooterstack4. DISCONNECTED
No newer events at this moment. Auto retry paused. Resume	

- If your server has a managed workflow associated with it, you can view logs for the workflow runs.

Note

The format for the log stream for the workflow is *username.workflowId.uniqueStreamSuffix*. For example, **decrypt-user.w-a1111222233334444.aaaa1111bbbb2222** could be the name of a log stream for user **decrypt-user** and workflow **w-a1111222233334444**.

CloudWatch > Log groups > /aws/transfer/s- > decrypt-user.w-

Log events
You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Actions Create metric filter

Filter events Clear 1m 30m 1h 12h Custom Display

Timestamp	Message
There are older events to load. Load more	
2023-03-21T13:37:57.795-04:00	<code>{"type": "StepStarted", "details": {"input": {"fileLocation": {"backingStore": "S3", "bucket": "...", "key": "decrypt-...</code>
2023-03-21T14:12:02.850-04:00	<pre> { "type": "StepStarted", "details": { "input": { "fileLocation": { "backingStore": "S3", "bucket": "...", "key": "decrypt-user/test.json.gpg", "versionId": "...", "etag": "..." } } }, "stepType": "DECRYPT", "stepName": "decrypt-step" }, "workflowId": "w-...", "executionId": "...", "transferDetails": { "serverId": "s-...", "username": "decrypt-user", "sessionId": "..." } </pre>
2023-03-21T14:12:03.464-04:00	<code>{"type": "StepCompleted", "details": {"output": {}}, "stepType": "DECRYPT", "stepName": "decrypt-step"}, "workflowId": "w-</code>

Note

For any expanded log entry, you can copy the entry to the clipboard by choosing **Copy**. For more details about CloudWatch logs, see [Viewing log data](#).

Creating Amazon CloudWatch alarms

The following example shows how to create Amazon CloudWatch alarms using the AWS Transfer Family metric, FilesIn.

CDK

```
new cloudwatch.Metric({
  namespace: "AWS/Transfer",
  metricName: "FilesIn",
  dimensionsMap: { ServerId: "s-000000000000000000" },
  statistic: "Average",
  period: cdk.Duration.minutes(1),
}).createAlarm(this, "AWS/Transfer FilesIn", {
  threshold: 1000,
  evaluationPeriods: 10,
  datapointsToAlarm: 5,
  comparisonOperator:
  cloudwatch.ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD,
});
```

AWS CloudFormation

```
Type: AWS::CloudWatch::Alarm
Properties:
  Namespace: AWS/Transfer
  MetricName: FilesIn
  Dimensions:
    - Name: ServerId
      Value: s-000000000000000000
  Statistic: Average
  Period: 60
  Threshold: 1000
  EvaluationPeriods: 10
  DatapointsToAlarm: 5
  ComparisonOperator: GreaterThanOrEqualToThreshold
```

Logging Amazon S3 API calls to S3 access logs

If you are [using Amazon S3 access logs to identify S3 requests](#) made on behalf of your file transfer users, RoleSessionName is used to display which IAM role was assumed to service

the file transfers. It also displays additional information such as the user name, session id, and server-id used for the transfers. The format is [AWS:Role Unique Identifier]/username.sessionid@server-id and is contained in the Requester field. For example, the following are the contents for a sample Requester field from an S3 access log for a file that was copied to the S3 bucket.

```
arn:aws:sts::AWS-Account-ID:assumed-role/IamRoleName/  
username.sessionid@server-id
```

In the Requester field above, it shows the IAM Role called `IamRoleName`. For more information about IAM role unique identifiers, see [Unique identifiers](#) in the *AWS Identity and Access Management User Guide*.

Examples to limit confused deputy problem

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. For more details, see [Cross-service confused deputy prevention](#).

Note

In the following examples, replace each *user input placeholder* with your own information.

In these examples, you can remove the ARN details for a workflow if your server doesn't have any workflows attached to it.

The following example logging/invoke policy allows any server (and workflow) in the account to assume the role.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowAllServersWithWorkflowAttached",  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "transfer.amazonaws.com"  
      },  
    },  
  ],  
}
```

```

    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "account-id"
      },
      "ArnLike": {
        "aws:SourceArn": [
          "arn:aws:transfer:region:account-id:server/*",
          "arn:aws:transfer:region:account-id:workflow/*"
        ]
      }
    }
  }
]
}

```

The following example logging/invocation policy allows a specific server (and workflow) to assume the role.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSpecificServerWithWorkflowAttached",
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        },
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:transfer:region:account-id:server/server-id",
            "arn:aws:transfer:region:account-id:workflow/workflow-id"
          ]
        }
      }
    }
  ]
}

```

CloudWatch log structure for Transfer Family

This topic describes the fields that are populated in Transfer Family logs: both for JSON structured log entries and legacy log entries.

Topics

- [JSON structured logs for Transfer Family](#)
- [Legacy logs for Transfer Family](#)

JSON structured logs for Transfer Family

The following table contains details for log entry fields for Transfer Family SFTP/FTP/FTPS actions, in the new JSON structured log format.

Field	Description	Example entry
activity-type	The action by the user	OPEN CLOSE PARTIAL_CLOSE DISCONNECTED CONNECTED
bytes-in	Number of bytes uploaded by the user	29238420042
bytes-out	Number of bytes downloaded by the user	23094032490328
ciphers	Specifies the SSH cipher negotiated for the connection (available ciphers are listed in Cryptographic algorithms)	aes256-gcm@openssh.com
client	The user's client software	SSH-2.0-OpenSSH_7.4
home-dir	The directory that the end user lands on when they connect to the endpoint if their home directory type is PATH: if they have a logical	/user-home-bucket/test

Field	Description	Example entry
	home directory, this value is always /	
kex	Specifies the negotiated SSH key exchange (KEX) for the connection (available KEX are listed in Cryptographic algorithms)	diffie-hellman-group14-sha256
message	Provides more information related to the error	<i><string></i>
method	The authentication method	publickey
mode	Specifies how a client opens a file	CREATE TRUNCATE WRITE
operation	The client operation on a file	OPEN CLOSE
path	Actual file path affected	/user-test-bucket/test-file-1.pdf
resource-arn	A system-assigned, unique identifier for a specific resource (for example, a server)	arn:aws:transfer:ap-northeast-1:12346789012:server/s-1234567890akeu2js2
role	The IAM role of the user	arn:aws:iam::0293883675:role/testuser-role
session-id	A system-assigned, unique identifier for a single session	9ca9a0e1cec6ad9d
source-ip	Client IP address	18.323.0.129
user	The end user's username	myname192

Field	Description	Example entry
user-policy	The permissions specified for the end user: this field is populated if the user's policy is a session policy.	The JSON code for the session policy that is being used

Legacy logs for Transfer Family

The following table contains details for log entries for various Transfer Family actions.

Note

These entries are not in the new JSON structured log format.

The following table contains details for log entries for various Transfer Family actions, in the new JSON structured log format.

Action	Corresponding logs within Amazon CloudWatch Logs
Authentication failures	ERRORS AUTH_FAILURE Method=publickey User=lhr Message="RSA SHA256:Lfz3R2nmLY4raK+b7Rb1rSvUIbAE+a+Hxg0c7l1JIZ0" SourceIP=3.8.172.211
COPY/TAG/DELETE/DECRYPT workflow	<pre>{"type":"StepStarted","details":{"input":{"fileLocation":{"backingStore":"EFS","filesystemId":"fs-12345678","path":"/lhr/regex.py"}}, "stepType":"TAG","stepName":"successful_tag_step","workflowId":"w-1111aaa a2222bbbb3","executionId":"81234abcd-1234-efgh-5678-ijklmnopqr90","transferDetails":{"serverId":"s-1234abcd5678efghi","username":"lhr","sessionId":"1234567890abcdef0"}}</pre>

Action	Corresponding logs within Amazon CloudWatch Logs
Custom step workflow	<pre>{ "type": "CustomStepInvoked", "details": { "output": { "token": "MzM4Mjg5YWUtYTEzMy00YjIzLWI3OGMtYzU4OGI2ZjQyMzE5" }, "stepType": "CUSTOM", "stepName": "efs-s3_copy_2", "workflowId": "w-9283e49d33297c3f7", "executionId": "1234abcd-1234-efgh-5678-ijklmnopqr90", "transferDetails": { "serverId": "s-zzzz1111aaaa22223", "username": "lhr", "sessionId": "1234567890abcdef0" } } }</pre>
Deletes	<pre>lhr.33a8fb495ffb383b DELETE Path=/bucket/user/123.jpg</pre>
Downloads	<pre>lhr.33a8fb495ffb383b OPEN Path=/bucket/user/123.jpg Mode=READ llhr.33a8fb495ffb383b CLOSE Path=/bucket/user/123.jpg BytesOut=3618546</pre>
Logins/Logouts	<pre>user.914984e553bcddb6 CONNECTED SourceIP=1.22.111.222 User=lhr HomeDir=LOGICAL Client=SSH-2.0-OpenSSH_7.4 Role=arn:aws::iam::123456789012:role/sftp-s3-access user.914984e553bcddb6 DISCONNECTED</pre>
Renames	<pre>lhr.33a8fb495ffb383b RENAME Path=/bucket/user/lambo.png NewPath=/bucket/user/ferrari.png</pre>

Action	Corresponding logs within Amazon CloudWatch Logs
Sample workflow error log	<pre>{ "type": "StepErrored", "details": { "errorType": "BAD_REQUEST", "errorMessage": "Cannot tag Efs file", "stepType": "TAG", "stepName": "successful_tag_step", "workflowId": "w-1234abcd5678efghi", "executionId": "81234abcd-1234-efgh-5678-ijklmnopqr90", "transferDetails": { "serverId": "s-1234abcd5678efghi", "username": "lhr", "sessionId": "1234567890abcdef0" } } }</pre>
Symlinks	<pre>lhr.eb49cf7b8651e6d5 CREATE_SYMLINK LinkPath=/fs-12345678/lhr/pqr.jpg TargetPath=abc.jpg</pre>
Uploads	<pre>lhr.33a8fb495ffb383b OPEN Path=/bucket/user/123.jpg Mode=CREATE TRUNCATE WRITE lhr.33a8fb495ffb383b CLOSE Path=/bucket/user/123.jpg BytesIn=3618546</pre>

Action	Corresponding logs within Amazon CloudWatch Logs
Workflows	<pre data-bbox="829 275 1495 1190">{"type":"ExecutionStarted","details":{"input":{"initialFileLocation":{"backingStore":"EFS","filesystemId":"fs-12345678","path":"/lhr/regex.py"}}},"workflowId":"w-1111aaaa2222bbb3","executionId":"1234abcd-1234-efgh-5678-ijklmnopqr90","transferDetails":{"serverId":"s-zzzz1111aaaa22223","username":"lhr","sessionId":"1234567890abcdef0"}} {"type":"StepStarted","details":{"input":{"fileLocation":{"backingStore":"EFS","filesystemId":"fs-12345678","path":"/lhr/regex.py"}},"stepType":"CUSTOM","stepName":"efs-s3_copy_2"},"workflowId":"w-9283e49d33297c3f7","executionId":"1234abcd-1234-efgh-5678-ijklmnopqr90","transferDetails":{"serverId":"s-18ca49dce5d842e0b","username":"lhr","sessionId":"1234567890abcdef0"}}</pre>

Example CloudWatch log entries

This topic presents example log entries.

Topics

- [Example transfer sessions log entries](#)
- [Example log entries for SFTP connectors](#)
- [Example log entries for Key exchange algorithm failures](#)

Example transfer sessions log entries

In this example, an SFTP user connects to a Transfer Family server, uploads a file, then disconnects from the session.

The following log entry reflects an SFTP user connecting to a Transfer Family server.

```
{
  "role": "arn:aws:iam::500655546075:role/scooter-transfer-s3",
  "activity-type": "CONNECTED",
  "ciphers": "chacha20-poly1305@openssh.com, chacha20-poly1305@openssh.com",
  "client": "SSH-2.0-OpenSSH_7.4",
  "source-ip": "52.94.133.133",
  "resource-arn": "arn:aws:transfer:us-east-1:500655546075:server/
s-3fe215d89f074ed2a",
  "home-dir": "/scooter-test/log-me",
  "user": "log-me",
  "kex": "ecdh-sha2-nistp256",
  "session-id": "9ca9a0e1cec6ad9d"
}
```

The following log entry reflects the SFTP user uploading a file into their Amazon S3 bucket.

```
{
  "mode": "CREATE|TRUNCATE|WRITE",
  "path": "/scooter-test/log-me/config-file",
  "activity-type": "OPEN",
  "resource-arn": "arn:aws:transfer:us-east-1:500655546075:server/
s-3fe215d89f074ed2a",
  "session-id": "9ca9a0e1cec6ad9d"
}
```

The following log entries reflect the SFTP user disconnecting from their SFTP session. First, the client closes the connection to the bucket, and then the client disconnects the SFTP session.

```
{
  "path": "/scooter-test/log-me/config-file",
  "activity-type": "CLOSE",
  "resource-arn": "arn:aws:transfer:us-east-1:500655546075:server/
s-3fe215d89f074ed2a",
  "bytes-in": "121",
  "session-id": "9ca9a0e1cec6ad9d"
}
```

```

}

{
  "activity-type": "DISCONNECTED",
  "resource-arn": "arn:aws:transfer:us-east-1:500655546075:server/
s-3fe215d89f074ed2a",
  "session-id": "9ca9a0e1cec6ad9d"
}

```

Example log entries for SFTP connectors

This section contains example logs for both a successful and an unsuccessful transfer. Logs are generated to a log group named `/aws/transfer/connector-id`, where *connector-id* is the identifier for your SFTP connector.

Note

Log entries for SFTP connectors are only generated when you execute a `StartFileTransfer` command.

This log entry is for a transfer that completed successfully.

```

{
  "operation": "RETRIEVE",
  "timestamp": "2023-10-25T16:33:27.373720Z",
  "connector-id": "connector-id",
  "transfer-id": "transfer-id",
  "file-transfer-id": "transfer-id/file-transfer-id",
  "url": "sftp://192.0.2.0",
  "file-path": "/remotebucket/remotefilepath",
  "status-code": "COMPLETED",
  "start-time": "2023-10-25T16:33:26.945481Z",
  "end-time": "2023-10-25T16:33:27.159823Z",
  "account-id": "480351544584",
  "connector-arn": "arn:aws:transfer:us-east-1:480351544584:connector/connector-id",
  "local-directory-path": "/connectors-localbucket"
  "bytes": 514
}

```

This log entry is for a transfer that timed out, and thus was not completed successfully.

```
{
  "operation": "RETRIEVE",
  "timestamp": "2023-10-25T22:33:47.625703Z",
  "connector-id": "connector-id",
  "transfer-id": "transfer-id",
  "file-transfer-id": "transfer-id/file-transfer-id",
  "url": "sftp://192.0.2.0",
  "file-path": "/remotebucket/remotefilepath",
  "status-code": "FAILED",
  "failure-code": "TIMEOUT_ERROR",
  "failure-message": "Transfer request timeout.",
  "account-id": "480351544584",
  "connector-arn": "arn:aws:transfer:us-east-1:480351544584:connector/connector-id",
  "local-directory-path": "/connectors-localbucket"
}
```

This log entry is for a SEND operation that succeeds.

```
{
  "operation": "SEND",
  "timestamp": "2024-04-24T18:16:12.513207284Z",
  "connector-id": "connector-id",
  "transfer-id": "transfer-id",
  "file-transfer-id": "transfer-id/file-transfer-id",
  "url": "sftp://server-id.server.transfer.us-east-1.amazonaws.com",
  "file-path": "/DOC-EXAMPLE-BUCKET/my-test-folder/connector-metrics-us-east-1-2024-01-02.csv",
  "status-code": "COMPLETED",
  "start-time": "2024-04-24T18:16:12.295235884Z",
  "end-time": "2024-04-24T18:16:12.461840732Z",
  "account-id": "255443218509",
  "connector-arn": "arn:aws:transfer:us-east-1:255443218509:connector/connector-id",
  "bytes": 275
}
```

Descriptions for some key fields in the previous log examples.

- `timestamp` represents when the log is added to CloudWatch. `start-time` and `end-time` correspond to when the connector actually starts and finishes a transfer.

- `transfer-id` is a unique identifier that is assigned for each `start-file-transfer` request. If the user passes multiple file paths in a single `start-file-transfer` API call, all the files share the same `transfer-id`.
- `file-transfer-id` is a unique value generated for each file transferred. Note that the initial portion of the `file-transfer-id` is the same as `transfer-id`.

Example log entries for Key exchange algorithm failures

This section contains example logs where the Key exchange algorithm (KEX) failed. These are examples from the **ERRORS** log stream for structured logs.

This log entry is an example where there is a host key type error.

```
{
  "activity-type": "KEX_FAILURE",
  "source-ip": "999.999.999.999",
  "resource-arn": "arn:aws:transfer:us-east-1:999999999999:server/
s-99999999999999999999",
  "message": "no matching host key type found",
  "kex": "ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,ecdsa-sha2-
nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@openssh.com,ecdsa-sha2-
nistp521-cert-v01@openssh.com,ssh-ed25519,ssh-rsa,ssh-dss"
}
```

This log entry is an example where there is a KEX mismatch.

```
{
  "activity-type": "KEX_FAILURE",
  "source-ip": "999.999.999.999",
  "resource-arn": "arn:aws:transfer:us-east-1:999999999999:server/
s-99999999999999999999",
  "message": "no matching key exchange method found",
  "kex": "diffie-hellman-group1-sha1,diffie-hellman-group14-sha1,diffie-hellman-
group14-sha256"
}
```

Using CloudWatch metrics for Transfer Family

Note

You can also get metrics for Transfer Family from within the Transfer Family console itself. For details, see [Monitoring usage in the console](#)

You can get information about your server using CloudWatch metrics. A *metric* represents a time-ordered set of data points that are published to CloudWatch. When using metrics, you must specify the Transfer Family namespace, metric name, and [dimension](#). For more information about metrics, see [Metrics](#) in the *Amazon CloudWatch User Guide*.

The following table describes the CloudWatch metrics for Transfer Family.

Namespace	Metric	Description
AWS/Transfer	BytesIn	The total number of bytes transferred into the server. Units: Count Period: 5 minutes
	BytesOut	The total number of bytes transferred out of the server. Unit: Count Period: 5 minutes
	FilesIn	The total number of files transferred into the server. For servers using the AS2 protocol, this metric represents the number of messages received. Units: Count Period: 5 minutes
	FilesOut	The total number of files transferred out of the server. Units: Count

Namespace	Metric	Description
		Period: 5 minutes
	InboundMessage	<p>The total number of AS2 messages successfully received from a trading partner.</p> <p>Units: Count</p> <p>Period: 5 minutes</p>
	InboundFailedMessage	<p>The total number of AS2 messages that were unsuccessfully received from a trading partner. That is, a trading partner sent a message, but the Transfer Family server was not able to successfully process it.</p> <p>Units: Count</p> <p>Period: 5 minutes</p>
	OnUploadExecutionsStarted	<p>The total number of workflow executions started on the server.</p> <p>Units: Count</p> <p>Period: 1 minute</p>
	OnUploadExecutionsSuccess	<p>The total number of successful workflow executions on the server.</p> <p>Units: Count</p> <p>Period: 1 minute</p>
	OnUploadExecutionsFailed	<p>The total number of unsuccessful workflow executions on the server.</p> <p>Units: Count</p> <p>Period: 1 minute</p>

Transfer Family dimensions

A *dimension* is a name/value pair that is part of the identity of a metric. For more information about dimensions, see [Dimensions](#) in the *Amazon CloudWatch User Guide*.

The following table describes the CloudWatch dimension for Transfer Family.

Dimension	Description
ServerId	The unique ID of the server.

Using AWS User Notifications with AWS Transfer Family

To get notified about AWS Transfer Family events, you can use [AWS User Notifications](#) to set up various delivery channels. When an event matches a rule that you specify, you receive a notification.

You can receive notifications for events through multiple channels, including email, [AWS Chatbot](#) chat notifications, or [AWS Console Mobile Application](#) push notifications. You can also see notifications in the [Console Notifications Center](#). User Notifications supports aggregation, which can reduce the number of notifications that you receive during specific events.

For more information, see the [Customize file delivery notifications using AWS Transfer Family managed workflows](#) blog post, and [What is AWS User Notifications?](#) in the *AWS User Notifications User Guide*.

Using queries to filter log entries

You can use CloudWatch queries to filter and identify log entries for Transfer Family. This section contains some examples.

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. You can create queries or rules.
 - To create a **Logs Insights** query, choose **Logs Insights** from the left navigation panel, and then enter the details for your query.

- To create a **Contributor Insights** rule, choose Insights > Contributor Insights from the left navigation panel and then enter the details for your rule.

3. Run the query or rule that you created.

View the top authentication failure contributors

In your structured logs, an authentication failure log entry looks similar to the following:

```
{
  "method": "password",
  "activity-type": "AUTH_FAILURE",
  "source-ip": "999.999.999.999",
  "resource-arn": "arn:aws:transfer:us-east-1:999999999999:server/s-0123456789abcdef",
  "message": "Invalid user name or password",
  "user": "exampleUser"
}
```

Run the following query to view the top contributors to authentication failures.

```
filter @logStream = 'ERRORS'
| filter `activity-type` = 'AUTH_FAILURE'
| stats count() as AuthFailures by user, method
| sort by AuthFailures desc
| limit 10
```

Rather than using **CloudWatch Logs Insights**, you can create a **CloudWatch Contributors Insights** rule to view authentication failures. Create a rule similar to the following.

```
{
  "AggregateOn": "Count",
  "Contribution": {
    "Filters": [
      {
        "Match": "$.activity-type",
        "In": [
          "AUTH_FAILURE"
        ]
      }
    ],
    "Keys": [
```

```
        "$.user"
      ]
    },
    "LogFormat": "JSON",
    "Schema": {
      "Name": "CloudWatchLogRule",
      "Version": 1
    },
    "LogGroupARNs": [
      "arn:aws:logs:us-east-1:999999999999:log-group:/customer/structured_logs"
    ]
  }
}
```

View log entries where a file was opened

In your structured logs, a file read log entry looks similar to the following:

```
{
  "mode": "READ",
  "path": "/fs-0df669c89d9bf7f45/avtester/example",
  "activity-type": "OPEN",
  "resource-arn": "arn:aws:transfer:us-east-1:999999999999:server/s-0123456789abcdef",
  "session-id": "0049cd844c7536c06a89"
}
```

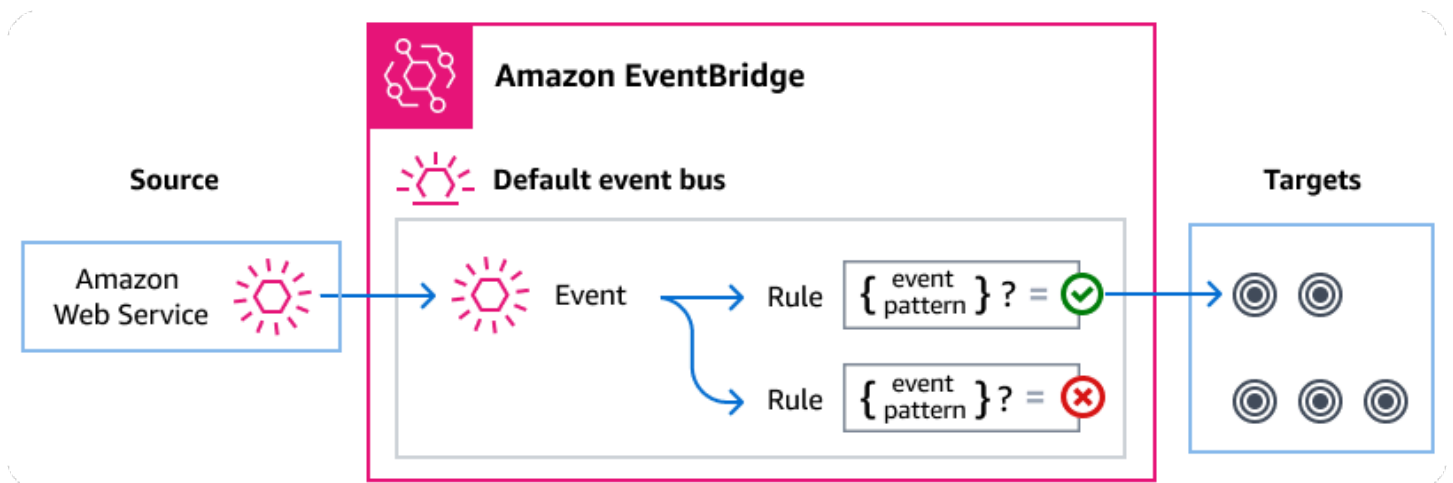
Run the following query to view log entries that indicate a file was opened.

```
filter `activity-type` = 'OPEN'
| display @timestamp, @logStream, `session-id`, mode, path
```

Managing Transfer Family events using Amazon EventBridge

Amazon EventBridge is a serverless service that uses events to connect application components together, which can make it easier for you to build scalable event-driven applications. Event-driven architecture is a style of building loosely coupled software systems that work together by emitting and responding to events. Events represent a change in a resource or environment.

As with many AWS services, Transfer Family generates and sends events to the EventBridge default event bus. Note that the default event bus is automatically provisioned in every AWS account. An event bus is a router that receives events and delivers them to zero or more destinations, or *targets*. You specify rules for the event bus that evaluates events as they arrive. Each rule checks whether an event matches the rule's *event pattern*. If the event matches, the event bus sends the event to one or more specified targets.



Topics

- [Transfer Family events](#)
- [Sending Transfer Family events by using EventBridge rules](#)
- [Amazon EventBridge permissions](#)
- [Additional EventBridge resources](#)
- [Transfer Family events detail reference](#)

Transfer Family events

Transfer Family automatically sends events to the default EventBridge event bus. You can create rules on the event bus where each rule includes an event pattern and one or more targets. Events that match a rule's event pattern are delivered to the specified targets on a [best effort basis](#), however, some events might be delivered out of order.

The following events are generated by Transfer Family. For more information, see [EventBridge events](#) in the *Amazon EventBridge User Guide*.

SFTP, FTPS, and FTP server events

Event detail type	Description
FTP File Server Download Completed	A file has been downloaded successfully for the FTP protocol.
FTP File Server Download Failed	An attempt to download a file has failed for the FTP protocol.
FTP File Server Upload Completed	A file has been uploaded successfully for the FTP protocol.
FTP File Server Upload Failed	An attempt to upload a file has failed for the FTP protocol.
FTPS File Server Download Completed	A file has been downloaded successfully for the FTPS protocol.
FTPS File Server Download Failed	An attempt to download a file has failed for the FTPS protocol.
FTPS File Server Upload Completed	A file has been uploaded successfully for the FTPS protocol.
FTPS File Server Upload Failed	An attempt to upload a file has failed for the FTPS protocol.
SFTP Server File Download Completed	A file has been downloaded successfully for the SFTP protocol.

Event detail type	Description
SFTP Server File Download Failed	An attempt to download a file has failed for the SFTP protocol.
SFTP Server File Upload Completed	A file has been uploaded successfully for the SFTP protocol.
SFTP Server File Upload Failed	An attempt to upload a file has failed for the SFTP protocol.

SFTP connector events

Event detail type	Description
SFTP Connector File Send Completed	A file transfer from a connector to a remote SFTP server has completed successfully.
SFTP Connector File Send Failed	A file transfer from a connector to a remote SFTP server has failed.
SFTP Connector File Retrieve Completed	A file transfer from a remote SFTP server to a connector has completed successfully.
SFTP Connector File Retrieve Failed	A file transfer from a remote SFTP server to a connector has failed.
SFTP Connector Directory Listing Completed	A start file directory listing call that has completed successfully.
SFTP Connector Directory Listing Failed	A start file directory listing that has failed.

A2S events

Event detail type	Description
AS2 Payload Receive Completed	The payload for an AS2 message has been received.
AS2 Payload Receive Failed	The payload for an AS2 message has not been received.
AS2 Payload Send Completed	The payload for an AS2 message has been sent successfully.
AS2 Payload Send Failed	The payload for an AS2 message has failed to send.
AS2 MDN Receive Completed	The message disposition notification for an AS2 message has been received.
AS2 MDN Receive Failed	The message disposition notification for an AS2 message has not been received.
AS2 MDN Send Completed	The message disposition notification for an AS2 message has been sent successfully.
AS2 MDN Send Failed	The message disposition notification for an AS2 message has failed to send.

Sending Transfer Family events by using EventBridge rules

If you want the EventBridge default event bus to send Transfer Family events to a target, you must create a rule that contains an event pattern that matches the data in your desired Transfer Family events.

You can create a rule by following these general steps:

1. Create an event pattern for the rule that specifies the following:
 - Transfer Family is the source of events being evaluated by the rule.
 - (Optional) Any other event data to match it against.

For more information, see [???](#).

2. (Optional) Create an *input transformer* that customizes the data from the event before EventBridge sends the information to the target of the rule.

For more information, see [Input transformation](#) in the *EventBridge User Guide*.

3. Specify the targets to which you want EventBridge to deliver events that match the event pattern.

Targets can be other AWS services, software as a service (SaaS) applications, API destinations, or other custom endpoints. For more information, see [Targets](#) in the *EventBridge User Guide*.

For comprehensive instructions on creating event bus rules, see [Creating rules that react to events](#) in the *EventBridge User Guide*.

Creating event patterns for Transfer Family events

When Transfer Family delivers an event to the default event bus, EventBridge uses the event pattern defined for each rule to determine if the event should be delivered to the rule's targets. An event pattern matches the data in the desired Transfer Family events. Each event pattern is a JSON object that contains the following:

- A `source` attribute that identifies the service sending the event. For Transfer Family events, the source is `aws.transfer`.
- (Optional) A `detail-type` attribute that contains an array of the event types to match.
- (Optional) A `detail` attribute containing any other event data on which to match.

For example, the following event pattern matches against all events from Transfer Family:

```
{
  "source": ["aws.transfer"]
}
```

The following event pattern example matches all of the SFTP connector events:

```
{
  "source": ["aws.transfer"],
  "detail-type": ["SFTP Connector File Send Completed", "SFTP Connector File Retrieve Completed"],
}
```

```
"SFTP Connector File Retrieve Failed", "SFTP Connector File Send
Failed"]
}
```

The following event pattern example matches all Transfer Family failed events:

```
{
  "source": ["aws.transfer"],
  "detail-type": [{"wildcard", "*Failed"}]
}
```

The following event pattern example matches successful SFTP downloads for user *username*:

```
{
  "source": ["aws.transfer"],
  "detail-type": ["SFTP Server File Download Completed"],
  "detail": {
    "username": [username]
  }
}
```

For more information on writing event patterns, see [Event patterns](#) in the *EventBridge User Guide*.

Testing event patterns for Transfer Family events in EventBridge

You can use the EventBridge Sandbox to quickly define and test an event pattern, without having to complete the broader process of creating or editing a rule. Using the Sandbox, you can define an event pattern and use a sample event to confirm that the pattern matches the desired events. EventBridge gives you the option of creating a new rule by using that event pattern directly from the sandbox.

For more information, see [Testing an event pattern using the EventBridge Sandbox](#) in the *EventBridge User Guide*.

Amazon EventBridge permissions

Transfer Family doesn't require any additional permissions to deliver events to Amazon EventBridge.

The targets that you specify might require specific permissions or configuration. For more details on using specific services for targets, see [Amazon EventBridge targets](#) in the *Amazon EventBridge User Guide*.

Additional EventBridge resources

Refer to the following topics in the [Amazon EventBridge User Guide](#) for more information on how to use EventBridge to process and manage events.

- For detailed information on how event buses work, see [Amazon EventBridge event bus](#).
- For information on event structure, see [Events](#).
- For information on constructing event patterns for EventBridge to use when matching events against rules, see [Event patterns](#).
- For information on creating rules to specify which events EventBridge processes, see [Rules](#).
- For information on how to specify what services or other destinations to which EventBridge sends matched events, see [Targets](#).

Transfer Family events detail reference

All events from AWS services have a common set of fields containing metadata about the event. These metadata can include the AWS service that is the source of the event, the time the event was generated, the account and Region in which the event took place, and others. For definitions of these general fields, see [Event structure reference](#) in the *Amazon EventBridge User Guide*.

In addition, each event has a `detail` field that contains data specific to that particular event. The following reference defines the detail fields for the various Transfer Family events.

When you use EventBridge to select and manage Transfer Family events, consider the following:

- The `source` field for all events from Transfer Family is set to `aws.transfer`.
- The `detail-type` field specifies the event type.

For example, `FTP File Server Download Completed`.

- The `detail` field contains the data that is specific to that particular event.

For information on constructing event patterns that enable rules to match Transfer Family events, see [Event patterns](#) in the *Amazon EventBridge User Guide*.

For more information on events and how EventBridge processes them, see [Amazon EventBridge events](#) in the *Amazon EventBridge User Guide*.

Topics

- [SFTP, FTPS, and FTP server events](#)
- [SFTP connector events](#)
- [AS2 events](#)

SFTP, FTPS, and FTP server events

The following are the detail fields for SFTP, FTPS, and FTP server events:

- FTP File Server Download Completed
- FTP File Server Download Failed
- FTP File Server Upload Completed
- FTP File Server Upload Failed
- FTPS File Server Download Completed
- FTPS File Server Download Failed
- FTPS File Server Upload Completed
- FTPS File Server Upload Failed
- SFTP Server File Download Completed
- SFTP Server File Download Failed
- SFTP Server File Upload Completed
- SFTP Server File Upload Failed

The `source` and `detail-type` fields are included below because they contain specific values for Transfer Family events. For definitions of the other metadata fields that are included in all events, see [Event structure reference](#) in the *Amazon EventBridge User Guide*.

```
{
  . . . ,
  "detail-type": "string",
  "source": "aws.transfer",
```

```
. . .,  
"detail": {  
  "failure-code" : "string",  
  "status-code" : "string",  
  "protocol" : "string",  
  "bytes" : "number",  
  "client-ip" : "string",  
  "failure-message" : "string",  
  "end-timestamp" : "string",  
  "etag" : "string",  
  "file-path" : "string",  
  "server-id" : "string",  
  "username" : "string",  
  "session-id" : "string",  
  "start-timestamp" : "string"  
}  
}
```

detail-type

Identifies the type of event.

For this event, the value is one of the SFTP, FTPS, or FTP server event names listed previously.

source

Identifies the service that generated the event. For Transfer Family events, this value is `aws.transfer`.

detail

A JSON object that contains information about the event. The service generating the event determines the content of this field.

For this event, the data includes the following:

failure-code

Category for why the transfer failed. Values: `PARTIAL_UPLOAD` | `PARTIAL_DOWNLOAD` | `UNKNOWN_ERROR`

status-code

Whether the transfer is successful. Values: `COMPLETED` | `FAILED`.

protocol

The protocol used for the transfer. Values: SFTP | FTPS | FTP

bytes

The number of bytes transferred.

client-ip

The IP address for the client involved in the transfer

failure-message

For failed transfers, the details for why the transfer failed.

end-timestamp

For successful transfers, the timestamp for when the file finishes being processed.

etag

The entity tag (only used for Amazon S3 files).

file-path

The path to the file being transferred.

server-id

The unique ID for the Transfer Family server.

username

The user that is performing the transfer.

session-id

The unique identifier for the transfer session.

start-timestamp

For successful transfers, the timestamp for when file processing begins.

Example SFTP Server File Download Failed example event

The following example shows an event where a download failed on an SFTP server (Amazon EFS is the storage being used).


```
{
  "version": "0",
  "id": "event-ID",
  "detail-type": "SFTP Server File Download Failed",
  "source": "aws.transfer",
  "account": "958412138249",
  "time": "2024-01-29T17:20:27Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:transfer:us-east-1:958412138249:server/s-1234abcd5678efghi"
  ],
  "detail": {
    "failure-code": "PARTIAL_DOWNLOAD",
    "status-code": "FAILED",
    "protocol": "SFTP",
    "bytes": 4100,
    "client-ip": "IP-address",
    "failure-message": "File was partially downloaded.",
    "end-timestamp": "2024-01-29T17:20:27.749749117Z",
    "file-path": "/fs-1234abcd5678efghi/user0/test-file",
    "server-id": "s-1234abcd5678efghi",
    "username": "test",
    "session-id": "session-ID",
    "start-timestamp": "2024-01-29T17:20:16.706282454Z"
  }
}
```

Example FTP File Server Upload Completed example event

The following example shows an event where an upload completed successfully on an FTP server (Amazon S3 is the storage being used).

```
{
  "version": "0",
  "id": "event-ID",
  "detail-type": "FTP Server File Upload Completed",
  "source": "aws.transfer",
  "account": "958412138249",
  "time": "2024-01-29T16:31:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:transfer:us-east-1:958412138249:server/s-1111aaaa2222bbbb3"
  ],
}
```

```

  "detail": {
    "status-code": "COMPLETED",
    "protocol": "FTP",
    "bytes": 1048576,
    "client-ip": "10.0.0.141",
    "end-timestamp": "2024-01-29T16:31:43.311866408Z",
    "etag": "b6d81b360a5672d80c27430f39153e2c",
    "file-path": "/DOC-EXAMPLE-BUCKET/test/1mb_file",
    "server-id": "s-1111aaaa2222bbbb3",
    "username": "test",
    "session-id": "event-ID",
    "start-timestamp": "2024-01-29T16:31:42.462088327Z"
  }
}

```

SFTP connector events

The following are the detail fields for SFTP connector events:

- SFTP Connector File Send Completed
- SFTP Connector File Send Failed
- SFTP Connector File Retrieve Completed
- SFTP Connector File Retrieve Failed
- SFTP Connector Directory Listing Completed
- SFTP Connector Directory Listing Failed

The source and detail-type fields are included below because they contain specific values for Transfer Family events. For definitions of the other metadata fields that are included in all events, see [Event structure reference](#) in the *Amazon EventBridge User Guide*.

```

{
  . . . ,
  "detail-type": "string",
  "source": "aws.transfer",
  . . . ,
  "detail": {
    "operation" : "string",
    "max-items" : "number",
    "connector-id" : "string",

```

```

"output-directory-path" : "string",
"listing-id" : "string",
"transfer-id" : "string",
"file-transfer-id" : "string",
"url" : "string",
"file-path" : "string",
"status-code" : "string",
"failure-code" : "string",
"failure-message" : "string",
"start-timestamp" : "string",
"end-timestamp" : "string",
"local-directory-path" : "string",
"remote-directory-path" : "string"
"item-count" : "number"
"truncated" : "boolean"
"bytes" : "number",
"local-file-location" : {
  "domain" : "string",
  "bucket" : "string",
  "key" : "string"
},
"output-file-location" : {
  "domain" : "string",
  "bucket" : "string",
  "key" : "string"
}
}
}
}

```

detail-type

Identifies the type of event.

For this event, the value is one of the SFTP connector event names listed previously.

source

Identifies the service that generated the event. For Transfer Family events, this value is `aws.transfer`.

detail

A JSON object that contains information about the event. The service that generates the event determines the content of this field.

For this event, the data includes the following:

`max-items`

The maximum number of directory/file names to return.

`operation`

Whether the `StartFileTransfer` request is sending or retrieving a file. Values: `SEND` | `RETRIEVE`.

`connector-id`

The unique identifier for the SFTP connector being used.

`output-directory-path`

The path (bucket and prefix) in Amazon S3 to store the results of the file/directory listing.

`listing-id`

A unique identifier for the `StartDirectoryListing` API call. This identifier can be used to check CloudWatch logs to see the status of listing request.

`transfer-id`

The unique identifier for the transfer event (a `StartFileTransfer` request).

`file-transfer-id`

The unique identifier for the file being transferred.

`url`

The URL of the partner's AS2 or SFTP endpoint.

`file-path`

The location and file that is being sent or retrieved.

`status-code`

Whether the transfer is successful. Values: `FAILED` | `COMPLETED`.

`failure-code`

For failed transfers, the reason code for why the transfer failed.

failure-message

For failed transfers, the details for why the transfer failed.

start-timestamp

For successful transfers, the timestamp for when file processing begins.

end-timestamp

For successful transfers, the timestamp for when file processing completes.

local-directory-path

For RETRIEVE requests, the location in which to place the retrieved file.

remote-directory-path

For SEND requests, the file directory in which to place the file on the partner's SFTP server. This is the value for the RemoteDirectoryPath that the user passed to the StartFileTransfer request. You can specify a default directory on the partner's SFTP server. If so, this field is empty.

item-count

The number of items (directories and files) returned for the listing request.

truncated

Whether the list output contains all of the items contained in the remote directory or not.

bytes

The number of bytes being transferred. The value is 0 for failed transfers.

local-file-location

This parameter contains the details of the location of the AWS storage file.

domain

The storage being used. Currently, the only value is S3.

bucket

The container for the object in Amazon S3.

key

The name assigned to the object in Amazon S3.

output-file-location

This parameter contains the details of the location for where to store the results of the directory listing in AWS storage.

domain

The storage being used. Currently, the only value is S3.

bucket

The container for the object in Amazon S3.

key

The name assigned to the object in Amazon S3.

Example SFTP Connector File Send Failed example event

The following example shows an event where an SFTP connector failed while trying to send a file to a remote SFTP server.

```
{
  "version": "0",
  "id": "event-ID",
  "detail-type": "SFTP Connector File Send Failed",
  "source": "aws.transfer",
  "account": "123456789012",
  "time": "2024-01-24T19:30:45Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:transfer:us-east-1:123456789012:connector/c-f1111aaaa2222bbbb3"
  ],
  "detail": {
    "operation": "SEND",
    "connector-id": "c-f1111aaaa2222bbbb3",
    "transfer-id": "transfer-ID",
    "file-transfer-id": "file-transfer-ID",
    "url": "sftp://s-21a23456789012a.server.transfer.us-east-1.amazonaws.com",
    "file-path": "/DOC-EXAMPLE-BUCKET/testfile.txt",
    "status-code": "FAILED",
    "failure-code": "CONNECTION_ERROR",
    "failure-message": "Unknown Host",
    "remote-directory-path": "",
    "bytes": 0,
  }
}
```

```

    "start-timestamp": "2024-01-24T18:29:33.658729Z",
    "end-timestamp": "2024-01-24T18:29:33.993196Z",
    "local-file-location": {
      "domain": "S3",
      "bucket": "DOC-EXAMPLE-BUCKET",
      "key": "testfile.txt"
    }
  }
}

```

Example SFTP Connector File Retrieve Completed example event

The following example shows an event where an SFTP connector successfully retrieved a file sent from a remote SFTP server.

```

{
  "version": "0",
  "id": "event-ID",
  "detail-type": "SFTP Connector File Retrieve Completed",
  "source": "aws.transfer",
  "account": "123456789012",
  "time": "2024-01-24T18:28:08Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:transfer:us-east-1:123456789012:connector/c-f1111aaaa2222bbbb3"
  ],
  "detail": {
    "operation": "RETRIEVE",
    "connector-id": "c-fc68000012345aa18",
    "transfer-id": "file-transfer-ID",
    "file-transfer-id": "file-transfer-ID",
    "url": "sftp://s-21a23456789012a.server.transfer.us-east-1.amazonaws.com",
    "file-path": "testfile.txt",
    "status-code": "COMPLETED",
    "local-directory-path": "/DOC-EXAMPLE-BUCKET",
    "bytes": 63533,
    "start-timestamp": "2024-01-24T18:28:07.632388Z",
    "end-timestamp": "2024-01-24T18:28:07.774898Z",
    "local-file-location": {
      "domain": "S3",
      "bucket": "DOC-EXAMPLE-BUCKET",
      "key": "testfile.txt"
    }
  }
}

```

```
}
}
```

Example SFTP Connector Directory Listing Completed example event

The following example shows an event where a start directory listing call retrieved a listing file from a remote SFTP server.

```
{
  "version": "0",
  "id": "event-ID",
  "detail-type": "SFTP Connector Directory Listing Completed",
  "source": "aws.transfer",
  "account": "123456789012",
  "time": "2024-01-24T18:28:08Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:transfer:us-east-1:123456789012:connector/c-f1111aaaa2222bbbb3"
  ],
  "detail": {
    "max-items": 10000,
    "connector-id": "c-fc68000012345aa18",
    "output-directory-path": "/DOC-EXAMPLE-BUCKET/example/file-listing-output",
    "listing-id": "123456-23aa-7980-abc1-1a2b3c4d5e",
    "url": "sftp://s-21a23456789012a.server.transfer.us-east-1.amazonaws.com",

    "status-code": "COMPLETED",
    "remote-directory-path": "/home",
    "item-count": 10000,
    "truncated": true,
    "start-timestamp": "2024-01-24T18:28:07.632388Z",
    "end-timestamp": "2024-01-24T18:28:07.774898Z",
    "output-file-location": {
      "domain": "S3",
      "bucket": "DOC-EXAMPLE-BUCKET",
      "key": "c-fc1ab90fd0d047e7a-70987273-49nn-4006-bab1-1a7290cc412ba.json"
    }
  }
}
```

AS2 events

The following are the detail fields for AS2 events:

- AS2 Payload Receive Completed
- AS2 Payload Receive Failed
- AS2 Payload Send Completed
- AS2 Payload Send Failed
- AS2 MDN Receive Completed
- AS2 MDN Receive Failed
- AS2 MDN Send Completed
- AS2 MDN Send Failed

The `source` and `detail-type` fields are included below because they contain specific values for Transfer Family events. For definitions of the other metadata fields that are included in all events, see [Event structure reference](#) in the *Amazon EventBridge User Guide*.

```
{
  . . . ,
  "detail-type": "string",
  "source": "aws.transfer",
  . . . ,
  "detail": {
    "s3-attributes" : {
      "file-bucket" : "string",
      "file-key" : "string",
      "json-bucket" : "string",
      "json-key" : "string",
      "mdn-bucket" : "string",
      "mdn-key" : "string"
    }
    "mdn-subject" : "string",
    "mdn-message-id" : "string",
    "disposition" : "string",
    "bytes" : "number",
    "as2-from" : "string",
    "as2-message-id" : "string",
    "as2-to" : "string",
    "connector-id" : "string",
    "client-ip" : "string",
    "agreement-id" : "string",
    "server-id" : "string",
    "requester-file-name" : "string",
```

```
"message-subject" : "string",
"start-timestamp" : "string",
"end-timestamp" : "string",
"status-code" : "string",
"failure-code" : "string",
"failure-message" : "string",
"transfer-id" : "string"
}
}
```

detail-type

Identifies the type of event.

For this event, the value is one of the AS2 events listed previously.

source

Identifies the service that generated the event. For Transfer Family events, this value is `aws.transfer`.

detail

A JSON object that contains information about the event. The service generating the event determines the content of this field.

s3-attributes

Identifies the Amazon S3 bucket and key for the file being transferred. For MDN events, it also identifies the bucket and key for the MDN file.

file-bucket

The container for the object in Amazon S3.

file-key

The name assigned to the object in Amazon S3.

json-bucket

For COMPLETED or FAILED transfers, the container for the JSON file.

json-key

For COMPLETED or FAILED transfers, the name assigned to the JSON file in Amazon S3.

`mdn-bucket`

For MDN events, the container for the MDN file.

`mdn-key`

For MDN events, the name assigned to the MDN file in Amazon S3.

`mdn-subject`

For MDN events, a text description for the message disposition.

`mdn-message-id`

For MDN events, a unique ID for the MDN message.

`disposition`

For MDN events, the category for the disposition.

`bytes`

The number of bytes in the message.

`as2-from`

The AS2 trading partner that is sending the message.

`as2-message-id`

A unique identifier for the AS2 message being transferred.

`as2-to`

The AS2 trading partner that is receiving the message.

`connector-id`

For AS2 messages being sent from a Transfer Family server to a trading partner, the unique identifier for the AS2 connector being used.

`client-ip`

For server events (transfers from a trading partner to a Transfer Family server), the IP address for the client involved in the transfer.

`agreement-id`

For server events, the unique identifier for the AS2 agreement.

server-id

For server events, a unique ID only for the Transfer Family server.

requester-file-name

For payload events, the original name for the file received during the transfer.

message-subject

A text description for the subject of the message.

start-timestamp

For successful transfers, the timestamp for when file processing begins.

end-timestamp

For successful transfers, the timestamp for when file processing completes.

status-code

The code that corresponds to the state of the AS2 message transfer process. Valid values: COMPLETED | FAILED | PROCESSING.

failure-code

For failed transfers, the category for why the transfer failed.

failure-message

For failed transfers, the details for why the transfer failed.

transfer-id

The unique identifier for the transfer event.

Example AS2 Payload Receive Completed example event

```
{
  "version": "0",
  "id": "event-ID",
  "detail-type": "AS2 Payload Receive Completed",
  "source": "aws.transfer",
  "account": "076722215406",
  "time": "2024-02-07T06:47:05Z",
  "region": "us-east-1",
```

```

    "resources": ["arn:aws:transfer:us-east-1:076722215406:connector/
c-1111aaaa2222bbbb3"],
    "detail": {
      "s3-attributes": {
        "file-key": "/inbound/processed/testAs2Message.dat",
        "file-bucket": "DOC-EXAMPLE-BUCKET"
      },
      "client-ip": "client-IP-address",
      "requester-file-name": "testAs2MessageVerifyFile.dat",
      "end-timestamp": "2024-02-07T06:47:06.040031Z",
      "as2-from": "as2-from-ID",
      "as2-message-id": "as2-message-ID",
      "message-subject": "Message from AS2 tests",
      "start-timestamp": "2024-02-07T06:47:05.410Z",
      "status-code": "PROCESSING",
      "bytes": 63,
      "as2-to": "as2-to-ID",
      "agreement-id": "a-1111aaaa2222bbbb3",
      "server-id": "s-1234abcd5678efghi"
    }
  }
}

```

Example AS2 MDN Receive Failed example event

```

{
  "version": "0",
  "id": "event-ID",
  "detail-type": "AS2 MDN Receive Failed",
  "source": "aws.transfer",
  "account": "889901007463",
  "time": "2024-02-06T22:05:09Z",
  "region": "us-east-1",
  "resources": ["arn:aws:transfer:us-east-1:076722215406:server/s-1111aaaa2222bbbb3"],
  "detail": {
    "mdn-subject": "Your Requested MDN Response re: Test run from Id 123456789abcde
to partner ijklmnop987654",
    "s3-attributes": {
      "json-bucket": "DOC-EXAMPLE-BUCKET1",
      "file-key": "/as2Integ/TestOutboundWrongCert.dat",
      "file-bucket": "DOC-EXAMPLE-BUCKET2",
      "json-key": "/as2Integ/failed/TestOutboundWrongCert.dat.json"
    },
    "mdn-message-id": "MDN-message-ID",
  }
}

```

```
"end-timestamp": "2024-02-06T22:05:09.479878Z",
"as2-from": "PartnerA",
"as2-message-id": "as2-message-ID",
"connector-id": "c-1234abcd5678efghj",
"message-subject": "Test run from Id 123456789abcde to partner ijklmnop987654",
"start-timestamp": "2024-02-06T22:05:03Z",
"failure-code": "VERIFICATION_FAILED_NO_MATCHING_KEY_FOUND",
"status-code": "FAILED",
"as2-to": "MyCompany",
"failure-message": "No public certificate matching message signature could be
found in profile: p-1234abcd5678efghj",
"transfer-id": "transfer-ID"
}
}
```

Security in AWS Transfer Family

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

To learn whether an AWS service is within the scope of specific compliance programs, see [AWS services in Scope by Compliance Program](#) and choose the compliance program that you are interested in. For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and Compliance Quick Start Guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- [Architecting for HIPAA Security and Compliance on Amazon Web Services](#) – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

Note

Not all AWS services are HIPAA eligible. For more information, see the [HIPAA Eligible Services Reference](#).

- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Customer Compliance Guides](#) – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).

- [Evaluating Resources with Rules](#) in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see [Security Hub controls reference](#).
- [Amazon GuardDuty](#) – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- [AWS Audit Manager](#) – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

This documentation helps you understand how to apply the shared responsibility model when using AWS Transfer Family. The following topics show you how to configure AWS Transfer Family to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS Transfer Family resources.

We offer a workshop that provides prescriptive guidance and a hands on lab on how you can build a scalable and secure file transfer architecture on AWS without needing to modify existing applications or manage server infrastructure. You can view the details for this workshop [here](#).

Topics

- [Security policies for AWS Transfer Family servers](#)
- [Security policies for AWS Transfer Family SFTP connectors](#)
- [Using hybrid post-quantum key exchange with AWS Transfer Family](#)
- [Data protection in AWS Transfer Family](#)
- [Identity and access management for AWS Transfer Family](#)
- [Compliance validation for AWS Transfer Family](#)
- [Resilience in AWS Transfer Family](#)
- [Infrastructure security in AWS Transfer Family](#)
- [Add a web application firewall](#)
- [Cross-service confused deputy prevention](#)

- [AWS managed policies for AWS Transfer Family](#)

Security policies for AWS Transfer Family servers

Server security policies in AWS Transfer Family allow you to limit the set of cryptographic algorithms (message authentication codes (MACs), key exchanges (KEXs), and cipher suites) associated with your server. For a list of supported cryptographic algorithms, see [Cryptographic algorithms](#). For a list of supported key algorithms for use with server host keys and service-managed user keys, see [Supported algorithms for user and server keys](#).

Note

We strongly recommend updating your servers to our latest security policy. Our latest security policy is the default. Any customer who creates a Transfer Family server using CloudFormation and accepts the default security policy will be automatically assigned the latest policy. If you are concerned about client compatibility, please affirmatively state which security policy you wish to use when creating or updating a server rather than using the default policy, which is subject to change.

To change the security policy for a server, see [Edit the security policy](#).

For more information on security in Transfer Family, see the blog post, [How Transfer Family can help you build a secure, compliant managed file transfer solution](#).

Topics

- [Cryptographic algorithms](#)
- [TransferSecurityPolicy-2024-01](#)
- [TransferSecurityPolicy-2023-05](#)
- [TransferSecurityPolicy-2022-03](#)
- [TransferSecurityPolicy-2020-06](#) and [TransferSecurityPolicy-Restricted-2020-06](#)
- [TransferSecurityPolicy-2018-11](#) and [TransferSecurityPolicy-Restricted-2018-11](#)
- [TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05](#)
- [TransferSecurityPolicy-FIPS-2023-05](#)
- [TransferSecurityPolicy-FIPS-2020-06](#)
- [Post Quantum security policies](#)

Note

`TransferSecurityPolicy-2024-01` is the default security policy attached to your server when creating a server using the console, API, or CLI.

Cryptographic algorithms

For host keys, we support the following algorithms:

- `rsa-sha2-256`
- `rsa-sha2-512`
- `ecdsa-sha2-nistp256`
- `ecdsa-sha2-nistp384`
- `ecdsa-sha2-nistp521`
- `ssh-ed25519`

Additionally, the following security policies allow `ssh-rsa`:

- `TransferSecurityPolicy-2018-11`
- `TransferSecurityPolicy-2020-06`
- `TransferSecurityPolicy-FIPS-2020-06`
- `TransferSecurityPolicy-FIPS-2023-05`
- `TransferSecurityPolicy-FIPS-2024-01`
- `TransferSecurityPolicy-PQ-SSH-FIPS-Experimental-2023-04`

Note

It is important to understand the distinction between the RSA key type—which is always `ssh-rsa`—and the RSA host key algorithm, which can be any of the supported algorithms.

The following is a list of supported cryptographic algorithms for each security policy.

Note

In the following table and policies, note the following use of algorithm types.

- SFTP servers only use algorithms in the **SshCiphers**, **SshKexs**, and **SshMacs** sections.
- FTPS servers only use algorithms in the **TlsCiphers** section.
- FTP servers, since they don't use encryption, do not use any of these algorithms.
- The FIPS-2024-05 and FIPS-2024-01 security policies are identical, except that FIPS-2024-05 doesn't support the `ssh-rsa` algorithm.
- Transfer Family has introduced new restricted policies that closely parallel existing policies:
 - The `TransferSecurityPolicy-Restricted-2018-11` and `TransferSecurityPolicy-2018-11` security policies are identical, except that the restricted policy doesn't support the `chacha20-poly1305@openssh.com` cipher.
 - The `TransferSecurityPolicy-Restricted-2020-06` and `TransferSecurityPolicy-2020-06` security policies are identical, except that the restricted policy doesn't support the `chacha20-poly1305@openssh.com` cipher.

* In the following table, the `chacha20-poly1305@openssh.com` cipher is included in the non-restricted policy only,

Security policy	2024-01	2023-05	2022-03	2020-06	FIPS-2024-05	FIPS-2023-05	FIPS-2020-06	2018-11
				2020-06 restricted	FIPS-2024-01			2018-11 restricted

SshCiphers

aes128-ctr	◆			◆	◆		◆	◆
aes128-gc	◆	◆	◆	◆	◆	◆	◆	◆

Security policy	2024-01	2023-05	2022-03	2020-06 2020-06 restrict ed	FIPS-2024 -05 FIPS-2024 -01	FIPS-2023 -05	FIPS-2020 -06	2018-11 2018-11 restrict ed
-----------------	---------	---------	---------	--------------------------------------	--------------------------------------	------------------	------------------	--------------------------------------

m@openssh.com

aes192-ctr	◆	◆	◆	◆	◆	◆	◆	◆
------------	---	---	---	---	---	---	---	---

aes256-ctr	◆	◆	◆	◆	◆	◆	◆	◆
------------	---	---	---	---	---	---	---	---

aes256-gcm@openssh.com	◆	◆	◆	◆	◆	◆	◆	◆
------------------------	---	---	---	---	---	---	---	---

chacha20-poly1305@openssh.com				◆*				◆*
-------------------------------	--	--	--	----	--	--	--	----

SshKexs

curve25519-sha256	◆	◆	◆					◆
-------------------	---	---	---	--	--	--	--	---

curve25519-sha256@libssh.org	◆	◆	◆					◆
------------------------------	---	---	---	--	--	--	--	---

Security policy	2024-01	2023-05	2022-03	2020-06 2020-06 restrict ed	FIPS-2024 -05 FIPS-2024 -01	FIPS-2023 -05	FIPS-2020 -06	2018-11 2018-11 restrict ed
diffie-he llman- gro up14- sha1								◆
diffie-he llman- gro up14- sha256				◆			◆	◆
diffie-he llman- gro up16- sha512	◆	◆	◆	◆	◆	◆	◆	◆
diffie-he llman- gro up18- sha512	◆	◆	◆	◆	◆	◆	◆	◆

Security policy	2024-01	2023-05	2022-03	2020-06 2020-06 restrict ed	FIPS-2024 -05 FIPS-2024 -01	FIPS-2023 -05	FIPS-2020 -06	2018-11 2018-11 restrict ed
diffie-he llman- group- exchan ge- sha256	◆	◆	◆	◆		◆	◆	◆
ecdh- nist p256- kybe r-512r3- sha256- d00 @openquan tumsafe.o rg	◆				◆			
ecdh- nist p384- kybe r-768r3- sha384- d00 @openquan tumsafe.o rg	◆				◆			

Security policy	2024-01	2023-05	2022-03	2020-06	FIPS-2024-05	FIPS-2023-05	FIPS-2020-06	2018-11
				2020-06 restricted	FIPS-2024-01			2018-11 restricted
ecdh-nistp521kyber-1024r3-sha512-d0@openquantumsafe.org	◆				◆			
ecdh-sha2-nistp256	◆			◆	◆		◆	◆
ecdh-sha2-nistp384	◆			◆	◆		◆	◆
ecdh-sha2-nistp521	◆			◆	◆		◆	◆
x25519kyber-512r3-sha256-d00@amazon.com	◆							

Security policy	2024-01	2023-05	2022-03	2020-06	FIPS-2024-05	FIPS-2023-05	FIPS-2020-06	2018-11
				2020-06 restricted	FIPS-2024-01			2018-11 restricted

SshMacs

hmac-sha1								◆
hmac-sha1-etm@openssh.com								◆
hmac-sha2-256			◆	◆			◆	◆
hmac-sha2-256-etm@openssh.com	◆	◆	◆	◆	◆	◆	◆	◆
hmac-sha2-512			◆	◆			◆	◆
hmac-sha2-512-etm@openssh.com	◆	◆	◆	◆	◆	◆	◆	◆

Security policy	2024-01	2023-05	2022-03	2020-06	FIPS-2024-05	FIPS-2023-05	FIPS-2020-06	2018-11
				2020-06 restricted	FIPS-2024-01			2018-11 restricted
umac-128-etm@openssh.com				◆				◆
umac-128@openssh.com				◆				◆
umac-64-etm@openssh.com								◆
umac-64@openssh.com								◆
TlsCiphers								
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	◆	◆	◆	◆	◆	◆	◆	◆
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	◆	◆	◆	◆	◆	◆	◆	◆

Security policy	2024-01	2023-05	2022-03	2020-06 2020-06 restrictive	FIPS-2024 -05 FIPS-2024 -01	FIPS-2023 -05	FIPS-2020 -06	2018-11 2018-11 restrictive
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	◆	◆	◆	◆	◆	◆	◆	◆
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	◆	◆	◆	◆	◆	◆	◆	◆
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	◆	◆	◆	◆	◆	◆	◆	◆
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	◆	◆	◆	◆	◆	◆	◆	◆
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	◆	◆	◆	◆	◆	◆	◆	◆

Security policy	2024-01	2023-05	2022-03	2020-06	FIPS-2024-05	FIPS-2023-05	FIPS-2020-06	2018-11
				2020-06 restricted	FIPS-2024-01			2018-11 restricted
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	◆	◆	◆	◆	◆	◆	◆	◆
TLS_RSA_WITH_AES_128_CBC_SHA256								◆
TLS_RSA_WITH_AES_256_CBC_SHA256								◆

TransferSecurityPolicy-2024-01

The following shows the TransferSecurityPolicy-2024-01 security policy.

```
{
  "SecurityPolicy": {
    "Fips": false,
    "SecurityPolicyName": "TransferSecurityPolicy-2024-01",
    "SshCiphers": [
      "aes128-gcm@openssh.com",
      "aes256-gcm@openssh.com",
      "aes128-ctr",
      "aes256-ctr",
      "aes192-ctr"
    ],
    "SshKexs": [
      "ecdh-nistp384-kyber-768r3-sha384-d00@openquantumsafe.org",

```

```

        "x25519-kyber-512r3-sha256-d00@amazon.com",
        "ecdh-nistp256-kyber-512r3-sha256-d00@openquantumsafe.org",
        "ecdh-nistp521-kyber-1024r3-sha512-d00@openquantumsafe.org",
        "ecdh-sha2-nistp256",
        "ecdh-sha2-nistp384",
        "ecdh-sha2-nistp521",
        "curve25519-sha256",
        "curve25519-sha256@libssh.org",
        "diffie-hellman-group18-sha512",
        "diffie-hellman-group16-sha512",
        "diffie-hellman-group-exchange-sha256"
    ],
    "SshMacs": [
        "hmac-sha2-256-etm@openssh.com",
        "hmac-sha2-512-etm@openssh.com"
    ],
    "TlsCiphers": [
        "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
        "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
        "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
        "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
        "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
        "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
        "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
        "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
    ]
}
}

```

TransferSecurityPolicy-2023-05

The following shows the TransferSecurityPolicy-2023-05 security policy.

```

{
  "SecurityPolicy": {
    "Fips": false,
    "SecurityPolicyName": "TransferSecurityPolicy-2023-05",
    "SshCiphers": [
      "aes256-gcm@openssh.com",
      "aes128-gcm@openssh.com",
      "aes256-ctr",
      "aes192-ctr"
    ],
  },
}

```

```

    "SshKexs": [
      "curve25519-sha256",
      "curve25519-sha256@libssh.org",
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group-exchange-sha256"
    ],
    "SshMacs": [
      "hmac-sha2-512-etm@openssh.com",
      "hmac-sha2-256-etm@openssh.com"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
    ]
  }
}

```

TransferSecurityPolicy-2022-03

The following shows the TransferSecurityPolicy-2022-03 security policy.

```

{
  "SecurityPolicy": {
    "Fips": false,
    "SecurityPolicyName": "TransferSecurityPolicy-2022-03",
    "SshCiphers": [
      "aes256-gcm@openssh.com",
      "aes128-gcm@openssh.com",
      "aes256-ctr",
      "aes192-ctr"
    ],
    "SshKexs": [
      "curve25519-sha256",
      "curve25519-sha256@libssh.org",
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",

```

```

    "diffie-hellman-group-exchange-sha256"
  ],
  "SshMacs": [
    "hmac-sha2-512-etm@openssh.com",
    "hmac-sha2-256-etm@openssh.com",
    "hmac-sha2-512",
    "hmac-sha2-256"
  ],
  "TlsCiphers": [
    "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
    "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
    "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
    "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
  ]
}
}
}

```

TransferSecurityPolicy-2020-06 and TransferSecurityPolicy-Restricted-2020-06

The following shows the TransferSecurityPolicy-2020-06 security policy.

Note

The TransferSecurityPolicy-Restricted-2020-06 and TransferSecurityPolicy-2020-06 security policies are identical, except that the restricted policy doesn't support the `chacha20-poly1305@openssh.com` cipher.

```

{
  "SecurityPolicy": {
    "Fips": false,
    "SecurityPolicyName": "TransferSecurityPolicy-2020-06",
    "SshCiphers": [
      "chacha20-poly1305@openssh.com", //Not included in TransferSecurityPolicy-
Restricted-2020-06
      "aes128-ctr",

```

```
    "aes192-ctr",
    "aes256-ctr",
    "aes128-gcm@openssh.com",
    "aes256-gcm@openssh.com"
  ],
  "SshKexs": [
    "ecdh-sha2-nistp256",
    "ecdh-sha2-nistp384",
    "ecdh-sha2-nistp521",
    "diffie-hellman-group-exchange-sha256",
    "diffie-hellman-group16-sha512",
    "diffie-hellman-group18-sha512",
    "diffie-hellman-group14-sha256"
  ],
  "SshMacs": [
    "umac-128-etm@openssh.com",
    "hmac-sha2-256-etm@openssh.com",
    "hmac-sha2-512-etm@openssh.com",
    "umac-128@openssh.com",
    "hmac-sha2-256",
    "hmac-sha2-512"
  ],
  "TlsCiphers": [
    "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
    "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
    "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
    "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
  ]
}
}
```

TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11

The following shows the TransferSecurityPolicy-2018-11 security policy.

Note

The TransferSecurityPolicy-Restricted-2018-11 and TransferSecurityPolicy-2018-11 security policies are identical, except that the restricted policy doesn't support the `chacha20-poly1305@openssh.com` cipher.

```
{
  "SecurityPolicy": {
    "Fips": false,
    "SecurityPolicyName": "TransferSecurityPolicy-2018-11",
    "SshCiphers": [
      "chacha20-poly1305@openssh.com", //Not included in TransferSecurityPolicy-
Restricted-2018-11
      "aes128-ctr",
      "aes192-ctr",
      "aes256-ctr",
      "aes128-gcm@openssh.com",
      "aes256-gcm@openssh.com"
    ],
    "SshKexs": [
      "curve25519-sha256",
      "curve25519-sha256@libssh.org",
      "ecdh-sha2-nistp256",
      "ecdh-sha2-nistp384",
      "ecdh-sha2-nistp521",
      "diffie-hellman-group-exchange-sha256",
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group14-sha256",
      "diffie-hellman-group14-sha1"
    ],
    "SshMacs": [
      "umac-64-etm@openssh.com",
      "umac-128-etm@openssh.com",
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512-etm@openssh.com",
      "hmac-sha1-etm@openssh.com",
      "umac-64@openssh.com",
      "umac-128@openssh.com",
      "hmac-sha2-256",
      "hmac-sha2-512",
    ]
  }
}
```



```

    "hmac-sha1"
  ],
  "TlsCiphers": [
    "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
    "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
    "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
    "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384",
    "TLS_RSA_WITH_AES_128_CBC_SHA256",
    "TLS_RSA_WITH_AES_256_CBC_SHA256"
  ]
}
}
}

```

TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05

The following shows the TransferSecurityPolicy-FIPS-2024-01 and TransferSecurityPolicy-FIPS-2024-05 security policies.

Note

The FIPS service endpoint and TransferSecurityPolicy-FIPS-2024-01 and TransferSecurityPolicy-FIPS-2024-05 security policies are only available in some AWS Regions. For more information, see [AWS Transfer Family endpoints and quotas](#) in the *AWS General Reference*.

The only difference between these two security policies is that TransferSecurityPolicy-FIPS-2024-01 supports the `ssh-rsa` algorithm, and TransferSecurityPolicy-FIPS-2024-05 doesn't.

```

{
  "SecurityPolicy": {
    "Fips": true,
    "SecurityPolicyName": "TransferSecurityPolicy-FIPS-2024-01",
    "SshCiphers": [
      "aes128-gcm@openssh.com",

```

```
        "aes256-gcm@openssh.com",
        "aes128-ctr",
        "aes256-ctr",
        "aes192-ctr"
    ],
    "SshKexs": [
        "ecdh-nistp384-kyber-768r3-sha384-d00@openquantumsafe.org",
        "ecdh-nistp256-kyber-512r3-sha256-d00@openquantumsafe.org",
        "ecdh-nistp521-kyber-1024r3-sha512-d00@openquantumsafe.org",
        "ecdh-sha2-nistp256",
        "ecdh-sha2-nistp384",
        "ecdh-sha2-nistp521",
        "diffie-hellman-group18-sha512",
        "diffie-hellman-group16-sha512",
        "diffie-hellman-group-exchange-sha256"
    ],
    "SshMacs": [
        "hmac-sha2-256-etm@openssh.com",
        "hmac-sha2-512-etm@openssh.com"
    ],
    "TlsCiphers": [
        "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
        "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
        "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
        "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
        "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
        "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
        "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
        "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
    ]
}
}
```

TransferSecurityPolicy-FIPS-2023-05

The FIPS certification details for AWS Transfer Family can be found at <https://csrc.nist.gov/projects/cryptographic-module-validation-program/validated-modules/search/all>

The following shows the TransferSecurityPolicy-FIPS-2023-05 security policy.

Note

The FIPS service endpoint and TransferSecurityPolicy-FIPS-2023-05 security policy is only available in some AWS Regions. For more information, see [AWS Transfer Family endpoints and quotas](#) in the *AWS General Reference*.

```
{
  "SecurityPolicy": {
    "Fips": true,
    "SecurityPolicyName": "TransferSecurityPolicy-FIPS-2023-05",
    "SshCiphers": [
      "aes256-gcm@openssh.com",
      "aes128-gcm@openssh.com",
      "aes256-ctr",
      "aes192-ctr"
    ],
    "SshKexs": [
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group-exchange-sha256"
    ],
    "SshMacs": [
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512-etm@openssh.com"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
    ]
  }
}
```

TransferSecurityPolicy-FIPS-2020-06

The FIPS certification details for AWS Transfer Family can be found at <https://csrc.nist.gov/projects/cryptographic-module-validation-program/validated-modules/search/all>

The following shows the TransferSecurityPolicy-FIPS-2020-06 security policy.

Note

The FIPS service endpoint and TransferSecurityPolicy-FIPS-2020-06 security policy are only available in some AWS Regions. For more information, see [AWS Transfer Family endpoints and quotas](#) in the *AWS General Reference*.

```
{
  "SecurityPolicy": {
    "Fips": true,
    "SecurityPolicyName": "TransferSecurityPolicy-FIPS-2020-06",
    "SshCiphers": [
      "aes128-ctr",
      "aes192-ctr",
      "aes256-ctr",
      "aes128-gcm@openssh.com",
      "aes256-gcm@openssh.com"
    ],
    "SshKexs": [
      "ecdh-sha2-nistp256",
      "ecdh-sha2-nistp384",
      "ecdh-sha2-nistp521",
      "diffie-hellman-group-exchange-sha256",
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group14-sha256"
    ],
    "SshMacs": [
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512-etm@openssh.com",
      "hmac-sha2-256",
      "hmac-sha2-512"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
```

```

    "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
    "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
    "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
    "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
  ]
}
}

```

Post Quantum security policies

This table lists the algorithms for the Transfer Family post quantum security policies. These policies are described in detail in [Using hybrid post-quantum key exchange with AWS Transfer Family](#).

The policy listings follow the table.

Security policy	TransferSecurityPolicy-PQ-SH-Experimental-2023-04	TransferSecurityPolicy-PQ-SSH-FIPS-Experimental-2023-04
SSH ciphers		
aes128-ctr		◆
aes128-gcm@openssh.com	◆	◆
aes192-ctr	◆	◆
aes256-ctr	◆	◆
aes256-gcm@openssh.com	◆	◆
KEXs		
ecdh-nistp256-kyber-512r3-sha256-d00@openquantumsafe.org	◆	◆

Security policy	TransferSecurityPolicy-PQ-SH-Experimental-2023-04	TransferSecurityPolicy-PQ-SSH-FIPS-Experimental-2023-04
ecdh-nistp384-kyber-768r3-sha384-d00@openquantumsafe.org	◆	◆
ecdh-nistp521-kyber-1024r3-sha512-d00@openquantumsafe.org	◆	◆
x25519-kyber-512r3-sha256-d00@amazon.com	◆	
diffie-hellman-group14-sha256		◆
diffie-hellman-group16-sha512	◆	◆
diffie-hellman-group18-sha512	◆	◆
ecdh-sha2-nistp384		◆
ecdh-sha2-nistp521		◆
diffie-hellman-group-exchange-sha256	◆	◆
ecdh-sha2-nistp256		◆
curve25519-sha256@libssh.org	◆	
curve25519-sha256	◆	

MACs

Security policy	TransferSecurityPolicy-PQ-SH-Experimental-2023-04	TransferSecurityPolicy-PQ-SSH-FIPS-Experimental-2023-04
hmac-sha2-256-etm@openssh.com	◆	◆
hmac-sha2-256	◆	◆
hmac-sha2-512-etm@openssh.com	◆	◆
hmac-sha2-512	◆	◆
TLS ciphers		
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	◆	◆
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	◆	◆
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	◆	◆
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	◆	◆
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	◆	◆
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	◆	◆
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	◆	◆
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	◆	◆

TransferSecurityPolicy-PQ-SSH-Experimental-2023-04

The following shows the TransferSecurityPolicy-PQ-SSH-Experimental-2023-04 security policy.

```
{
  "SecurityPolicy": {
    "Fips": false,
    "SecurityPolicyName": "TransferSecurityPolicy-PQ-SSH-Experimental-2023-04",
    "SshCiphers": [
      "aes256-gcm@openssh.com",
      "aes128-gcm@openssh.com",
      "aes256-ctr",
      "aes192-ctr"
    ],
    "SshKexs": [
      "ecdh-nistp384-kyber-768r3-sha384-d00@openquantumsafe.org",
      "x25519-kyber-512r3-sha256-d00@amazon.com",
      "ecdh-nistp256-kyber-512r3-sha256-d00@openquantumsafe.org",
      "ecdh-nistp521-kyber-1024r3-sha512-d00@openquantumsafe.org",
      "curve25519-sha256",
      "curve25519-sha256@libssh.org",
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group-exchange-sha256"
    ],
    "SshMacs": [
      "hmac-sha2-512-etm@openssh.com",
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512",
      "hmac-sha2-256"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
    ]
  }
}
```


TransferSecurityPolicy-PQ-SSH-FIPS-Experimental-2023-04

The following shows the TransferSecurityPolicy-PQ-SSH-FIPS-Experimental-2023-04 security policy.

```
{
  "SecurityPolicy": {
    "Fips": true,
    "SecurityPolicyName": "TransferSecurityPolicy-PQ-SSH-FIPS-
Experimental-2023-04",
    "SshCiphers": [
      "aes256-gcm@openssh.com",
      "aes128-gcm@openssh.com",
      "aes256-ctr",
      "aes192-ctr",
      "aes128-ctr"
    ],
    "SshKexs": [
      "ecdh-nistp384-kyber-768r3-sha384-d00@openquantumsafe.org",
      "ecdh-nistp256-kyber-512r3-sha256-d00@openquantumsafe.org",
      "ecdh-nistp521-kyber-1024r3-sha512-d00@openquantumsafe.org",
      "ecdh-sha2-nistp256",
      "ecdh-sha2-nistp384",
      "ecdh-sha2-nistp521",
      "diffie-hellman-group-exchange-sha256",
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group14-sha256"
    ],
    "SshMacs": [
      "hmac-sha2-512-etm@openssh.com",
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512",
      "hmac-sha2-256"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
```

```

    "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
  ]
}
}

```

Security policies for AWS Transfer Family SFTP connectors

SFTP connector security policies in AWS Transfer Family allow you to limit the set of cryptographic algorithms (message authentication codes (MACs), key exchanges (KEXs), and cipher suites) associated with your SFTP connector. The following is a list of supported cryptographic algorithms for each SFTP connector security policy.

Note

`TransferSFTPConnectorSecurityPolicy-2024-03` is the default security policy that is applied to SFTP connectors.

You can change the security policy for your connector. Select **Connectors** from the Transfer Family left navigation pane, and select your connector. Then select **Edit** in the **Sftp configuration** section. In the **Cryptographic algorithm options** section, choose any available security policy from the dropdown list in the **Security Policy** field.

Security policy	TransferSFTPConnectorSecurityPolicy-2024-03	TransferSFTPConnectorSecurityPolicy-2023-07
Ciphers		
aes128-ctr		◆
aes128-gcm@openssh.com	◆	◆
aes192-ctr	◆	◆
aes256-ctr	◆	◆
aes256-gcm@openssh.com	◆	◆
Kexs		

Security policy	TransferSFTPConnectorSecurityPolicy-2024-03	TransferSFTPConnectorSecurityPolicy-2023-07
curve25519-sha256	◆	◆
curve25519-sha256@libssh.org	◆	◆
diffie-hellman-group14-sha1		◆
diffie-hellman-group16-sha512	◆	◆
diffie-hellman-group18-sha512	◆	◆
diffie-hellman-group-exchange-sha256	◆	◆
Macs		
hmac-sha2-512-etm@openssh.com	◆	◆
hmac-sha2-256-etm@openssh.com	◆	◆
hmac-sha2-512	◆	◆
hmac-sha2-256	◆	◆
hmac-sha1		◆
hmac-sha1-96		◆
Host Key Algorithms		
rsa-sha2-256	◆	◆
rsa-sha2-512	◆	◆

Security policy	TransferSFTPConnec torSecurityPolicy-2024-03	TransferSFTPConnec torSecurityPolicy-2023-07
ecdsa-sha2-nistp256	◆	◆
ecdsa-sha2-nistp384	◆	◆
ecdsa-sha2-nistp521	◆	◆
ssh-rsa		◆

Using hybrid post-quantum key exchange with AWS Transfer Family

AWS Transfer Family supports a hybrid post-quantum key establishment option for the Secure Shell (SSH) protocol. Post-quantum key establishment is needed because it's already possible to record network traffic and save it for decryption in future by a quantum computer, which is called a *store-now-harvest-later* attack.

You can use this option when you connect to Transfer Family for secure file transfers into and out of Amazon Simple Storage Service (Amazon S3) storage or Amazon Elastic File System (Amazon EFS). Post-quantum hybrid key establishment in SSH introduces post-quantum key establishment mechanisms, which it uses in conjunction with classical key exchange algorithms. SSH keys created with classical cipher suites are safe from brute-force attacks with current technology. However, classical encryption isn't expected to remain secure after the emergence of large-scale quantum computing in the future.

If your organization relies on the long-term confidentiality of data passed over a Transfer Family connection, you should consider a plan to migrate to post-quantum cryptography before large-scale quantum computers become available for use.

To protect data encrypted today against potential future attacks, AWS is participating with the cryptographic community in the development of quantum-resistant or post-quantum algorithms. We've implemented hybrid post-quantum key exchange cipher suites in Transfer Family that combine classic and post-quantum elements.

These hybrid cipher suites are available for use on your production workloads in most AWS Regions. However, because the performance characteristics and bandwidth requirements of hybrid

cipher suites are different from those of classic key exchange mechanisms, we recommend that you test them on your Transfer Family connections.

Find out more about post-quantum cryptography in the [Post-Quantum Cryptography](#) security blog post.

Contents

- [About post-quantum hybrid key exchange in SSH](#)
- [How post-quantum hybrid key establishment works in Transfer Family](#)
 - [Why Kyber?](#)
 - [Post-quantum hybrid SSH key exchange and cryptographic requirements \(FIPS 140\)](#)
- [Testing post-quantum hybrid key exchange in Transfer Family](#)
 - [Enable post-quantum hybrid key exchange on your SFTP endpoint](#)
 - [Set up an SFTP client that supports post-quantum hybrid key exchange](#)
 - [Confirm post-quantum hybrid key exchange in SFTP](#)

About post-quantum hybrid key exchange in SSH

Transfer Family supports post-quantum hybrid key exchange cipher suites, which uses both the classical [Elliptic Curve Diffie-Hellman \(ECDH\)](#) key exchange algorithm, and CRYSTALS [Kyber](#). Kyber is a post-quantum public-key encryption and key-establishment algorithm that the [National Institute for Standards and Technology\(NIST\)](#) has designated as its first standard post-quantum key-agreement algorithm.

The client and server still do an ECDH key exchange. Additionally, the server encapsulates a post-quantum shared secret to the client's post-quantum KEM public key, which is advertised in the client's SSH key exchange message. This strategy combines the high assurance of a classical key exchange with the security of the proposed post-quantum key exchanges, to help ensure that the handshakes are protected as long as the ECDH or the post-quantum shared secret cannot be broken.

How post-quantum hybrid key establishment works in Transfer Family

AWS recently announced support for post-quantum key exchange in SFTP file transfers in AWS Transfer Family. Transfer Family securely scales business-to-business file transfers to AWS Storage services using SFTP and other protocols. SFTP is a more secure version of the File Transfer Protocol

(FTP) that runs over SSH. The post-quantum key exchange support of Transfer Family raises the security bar for data transfers over SFTP.

The post-quantum hybrid key exchange SFTP support in Transfer Family includes combining post-quantum algorithms Kyber-512, Kyber-768, and Kyber-1024, with ECDH over P256, P384, P521, or Curve25519 curves. The following corresponding SSH key exchange methods are specified in [the post-quantum hybrid SSH key exchange draft](#).

- `ecdh-nistp256-kyber-512r3-sha256-d00@openquantumsafe.org`
- `ecdh-nistp384-kyber-768r3-sha384-d00@openquantumsafe.org`
- `ecdh-nistp521-kyber-1024r3-sha512-d00@openquantumsafe.org`
- `x25519-kyber-512r3-sha256-d00@amazon.com`

Note

These new key exchange methods may change as the draft evolves towards standardization, or when NIST ratifies the Kyber algorithm.

Why Kyber?

AWS is committed to supporting standardized, interoperable algorithms. Kyber is the first post-quantum encryption algorithm selected for standardization by the [NIST Post-Quantum Cryptography project](#). Some standards bodies are already integrating Kyber into protocols. AWS already supports Kyber in TLS in some AWS API endpoints.

As part of this commitment, AWS has submitted a draft proposal to the IETF for post-quantum cryptography that combines Kyber with NIST-approved curves like P256 for SSH. To help enhance security for our customers, the AWS implementation of the post-quantum key exchange in SFTP and SSH follows that draft. We plan to support future updates to it until our proposal is adopted by the IETF and becomes a standard.

The new key exchange methods (listed in section [How post-quantum hybrid key establishment works in Transfer Family](#)) might change as the draft evolves towards standardization or when NIST ratifies the Kyber algorithm.

Note

Post-quantum algorithm support is currently available for post-quantum hybrid key exchange in TLS for AWS KMS (see [Using hybrid post-quantum TLS with AWS KMS](#)), AWS Certificate Manager, and AWS Secrets Manager API endpoints.

Post-quantum hybrid SSH key exchange and cryptographic requirements (FIPS 140)

For customers that require FIPS compliance, Transfer Family provides FIPS-approved cryptography in SSH by using the AWS FIPS 140-certified, open-source cryptographic library, AWS-LC. The post-quantum hybrid key exchange methods supported in the TransferSecurityPolicy-PQ-SSH-FIPS-Experimental-2023-04 in Transfer Family are FIPS approved according to [NIST's SP 800-56Cr2 \(section 2\)](#). The German Federal Office for Information Security ([BSI](#)) and the Agence nationale de la sécurité des systèmes d'information ([ANSSI](#)) of France also recommend such post-quantum hybrid key exchange methods.

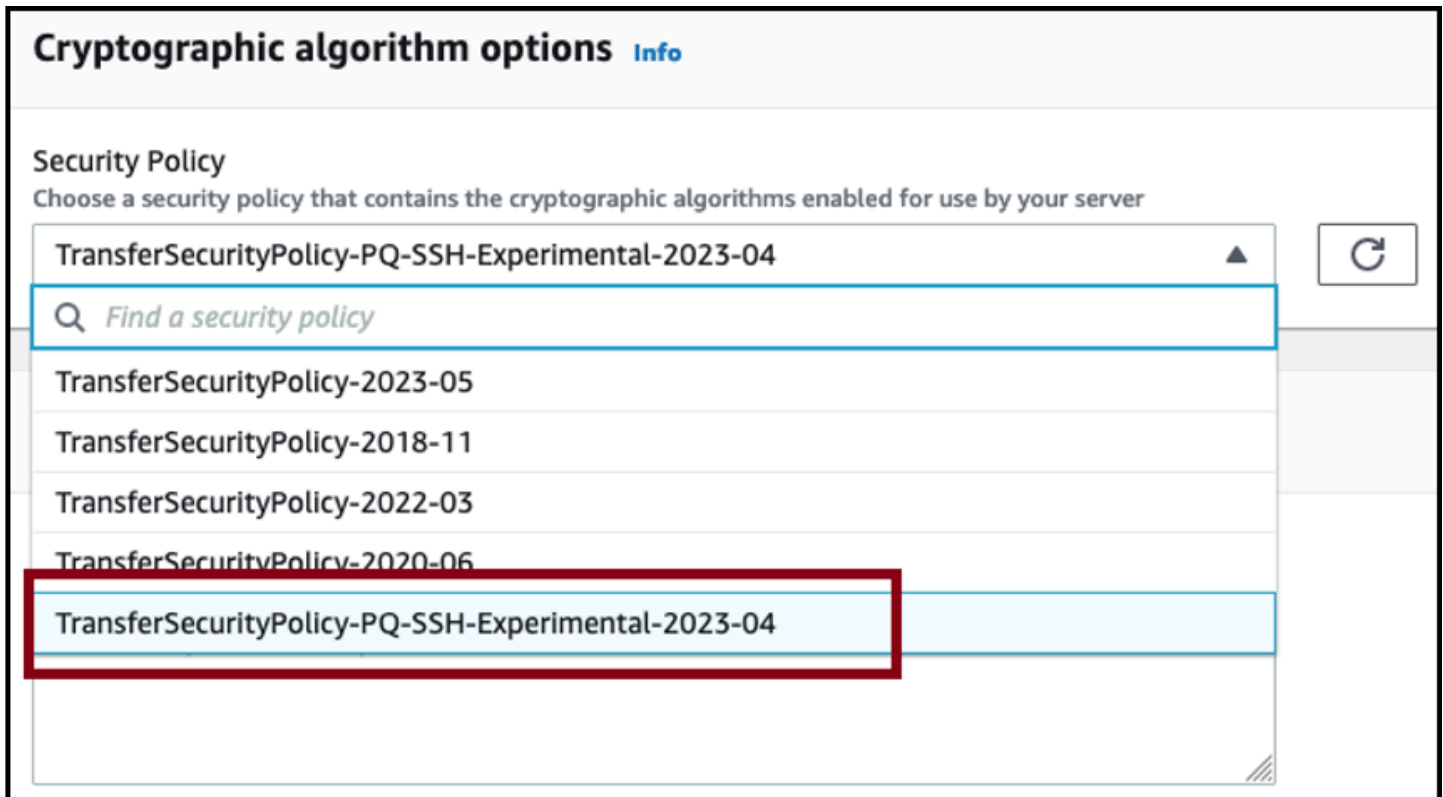
Testing post-quantum hybrid key exchange in Transfer Family

This section describes the steps you take to test post-quantum hybrid key exchange.

1. [Enable post-quantum hybrid key exchange on your SFTP endpoint.](#)
2. Use an SFTP client (such as [Set up an SFTP client that supports post-quantum hybrid key exchange](#)) that supports post-quantum hybrid key exchange by following the guidance in the aforementioned draft specification.
3. Transfer a file using a Transfer Family server.
4. [Confirm post-quantum hybrid key exchange in SFTP.](#)

Enable post-quantum hybrid key exchange on your SFTP endpoint

You can choose the SSH policy when you create a new SFTP server endpoint in Transfer Family, or by editing the Cryptographic algorithm options in an existing SFTP endpoint. The following snapshot shows an example of the AWS Management Console where you update the SSH policy.



The SSH policy names that support post-quantum key exchange are **TransferSecurityPolicy-PQ-SSH-Experimental-2023-04** and **TransferSecurityPolicy-PQ-SSH-FIPS-Experimental-2023-04**. For more details on Transfer Family policies, see [Security policies for AWS Transfer Family servers](#).

Set up an SFTP client that supports post-quantum hybrid key exchange

After you select the correct post-quantum SSH policy in your SFTP Transfer Family endpoint, you can experiment with post-quantum SFTP in Transfer Family. You can use an SFTP client (such as [OQS OpenSSH](#)) that supports post-quantum hybrid key exchange by following the guidance in the aforementioned draft specification.

OQS OpenSSH is an open-source fork of OpenSSH that adds quantum-safe cryptography to SSH by using `liboqs`. `liboqs` is an open-source C library that implements quantum-resistant cryptographic algorithms. OQS OpenSSH and `liboqs` are part of the Open Quantum Safe (OQS) project.

To test post-quantum hybrid key exchange in Transfer Family SFTP with OQS OpenSSH, you need to build OQS OpenSSH as explained in the project's [README](#). After you build OQS OpenSSH, you can run the example SFTP client to connect to your SFTP endpoint (for example,

s-1111aaaa2222bbbb3.server.transfer.us-west-2.amazonaws.com) by using the post-quantum hybrid key exchange methods, as shown in the following command.

```
./sftp -S ./ssh -v -o \  
  KexAlgorithms=ecdh-nistp384-kyber-768r3-sha384-d00@openquantumsafe.org \  
  -i username_private_key_PEM_file \  
  username@server-id.server.transfer.region-id.amazonaws.com
```

In the previous command, replace the following items with your own information:

- Replace *username_private_key_PEM_file* with the SFTP user's private key PEM-encoded file
- Replace *username* with the SFTP user name
- Replace *server-id* with the Transfer Family server ID
- Replace *region-id* with the actual region where your Transfer Family server is located

Confirm post-quantum hybrid key exchange in SFTP

To confirm that post-quantum hybrid key exchange was used during an SSH connection for SFTP to Transfer Family, check the client output. Optionally, you can use a packet capture program. If you use the Open Quantum Safe OpenSSH client, the output should look similar to the following (omitting irrelevant information for brevity):

```
./sftp -S ./ssh -v -o KexAlgorithms=ecdh-nistp384-kyber-768r3-sha384-  
d00@openquantumsafe.org -  
i username_private_key_PEM_file username@s-1111aaaa2222bbbb3.server.transfer.us-  
west-2.amazonaws.com  
OpenSSH_8.9-2022-01_p1, Open Quantum Safe 2022-08, OpenSSL 3.0.2 15 Mar 2022  
debug1: Reading configuration data /home/lab/openssh/oqs-test/tmp/ssh_config  
debug1: Authenticator provider $SSH_SK_PROVIDER did not resolve; disabling  
debug1: Connecting to s-1111aaaa2222bbbb3.server.transfer.us-west-2.amazonaws.com  
  [xx.yy.zz..12] port 22.  
debug1: Connection established.  
[...]  
debug1: Local version string SSH-2.0-OpenSSH_8.9-2022-01_  
debug1: Remote protocol version 2.0, remote software version AWS_SFTP_1.1  
debug1: compat_banner: no match: AWS_SFTP_1.1  
debug1: Authenticating to s-1111aaaa2222bbbb3.server.transfer.us-  
west-2.amazonaws.com:22 as 'username'  
debug1: load_hostkeys: fopen /home/lab/.ssh/known_hosts2: No such file or directory  
[...]
```

```
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: ecdh-nistp384-kyber-768r3-sha384-d00@openquantumsafe.org
debug1: kex: host key algorithm: ssh-ed25519
debug1: kex: server->client cipher: aes192-ctr MAC: hmac-sha2-256-etm@openssh.com
compression: none
debug1: kex: client->server cipher: aes192-ctr MAC: hmac-sha2-256-etm@openssh.com
compression: none
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: SSH2_MSG_KEX_ECDH_REPLY received
debug1: Server host key: ssh-ed25519 SHA256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649
[...]
debug1: rekey out after 4294967296 blocks
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: rekey in after 4294967296 blocks
[...]
Authenticated to AWS.Transfer.PQ.SFTP.test-endpoint.aws.com ([xx.yy.zz..12]:22) using
"publickey".s
debug1: channel 0: new [client-session]
[...]
Connected to s-1111aaaa2222bbbb3.server.transfer.us-west-2.amazonaws.com.
sftp>
```

The output shows that client negotiation occurred using the post-quantum hybrid *ecdh-nistp384-kyber-768r3-sha384-d00@openquantumsafe.org* method and successfully established an SFTP session.

Data protection in AWS Transfer Family

The AWS [shared responsibility model](#) applies to data protection in AWS Transfer Family (Transfer Family). As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data privacy FAQ](#). For information about data protection in Europe, see the [AWS shared responsibility model and GDPR](#) blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS IAM Identity Center. That way each user is given only the

permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We support TLS 1.2.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- 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](#).

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Transfer Family or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any configuration data that you enter into Transfer Family service configuration, or other services' configurations, might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

In contrast, data from upload and download operations into and out of Transfer Family servers is treated as completely private and never exists outside of encrypted channels—such as an SFTP or FTPS connection. This data is only ever accessible to authorized persons.

Topics

- [Data encryption in Amazon S3](#)
- [Managing SSH and PGP keys in Transfer Family](#)

Data encryption in Amazon S3

AWS Transfer Family uses the default encryption options you set for your Amazon S3 bucket to encrypt your data. When you enable encryption on a bucket, all objects are encrypted when they are stored in the bucket. The objects are encrypted by using server-side encryption with either Amazon S3 managed keys (SSE-S3) or AWS Key Management Service (AWS KMS) managed keys

(SSE-KMS). For information about server-side encryption, see [Protecting data using server-side encryption](#) in the *Amazon Simple Storage Service User Guide*.

The following steps show you how to encrypt data in AWS Transfer Family.

To allow encryption in AWS Transfer Family

1. Enable default encryption for your Amazon S3 bucket. For instructions, see [Amazon S3 default encryption for S3 buckets](#) in the *Amazon Simple Storage Service User Guide*.
2. Update the AWS Identity and Access Management (IAM) role policy that is attached to the user to grant the required AWS Key Management Service (AWS KMS) permissions.
3. If you are using a session policy for the user, the session policy must grant the required AWS KMS permissions.

The following example shows an IAM policy that grants the minimum permissions required when using AWS Transfer Family with an Amazon S3 bucket that is enabled for AWS KMS encryption. Include this example policy in both the user IAM role policy and session policy, if you are using one.

```
{
  "Sid": "Stmt1544140969635",
  "Action": [
    "kms:Decrypt",
    "kms:Encrypt",
    "kms:GenerateDataKey"
  ],
  "Effect": "Allow",
  "Resource": "arn:aws:kms:region:account-id:key/kms-key-id"
}
```

Note

The KMS key ID that you specify in this policy must be the same as the one specified for the default encryption in step 1.

Root, or the IAM role that is used for the user, must be allowed in the AWS KMS key policy. For information about the AWS KMS key policy, see [Using key policies in AWS KMS](#) in the *AWS Key Management Service Developer Guide*.

Managing SSH and PGP keys in Transfer Family

In this section, you can find information about SSH keys, including how to generate them and how to rotate them. For details about using Transfer Family with AWS Lambda to manage keys, see the blog post [Enabling user self-service key management with AAWS Transfer Family and AWS Lambda](#).

Note

AWS Transfer Family accepts RSA, ECDSA, and ED25519 keys.

This section also covers how to generate and manage Pretty Good Privacy (PGP) keys.

Topics

- [Supported algorithms for user and server keys](#)
- [Generate SSH keys for service-managed users](#)
- [Rotate SSH keys](#)
- [Generate and manage PGP keys](#)
- [Supported PGP clients](#)

Supported algorithms for user and server keys

The following key algorithms are supported for user and server key-pairs within AWS Transfer Family.

Note

For algorithms to use with PGP decryption in workflows, see [Algorithms supported for PGP key-pairs](#).

- For ED25519: `ssh-ed25519`
- For RSA:
 - `rsa-sha2-256`
 - `rsa-sha2-512`

- For ECDSA:
 - `ecdsa-sha2-nistp256`
 - `ecdsa-sha2-nistp384`
 - `ecdsa-sha2-nistp521`

Note

We support `ssh-rsa` with SHA1 for our older security policies. For details, see [Cryptographic algorithms](#).

Generate SSH keys for service-managed users

You can set up your server to authenticate users using the service managed authentication method, where usernames and SSH keys are stored within the service. The user's public SSH key is uploaded to the server as a user's property. This key is used by the server as part of a standard key-based authentication process. Each user can have multiple public SSH keys on file with an individual server. For limits on number of keys that can be stored per user, see [AWS Transfer Family endpoints and quotas](#) in the *Amazon Web Services General Reference*.

As an alternative to the service managed authentication method, you can authenticate users using a custom identity provider, or AWS Directory Service for Microsoft Active Directory. For more information, see [Working with custom identity providers](#) or [Using AWS Directory Service for Microsoft Active Directory](#).

A server can only authenticate users using one method (service managed, directory service, or custom identity provider), and that method cannot be changed after the server is created.

Topics

- [Creating SSH keys on macOS, Linux, or Unix](#)
- [Creating SSH keys on Microsoft Windows](#)
- [Convert an SSH2 public key to PEM format](#)

Creating SSH keys on macOS, Linux, or Unix

On the macOS, Linux, or Unix operating systems, you use the `ssh-keygen` command to create an SSH public key and SSH private key also known as a key pair.

To create SSH keys on a macOS, Linux, or Unix operating system

1. On macOS, Linux, or Unix operating systems, open a command terminal.
2. AWS Transfer Family accepts RSA-, ECDSA-, and ED25519-formatted keys. Choose the appropriate command based on the type of key-pair you are generating.

Note

In the following examples, we do not specify a passphrase: in this case, the tool asks you to enter your passphrase and then repeat it to verify. Creating a passphrase offers better protection for your private key, and might also improve overall system security. You cannot recover your passphrase: if you forget it, you must create a new key. However, if you are generating a server host key, you *must* specify an empty passphrase, by specifying the `-N ""` option in the command (or by pressing **Enter** twice when prompted), because Transfer Family servers cannot request a password at start-up.

- To generate an RSA 4096-bit key pair:

```
ssh-keygen -t rsa -b 4096 -f key_name
```

- To generate an ECDSA 521-bit key-pair (ECDSA has bit sizes of 256, 384, and 521):

```
ssh-keygen -t ecdsa -b 521 -f key_name
```

- To generate an ED25519 key pair:

```
ssh-keygen -t ed25519 -f key_name
```

Note

key_name is the SSH key pair file name.

The following shows an example of the `ssh-keygen` output.

```
ssh-keygen -t rsa -b 4096 -f key_name
Generating public/private rsa key pair.

Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in key_name.
Your public key has been saved in key_name.pub.
The key fingerprint is:
SHA256:8tDDwPmanTFcEzjTwPGETVW0GW1nVz+gtCCE8hL7PrQ bob.amazon.com
The key's randomart image is:
+---[RSA 4096]-----+
|  . ....E      |
| . = ...      |
|. . . = ..o    |
| . o + oo =    |
| + = .S.= *    |
| . o o ..B + o |
|   .o.+.* .    |
|   =o*+*.      |
|   ..*o*+.     |
+-----[SHA256]-----+
```

Note

When you run the `ssh-keygen` command as shown preceding, it creates the public and private keys as files in the current directory.

Your SSH key pair is now ready to use. Follow steps 3 and 4 to store the SSH public key for your service-managed users. These users use the keys when they transfer files on Transfer Family server endpoints.

3. Navigate to the `key_name.pub` file and open it.
4. Copy the text and paste it in **SSH public key** for the service-managed user.
 - a. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>, then select **Servers** from the navigation pane.
 - b. On the **Servers** page, select the **Server ID** for server that contains the user that you want to update.

- c. Select the user for which you are adding a public key.
- d. In the **SSH public keys** pane, choose **Add SSH public key**.

The screenshot shows the AWS Transfer Family console for a user named 'OneUser'. The breadcrumb navigation is 'Transfer Family > Servers > s-[server icon] > User: OneUser'. The page title is 'User: OneUser' with 'View logs' and 'Delete' buttons. The 'User configuration' section includes: Role (with a link to 'Role'), Policy (with a 'View' button), Posix Profile (User ID: 2001, Group ID: 2001, Secondary Group IDs: -), and Home directory (/fs-[redacted]/[redacted] Restricted). Below this is the 'SSH public keys (1)' section with 'Delete' and 'Add SSH public key' buttons. A table lists one key with columns for 'Date imported' (6/14/2022, 12:53:34 PM) and 'Fingerprint' (SHA256-[redacted]).

- e. Paste the text of the public key you generated into the SSH public key text box, and then choose **Add key**.

The screenshot shows the 'Add key' dialog box in the AWS Transfer Family console. The breadcrumb navigation is 'Transfer Family > Servers > s-[server icon] > OneUser > Add key'. The title is 'Add key'. The 'SSH public keys' section contains the instruction 'SSH public key Info Paste the contents of SSH public key' and a text input field with the placeholder 'Enter SSH public key'. At the bottom right are 'Cancel' and 'Add key' buttons.

The new key is listed in the SSH public key pane.

SSH public keys (2)		Delete	Add SSH public key
<input type="checkbox"/>	Date imported	Fingerprint	< 1 >
<input type="checkbox"/>	6/14/2022, 12:53:34 PM	SHA256-	
<input type="checkbox"/>	10/20/2022, 4:26:51 PM	SHA256-	

Creating SSH keys on Microsoft Windows

Windows uses a slightly different SSH key pair format. The public key must be in the PUB format, and the private key must be in the PPK format. On Windows, you can use PuTTYgen to create an SSH key pair in the appropriate formats. You can also use PuTTYgen to convert a private key generated using `ssh-keygen` to a `.ppk` file.

Note

If you present WinSCP with a private key file not in `.ppk` format, that client offers to convert the key into `.ppk` format for you.

For a tutorial about creating SSH keys by using PuTTYgen on Windows, see the [SSH.com website](https://www.ssh.com).

Convert an SSH2 public key to PEM format

AWS Transfer Family only accepts PEM formatted public keys. If you have an SSH2 public key, you need to convert it. An SSH2 public key has the following format:

```
----- BEGIN SSH2 PUBLIC KEY -----
Comment: "rsa-key-20160402"
AAAAB3NzaC1yc2EAAAABJQAAAQEaIL0jjDdFqK/kYThqKt7THrjABTPWvXmB3URI
:
:
----- END SSH2 PUBLIC KEY -----
```

A PEM public key has the following format:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAA...
```

Run the following command to convert an SSH2-formatted public key into a PEM-formatted public key. Replace *ssh2-key* with the name of your SSH2 key, and *PEM-key* with the name of your PEM key.

```
ssh-keygen -i -f ssh2-key.pub > PEM-key.pub
```

Rotate SSH keys

For security, we recommend the best practice of rotating your SSH keys. Usually, this rotation is specified as a part of a security policy and is implemented in some automated fashion. Depending upon the level of security, for a highly sensitive communication, an SSH key pair might be used only once. Doing this eliminates any risk due to stored keys. However, it is much more common to store SSH credentials for a period of time and set an interval that doesn't place undue burden on users. A time interval of three months is common.

There are two methods used to perform SSH key rotation:

- On the console, you can upload a new SSH public key and delete an existing SSH public key.
- Using the API, you can update existing users by using the [DeleteSshPublicKey](#) API to delete a user's Secure Shell (SSH) public key and the [ImportSshPublicKey](#) API to add a new Secure Shell (SSH) public key to the user's account.

Console

To perform a key rotation in the console

1. Open the AWS Transfer Family console at <https://console.aws.amazon.com/transfer/>.
2. Navigate to the **Servers** page.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.
4. Under **Users**, select the check box of the user whose SSH public key that you want to rotate, then choose **Actions**, and then choose **Add key** to see the **Add key** page.

or

Choose the username to see the **User details** page, and then choose **Add SSH public key** to see the **Add key** page.

5. Enter the new SSH public key and choose **Add key**.

⚠ Important

The format of the SSH public key depends on the type of key you generated.

- For RSA keys, the format is `ssh-rsa string`.
- For ED25519 keys, the format is `ssh-ed25519 string`.
- For ECDSA keys, the key begins with `ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp384`, or `ecdsa-sha2-nistp521`, depending on the size of the key you generated. The beginning string is then followed by *string*, similar to the other key types.

You are returned to the **User details** page, and the new SSH public key that you just entered appears in the **SSH public keys** section.

6. Select the check box of the old you key that you want to delete and then choose **Delete**.
7. Confirm the deletion operation by entering the word `delete`, and then choose **Delete**.

API**To perform a key rotation using the API**

1. On macOS, Linux, or Unix operating systems, open a command terminal.
2. Retrieve the SSH key that you want to delete by entering the following command. To use this command, replace *serverID* with the server ID for your Transfer Family server, and replace *username* with your username.

```
aws transfer describe-user --server-id='serverID' --user-name='username'
```

The command returns details about the user. Copy the contents of the `"SshPublicKeyId"` field. You will need to enter this value later in this procedure.

```
"SshPublicKeys": [ { "SshPublicKeyBody": "public-key", "SshPublicKeyId":  
  "keyID",  
  "DateImported": 1621969331.072 } ],
```

- Next, import a new SSH key for your user. At the prompt, enter the following command. To use this command, replace *serverID* with the server ID for your Transfer Family server, replace *username* with your username, and replace *public-key* with the fingerprint of your new public key.

```
aws transfer import-ssh-public-key --server-id='serverID' --user-name='username'
--ssh-public-key-body='public-key'
```

If the command is successful, no output is returned.

- Finally, delete the old key by running the following command. To use this command, replace *serverID* with the server ID for your Transfer Family server, replace *username* with your username, and replace *keyID-from-step-2* with the key ID value that you copied in step 2 of this procedure

```
aws transfer delete-ssh-public-key --server-id='serverID' --user-name='username'
--ssh-public-key-id='keyID-from-step-2'
```

- (Optional) To confirm that the old key no longer exists, repeat step 2.

Generate and manage PGP keys

You can use Pretty Good Privacy (PGP) decryption with the files that Transfer Family processes with workflows. To use decryption in a workflow step, provide a PGP key.

The AWS storage blog has a post that describes how to simply decrypt files without writing any code using Transfer Family Managed workflows, [Encrypt and decrypt files with PGP and AWS Transfer Family](#).

Generate PGP keys

The operator that you use to generate your PGP keys depends on your operating system and the version of the key-generation software that you're using.

If you're using Linux or Unix, use your package installer to install gpg. Depending on your Linux distribution, one of the following commands should work for you.

```
sudo yum install gnupg
```

```
sudo apt-get install gnupg
```

For Windows or macOS, you can download what you need from <https://gnupg.org/download/>.

After you install your PGP key generator software, you run the `gpg --full-gen-key` or `gpg --gen-key` command to generate a key pair.

Note

If you're using GnuPG version 2.3.0 or newer, you must run `gpg --full-gen-key`. When prompted for the type of key to create, choose RSA or ECC. However, if you choose ECC, make sure to choose either NIST or BrainPool for the elliptic curve. **Do not** choose Curve 25519.

Algorithms supported for PGP key-pairs

We support the following algorithms for PGP key pairs:

- RSA
- Elgamal
- ECC:
 - NIST
 - BrainPool

Note

We don't support cCurve25519 keys.

Useful gpg subcommands

The following are some useful subcommands for `gpg`:

- `gpg --help` – This command lists the available options and might include some examples.
- `gpg --list-keys` – This command lists the details for all the key pairs that you have created.
- `gpg --fingerprint` – This command lists the details for all your key pairs, including each key's fingerprint.

- `gpg --export -a user-name` – This command exports the public key portion of the key for the *user-name* that was used when the key was generated.

Manage PGP keys

To manage your PGP keys, use AWS Secrets Manager.

Note

Your secret name includes your Transfer Family server ID. This means you should have already identified or created a server *before* you can store your PGP key information in AWS Secrets Manager.

If you want to use one key and passphrase for all of your users, you can store the PGP key block information under the secret name `aws/transfer/server-id@pgp-default`, where *server-id* is the ID for your Transfer Family server. Transfer Family uses this default key if there is no key where the *user-name* matches the user that's executing the workflow.

You can create a key for a specific user. In this case, the format for the secret name is `aws/transfer/server-id/user-name`, where *user-name* matches the user that's running the workflow for a Transfer Family server.

Note

You can store a maximum of 3 PGP private keys, per Transfer Family server, per user.

To configure PGP keys for use with decryption

1. Depending on the version of GPG that you are using, run one of the following commands to generate a PGP key pair that doesn't use a Curve 25519 encryption algorithm.
 - If you are using **GnuPG** version 2.3.0 or newer, run the following command:

```
gpg --full-gen-key
```

You can choose **RSA**, or, if you choose **ECC**, you can choose either **NIST** or **BrainPool** for the elliptic curve. If you run `gpg --gen-key` instead, you create a key pair that uses the ECC Curve 25519 encryption algorithm, which we don't currently support for PGP keys.

- For versions of **GnuPG** prior to 2.3.0, you can use the following command, since RSA is the default encryption type.

```
gpg --gen-key
```

Important

During the key-generation process, you must provide a passphrase and an email address. Make sure to take note of these values. You must provide the passphrase when you enter the key's details into AWS Secrets Manager later in this procedure. And you must provide the same email address to export the private key in the next step.

2. Run the following command to export the private key. To use this command, replace *private.pgp* with the name of the file in which to save the private key block, and *marymajor@example.com* with the email address that you used when you generated the key pair.

```
gpg --output private.pgp --armor --export-secret-key marymajor@example.com
```

3. Use AWS Secrets Manager to store your PGP key.
 - a. Sign in to the AWS Management Console and open the AWS Secrets Manager console at <https://console.aws.amazon.com/secretsmanager/>.
 - b. In the left navigation pane, choose **Secrets**.
 - c. On the **Secrets** page, choose **Store a new secret**.
 - d. On the **Choose secret type** page, for **Secret type**, select **Other type of secret**.
 - e. In the **Key/value pairs** section, choose the **Key/value** tab.
 - **Key** – Enter **PGPPrivateKey**.

Note

You must enter the **PGPprivateKey** string exactly: do not add any spaces before or between characters.

- **value** – Paste the text of your private key into the value field. You can find the text of your private key in the file (for example, `private.pgp`) that you specified when you exported your key earlier in this procedure. The key begins with `-----BEGIN PGP PRIVATE KEY BLOCK-----` and ends with `-----END PGP PRIVATE KEY BLOCK-----`.

Note

Make sure that the text block contains only the private key and does not contain the public key as well.

f. Select **Add row** and in the **Key/value pairs** section, choose the **Key/value** tab.

- **Key** – Enter **PGPPassphrase**.

Note

You must enter the **PGPPassphrase** string exactly: do not add any spaces before or between characters.

- **value** – Enter the passphrase you used when you generated your PGP key pair.

Choose secret type

Secret type [Info](#)

Credentials for Amazon RDS database

Credentials for Amazon DocumentDB database

Credentials for Amazon Redshift cluster

Credentials for other database

Other type of secret
API key, OAuth token, other.

Key/value pairs [Info](#)

Key/value

Plaintext

PGPrivateKey	-----BEGIN PGP PRIVATE KEY BLOCK-----	Remove
PGPassphrase	mypassphrase	Remove

+ Add row

Encryption key [Info](#)

You can encrypt using the KMS key that Secrets Manager creates or a customer managed KMS key that you create.

aws/secretsmanager

▼

↻

[Add new key](#)

Note

You can add up to 3 sets of keys and passphrases. To add a second set, add two new rows, and enter **PGPrivateKey2** and **PGPassphrase2** for the keys, and paste in another private key and passphrase. To add a third set, key values must be **PGPrivateKey3** and **PGPassphrase3**.

- g. Choose **Next**.
- h. On the **Configure secret** page, enter a name and description for your secret.
 - If you're creating a default key, that is, a key that can be used by any Transfer Family user, enter **aws/transfer/*server-id*@pgp-default**. Replace *server-id* with the ID of the server that contains the workflow that has a decrypt step.
 - If you're creating a key to be used by a specific Transfer Family user, enter **aws/transfer/*server-id*/*user-name***. Replace *server-id* with the ID of the server that contains the workflow that has a decrypt step, and replace *user-name* with the name of the user that's running the workflow. The *user-name* is stored in the identity provider that the Transfer Family server is using.

- i. Choose **Next** and accept the defaults on the **Configure rotation** page. Then choose **Next**.
- j. On the **Review** page, choose **Store** to create and store the secret.

The following screenshot shows the details for the user **marymajor** for a specific Transfer Family server. This example shows three keys and their corresponding passphrases.

The screenshot shows the AWS Secrets Manager console for a secret named `/aws/transfer/s-.../marymajor`. The **Secret details** section includes:

- Encryption key:** `aws/secretsmanager`
- Secret name:** `/aws/transfer/s-.../marymajor`
- Secret ARN:** `arn:aws:secretsmanager:us-east-2:...:secret:/aws/transfer/s-.../marymajor-...`
- Secret description:** Contains the PGP secret keys and corresponding passphrases to use for user marymajor on Transfer Family server s-...

The **Secret value** section is set to **Plaintext** and displays a table of secret values:

Secret key	Secret value
PGPPrivateKey	-----BEGIN PGP PRIVATE KEY BLOCK----- [redacted]
PGPPassphrase	mypassphrase
PGPPrivateKey2	-----BEGIN PGP PRIVATE KEY BLOCK----- [redacted]
PGPPassphrase2	mypassphrase2
PGPPrivateKey3	-----BEGIN PGP PRIVATE KEY BLOCK----- [redacted]
PGPPassphrase3	mypassphrase3

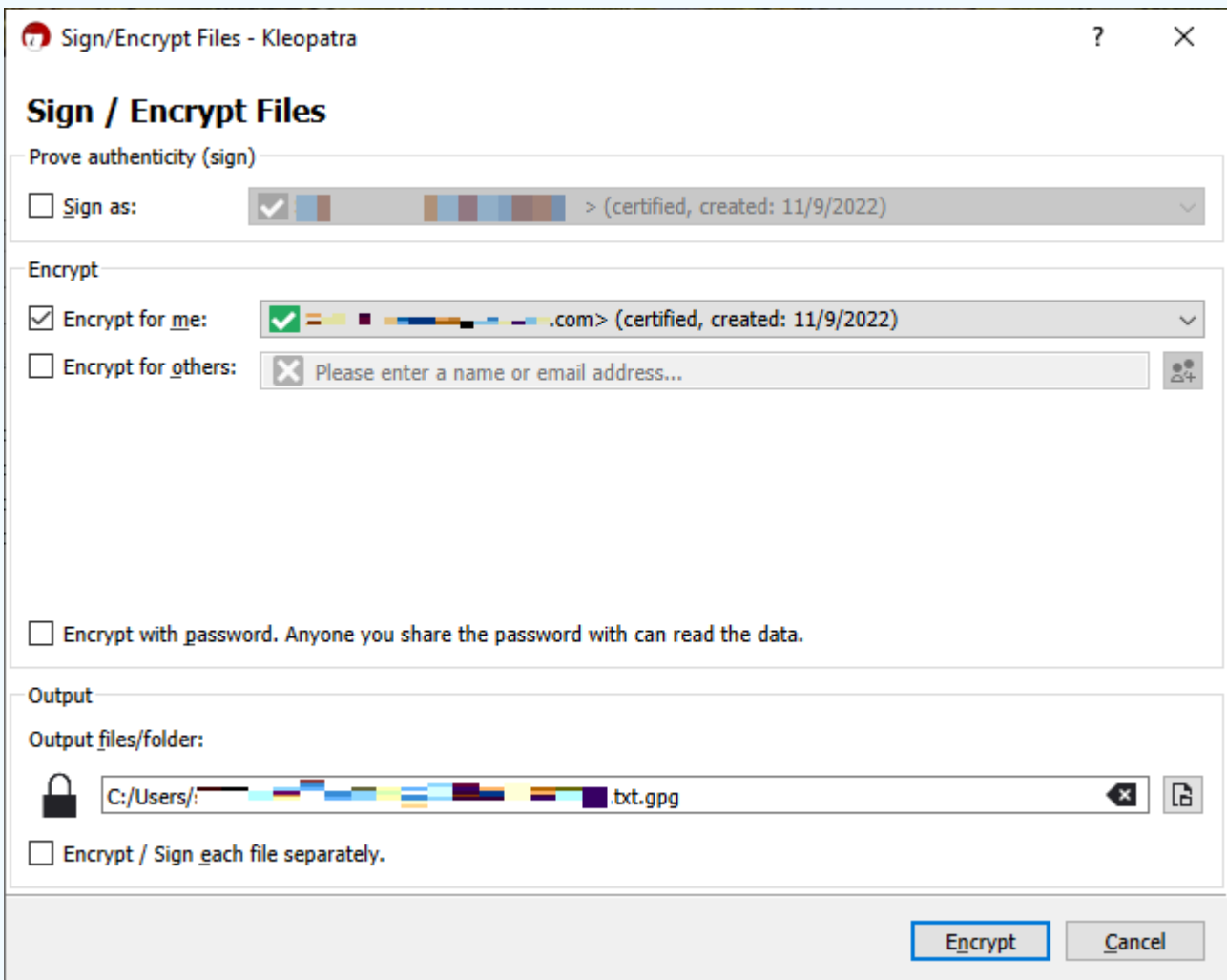
Supported PGP clients

The following clients have been tested with Transfer Family and can be used to generate PGP keys, and to encrypt files that you intend to decrypt with a workflow.

- **Gpg4win + Kleopatra.**

Note

When you select **Sign / Encrypt Files**, make sure to clear the selection for **Sign as**: we do not currently support signing for encrypted files.



If you sign the encrypted file and attempt to upload it to a Transfer Family server with a decryption workflow, you receive the following error:

Encrypted file with signed message unsupported

- Major **GnuPG** versions: 2.4, 2.3, 2.2, 2.0, and 1.4.

Note that other PGP clients might work as well, but only the clients mentioned here have been tested with Transfer Family.

Identity and access management for AWS Transfer Family

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in)

and *authorized* (have permissions) to use AWS Transfer Family resources. IAM is an AWS service that you can use with no additional charge.

Topics

- [Audience](#)
- [Authenticating with identities](#)
- [Managing access using policies](#)
- [How AWS Transfer Family works with IAM](#)
- [AWS Transfer Family identity-based policy examples](#)
- [AWS Transfer Family tag-based policy examples](#)
- [Troubleshooting AWS Transfer Family identity and access](#)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS Transfer Family.

Service user – If you use the AWS Transfer Family service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS Transfer Family features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS Transfer Family, see [Troubleshooting AWS Transfer Family identity and access](#).

Service administrator – If you're in charge of AWS Transfer Family resources at your company, you probably have full access to AWS Transfer Family. It's your job to determine which AWS Transfer Family features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS Transfer Family, see [How AWS Transfer Family works with IAM](#).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS Transfer Family. To view example AWS Transfer Family identity-based policies that you can use in IAM, see [AWS Transfer Family identity-based policy examples](#).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see [How to sign in to your AWS account](#) in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see [Signing AWS API requests](#) in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see [Multi-factor authentication](#) in the *AWS IAM Identity Center User Guide* and [Using multi-factor authentication \(MFA\) in AWS](#) in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see [Tasks that require root user credentials](#) in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see [What is IAM Identity Center?](#) in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An [IAM user](#) is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see [Rotate access keys regularly for use cases that require long-term credentials](#) in the *IAM User Guide*.

An [IAM group](#) is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [When to create an IAM user \(instead of a role\)](#) in the *IAM User Guide*.

IAM roles

An [IAM role](#) is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by [switching roles](#). You can assume a role by calling an AWS CLI or

AWS API operation or by using a custom URL. For more information about methods for using roles, see [Using IAM roles](#) in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see [Creating a role for a third-party Identity Provider](#) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see [Permission sets](#) in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [Cross account resource access in IAM](#) in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
- **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see [Forward access sessions](#).
- **Service role** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see [When to create an IAM role \(instead of a user\)](#) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see [Overview of JSON policies](#) in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM policies](#) in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choosing between managed policies and inline policies](#) in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list \(ACL\) overview](#) in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [How SCPs work](#) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see [Session policies](#) in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy evaluation logic](#) in the *IAM User Guide*.

How AWS Transfer Family works with IAM

Before you use AWS Identity and Access Management (IAM) to manage access to AWS Transfer Family, you should understand what IAM features are available to use with AWS Transfer Family. To get a high-level view of how AWS Transfer Family and other AWS services work with IAM, see [AWS services that work with IAM](#) in the *IAM User Guide*.

Topics

- [AWS Transfer Family identity-based policies](#)
- [AWS Transfer Family resource-based policies](#)
- [Authorization based on AWS Transfer Family tags](#)
- [AWS Transfer Family IAM roles](#)

AWS Transfer Family identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. AWS Transfer Family supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see [IAM JSON policy elements reference](#) in the *AWS Identity and Access Management User Guide*.

Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in AWS Transfer Family use the following prefix before the action: `transfer:`. For example, to grant someone permission to create a server, with the Transfer Family `CreateServer` API operation, you include the `transfer:CreateServer` action in their policy. Policy statements must include either an `Action` or `NotAction` element. AWS Transfer Family defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows.

```
"Action": [  
    "transfer:action1",
```

```
"transfer:action2"
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word Describe, include the following action.

```
"Action": "transfer:Describe*"
```

To see a list of AWS Transfer Family actions, see [Actions defined by AWS Transfer Family](#) in the *Service Authorization Reference*.

Resources

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its [Amazon Resource Name \(ARN\)](#). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

The Transfer Family server resource has the following ARN.

```
arn:aws:transfer:${Region}:${Account}:server/${ServerId}
```

For example, to specify the s-01234567890abcdef Transfer Family server in your statement, use the following ARN.

```
"Resource": "arn:aws:transfer:us-east-1:123456789012:server/s-01234567890abcdef"
```

For more information about the format of ARNs, see [Amazon Resource Names \(ARNs\)](#) in the *Service Authorization Reference*, or [IAM ARNs](#) in the *IAM User Guide*.

To specify all instances that belong to a specific account, use the wildcard (*).

```
"Resource": "arn:aws:transfer:us-east-1:123456789012:server/*"
```

Some AWS Transfer Family actions are performed on multiple resources, such as those used in IAM policies. In those cases, you must use the wildcard (*).

```
"Resource": "arn:aws:transfer:*:123456789012:server/*"
```

In some cases you need to specify more than one type of resource, for example, if you create a policy that allows access to Transfer Family servers and users. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [  
    "resource1",  
    "resource2"  
]
```

To see a list of AWS Transfer Family resources, see [Resource types defined by AWS Transfer Family](#) in the *Service Authorization Reference*.

Condition keys

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or *Condition block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use [condition operators](#), such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM policy elements: variables and tags](#) in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see [AWS global condition context keys](#) in the *IAM User Guide*.

AWS Transfer Family defines its own set of condition keys and also supports using some global condition keys. To see a list of AWS Transfer Family condition keys, see [Condition keys for AWS Transfer Family](#) in the *Service Authorization Reference*.

Examples

To view examples of AWS Transfer Family identity-based policies, see [AWS Transfer Family identity-based policy examples](#).

AWS Transfer Family resource-based policies

Resource-based policies are JSON policy documents that specify what actions a specified principal can perform on the AWS Transfer Family resource and under what conditions. Amazon S3 supports resource-based permissions policies for Amazon S3 *buckets*. Resource-based policies let you grant usage permission to other accounts on a per-resource basis. You can also use a resource-based policy to allow an AWS service to access your Amazon S3 *buckets*.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the [principal in a resource-based policy](#). Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, you must also grant the principal entity permission to access the resource. Grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see [How IAM roles differ from resource-based policies](#) in the *AWS Identity and Access Management User Guide*.

The Amazon S3 service supports only one type of resource-based policy called a *bucket policy*, which is attached to a *bucket*. This policy defines which principal entities (accounts, users, roles, and federated users) can perform actions on the object.

Examples

To view examples of AWS Transfer Family resource-based policies, see [AWS Transfer Family tag-based policy examples](#).

Authorization based on AWS Transfer Family tags

You can attach tags to AWS Transfer Family resources or pass tags in a request to AWS Transfer Family. To control access based on tags, you provide tag information in the [condition element](#) of a policy using the `transfer:ResourceTag/key-name`, `aws:RequestTag/key-name`, or

`aws` : TagKeys condition keys. For information about how to use tags to control access to AWS Transfer Family resources, see [AWS Transfer Family tag-based policy examples](#).

AWS Transfer Family IAM roles

An [IAM role](#) is an entity within your AWS account that has specific permissions.

Using temporary credentials with AWS Transfer Family

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as [AssumeRole](#) or [GetFederationToken](#).

AWS Transfer Family supports using temporary credentials.

AWS Transfer Family identity-based policy examples

By default, IAM users and roles don't have permission to create or modify AWS Transfer Family resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see [Creating policies on the JSON tab](#) in the *AWS Identity and Access Management User Guide*.

Topics

- [Policy best practices](#)
- [Using the AWS Transfer Family console](#)
- [Allow users to view their own permissions](#)

Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS Transfer Family resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies*

that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see [AWS managed policies](#) or [AWS managed policies for job functions](#) in the *IAM User Guide*.

- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see [Policies and permissions in IAM](#) in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see [IAM Access Analyzer policy validation](#) in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see [Configuring MFA-protected API access](#) in the *IAM User Guide*.

For more information about best practices in IAM, see [Security best practices in IAM](#) in the *IAM User Guide*.

Using the AWS Transfer Family console

To access the AWS Transfer Family console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the AWS Transfer Family resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM users or roles) with that policy. For more information, see [Adding permissions to a user](#) in the *AWS Identity and Access Management User Guide*.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS Transfer Family tag-based policy examples

The following are examples of how to control access to AWS Transfer Family resources based on tags.

Using tags to control access to AWS Transfer Family resources

Conditions in IAM policies are part of the syntax that you use to specify permissions to AWS Transfer Family resources. You can control access to AWS Transfer Family resources (such as users, servers, roles, and other entities) based on tags on those resources. Tags are key-value pairs. For more information about tagging resources, see [Tagging AWS resources](#) in the *AWS General Reference*.

In AWS Transfer Family, resources can have tags, and some actions can include tags. When you create an IAM policy, you can use tag condition keys to control the following:

- Which users can perform actions on an AWS Transfer Family resource, based on tags that the resource has.
- What tags can be passed in an action's request.
- Whether specific tag keys can be used in a request.

By using tag-based access control, you can apply finer control than at the API level. You also can apply more dynamic control than by using resource-based access control. You can create IAM policies that allow or deny an operation based on tags provided in the request (request tags). You can also create IAM policies based on tags on the resource that is being operated on (resource tags). In general, resource tags are for tags that are already on resources, request tags are for when you're adding tags to or removing tags from a resource.

For the complete syntax and semantics of tag condition keys, see [Controlling access to AWS resources using resource tags](#) in the *IAM User Guide*. For details about specifying IAM policies with API Gateway, see [Control access to an API with IAM permissions](#) in the *API Gateway Developer Guide*.

Example 1: Deny actions based on resource tags

You can deny an action to be performed on a resource based on tags. The following example policy denies `TagResource`, `UntagResource`, `StartServer`, `StopServer`, `DescribeServer`, and `DescribeUser` operations if the user or server resource is tagged with the key `stage` and the value `prod`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "transfer:TagResource",
        "transfer:UntagResource",
        "transfer:StartServer",
        "transfer:StopServer",
        "transfer:DescribeServer",
        "transfer:DescribeUser"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/stage": "prod"
        }
      }
    }
  ]
}
```

Example 2: Allow actions based on resource tags

You can allow an action to be performed on a resource based on tags. The following example policy allows TagResource, UntagResource, StartServer, StopServer, DescribeServer, and DescribeUser operations if the user or server resource is tagged with the key stage and the value prod.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "transfer:TagResource",
        "transfer:UntagResource",
        "transfer:StartServer",
        "transfer:StopServer",
        "transfer:DescribeServer",
        "transfer:DescribeUser"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/stage": "prod"
      }
    }
  }
]
}

```

Example 3: Deny creation of a user or server based on request tags

The following example policy contains two statements. The first statement denies the `CreateServer` operation on all resources if the cost center key for the tag doesn't have a value.

The second statement denies the `CreateServer` operation if the cost center key for the tag contains any other value besides 1, 2 or 3.

Note

This policy does allow creating or deleting a resource that contains a key called `costcenter` and a value of 1, 2, or 3.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "transfer:CreateServer"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "Null": {
          "aws:RequestTag/costcenter": "true"
        }
      }
    }
  ],
}

```

```
{
  "Effect": "Deny",
  "Action": "transfer:CreateServer",
  "Resource": [
    "*"
  ],
  "Condition": {
    "ForAnyValue:StringNotEquals": {
      "aws:RequestTag/costcenter": [
        "1",
        "2",
        "3"
      ]
    }
  }
}
```

Troubleshooting AWS Transfer Family identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Transfer Family and IAM.

Topics

- [I am not authorized to perform an action in AWS Transfer Family](#)
- [I am not authorized to perform iam:PassRole](#)
- [I want to allow people outside of my AWS account to access my AWS Transfer Family resources](#)

I am not authorized to perform an action in AWS Transfer Family

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your sign-in credentials.

The following example error occurs when the mateojackson IAM user tries to use the console to view details about a *widget* but does not have `transfer:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
transfer:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *my-example-widget* resource using the `transfer;:GetWidget` action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to AWS Transfer Family.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS Transfer Family. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

The following example policy contains the permission to pass a role to AWS Transfer Family.

```
{
  "Version": "2012-10-17",
  "Statement": [
    { "Action": "iam:PassRole",
      "Resource": "arn:aws::iam::123456789012:role/*",
      "Effect": "Allow"
    }
  ]
}
```

I want to allow people outside of my AWS account to access my AWS Transfer Family resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS Transfer Family supports these features, see [How AWS Transfer Family works with IAM](#).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing access to an IAM user in another AWS account that you own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing access to AWS accounts owned by third parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing access to externally authenticated users \(identity federation\)](#) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

Compliance validation for AWS Transfer Family

Third-party auditors assess the security and compliance of AWS Transfer Family as part of multiple AWS compliance programs. These include SOC, PCI, HIPAA, and others. For the complete list, see [AWS Services in Scope by Compliance Program](#).

For a list of AWS services in scope of specific compliance programs, see [AWS services in scope by compliance program](#). For general information, see [AWS compliance programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading reports in AWS Artifact](#).

Your compliance responsibility when using AWS Transfer Family is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and compliance quick start guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA security and compliance whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS compliance resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Config](#) – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in AWS Transfer Family

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

AWS Transfer Family supports up to 3 Availability Zones and is backed by an auto scaling, redundant fleet for your connection and transfer requests.

Note the following:

- For public endpoints:
 - Availability Zone-level redundancy is built into the service
 - There are redundant fleets for each AZ.
 - This redundancy is provided automatically
- For endpoints in a Virtual Private Cloud (VPC), see [Create a server in a virtual private cloud](#).

See also

- For more information about AWS Regions and Availability Zones, see [AWS global infrastructure](#).

- For an example on how to build for higher redundancy and minimize network latency by using Latency-based routing, see the blog post [Minimize network latency with your AWS Transfer Family servers](#).

Infrastructure security in AWS Transfer Family

As a managed service, AWS Transfer Family is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see [AWS Cloud Security](#). To design your AWS environment using the best practices for infrastructure security, see [Infrastructure Protection](#) in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access AWS Transfer Family through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

Add a web application firewall

AWS WAF is a web application firewall that helps protect web applications and APIs from attacks. You can use it to configure a set of rules known as a *web access control list* (web ACL) that allow, block, or count web requests based on customizable web security rules and conditions that you define. For more information, see [Using AWS WAF to protect your APIs](#).

To add AWS WAF

1. Open the API Gateway console at <https://console.aws.amazon.com/apigateway/>.
2. In the **APIs** navigation pane, and then choose your custom identity provider template.
3. Choose **Stages**.
4. In the **Stages** pane, choose the name of the stage.

5. In the **Stage Editor** pane, choose the **Settings** tab.
6. Do one of the following:
 - Under **Web Application Firewall (WAF)**, for **Web ACL**, choose the web ACL that you want to associate with this stage.
 - If the web ACL you need doesn't exist, you will need to create one by doing the following:
 1. Choose **Create Web ACL**.
 2. On the AWS WAF service homepage, choose **Create web ACL**.
 3. In **Web ACL details**, for **Name**, type the name of the web ACL.
 4. In **Rules**, choose **Add rules**, then choose **Add my own rules and rule groups**.
 5. For **Rule type**, choose IP set to identify a specific list of IP addresses.
 6. For **Rule**, enter the name of the rule.
 7. For **IP set**, choose an existing IP set. To create an IP set, see [Creating an IP set](#).
 8. For **IP address to use as the originating address**, choose **IP address in header**.
 9. For **Header field name**, enter `SourceIP`.
 - 10 For **Position inside header**, choose **First IP address**.
 - 11 For **Fallback for missing IP address**, choose **Match** or **No Match** depending on how you want to handle an invalid (or missing) IP address in the header.
 - 12 For **Action**, choose the action of the IP set.
 - 13 For **Default web ACL action for requests that don't match any rules**, choose **Allow** or **Block** and then click **Next**.
 - 14 For steps 4 and 5, choose **Next**.
 - 15 In **Review and create**, review your choices, and then choose **Create web ACL**.
7. Choose **Save Changes**.
8. Choose **Resources**.
9. For **Actions**, choose **Deploy API**.

For information on how secure AWS Transfer Family with AWS web application firewall, see [Securing AWS Transfer Family with AWS application firewall and Amazon API Gateway](#) in the AWS storage blog.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way that it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account. For a detailed description of this problem, see [the confused deputy problem](#) in the *IAM User Guide*.

We recommend using the [aws:SourceArn](#) and [aws:SourceAccount](#) global condition context keys in resource policies to limit the permissions that AWS Transfer Family has for the resource. If you use both global condition context keys, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement.

The most effective way to protect against the confused deputy problem is to use the exact Amazon Resource Name (ARN) of the resource you want to allow. If you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcard characters (*) for the unknown portions of the ARN. For example, `arn:aws:transfer::region::account-id:server/*`.

AWS Transfer Family uses the following types of roles:

- **User role** – Allows service-managed users to access the necessary Transfer Family resources. AWS Transfer Family assumes this role in the context of a Transfer Family user ARN.
- **Access role** – Provides access to only the Amazon S3 files that are being transferred. For inbound AS2 transfers, the access role uses the Amazon Resource Name (ARN) for the agreement. For outbound AS2 transfers, the access role uses the ARN for the connector.
- **Invocation role** – For use with Amazon API Gateway as the server's custom identity provider. Transfer Family assumes this role in the context of a Transfer Family server ARN.
- **Logging role** – Used to log entries into Amazon CloudWatch. Transfer Family uses this role to log success and failure details along with information about file transfers. Transfer Family assumes this role in the context of a Transfer Family server ARN. For outbound AS2 transfers, the logging role uses the connector ARN.

- **Execution role** – Allows a Transfer Family user to call and launch workflows. Transfer Family assumes this role in the context of a Transfer Family workflow ARN.

For more information, see [Policies and permissions in IAM](#) in the *IAM User Guide*.

Note

In the following examples, replace each *user input placeholder* with your own information.

Note

In our examples, we use both `ArnLike` and `ArnEquals`. They are functionally identical, and therefore you may use either when you construct your policies. Transfer Family documentation uses `ArnLike` when the condition contains a wildcard character, and `ArnEquals` to indicate an exact match condition.

AWS Transfer Family user role cross-service confused deputy prevention

The following example policy allows any user of any server in the account to assume the role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        },
        "ArnLike": {
```

```

        "aws:SourceArn": "arn:aws:transfer:region:account-id:user/*"
    }
}

```

The following example policy allows any user of a specific server to assume the role.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:transfer:region:account-id:user/server-
id/*"
        }
      }
    }
  ]
}

```

The following example policy allows a specific user of a specific server to assume the role.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      },

```

```

        "Action": "sts:AssumeRole",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": "arn:aws:transfer:region:account-id:user/server-
id/user-name"
            }
        }
    ]
}

```

AWS Transfer Family workflow role cross-service confused deputy prevention

The following example policy allows any workflow in the account to assume the role.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:transfer:region:account-id:workflow/*"
        }
      }
    }
  ]
}

```

The following example policy allows a specific workflow to assume the role.

```

{
  "Version": "2012-10-17",

```

```

    "Statement": [
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
          "Service": "transfer.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "ArnLike": {
            "aws:SourceArn": "arn:aws:transfer:region:account-
id:workflow/workflow-id"
          }
        }
      }
    ]
  }
}

```

AWS Transfer Family logging and invocation role cross-service confused deputy prevention

Note

The following examples can be used in both logging and invocation roles. In these examples, you can remove the ARN details for a workflow if your server doesn't have any workflows attached to it.

The following example logging/invoke policy allows any server (and workflow) in the account to assume the role.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllServersWithWorkflowAttached",
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      },
      "Action": "sts:AssumeRole",

```



```

    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "account-id"
      },
      "ArnLike": {
        "aws:SourceArn": [
          "arn:aws:transfer:region:account-id:server/*",
          "arn:aws:transfer:region:account-id:workflow/*"
        ]
      }
    }
  }
]
}

```

The following example logging/invocation policy allows a specific server (and workflow) to assume the role.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSpecificServerWithWorkflowAttached",
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        },
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:transfer:region:account-id:server/server-id",
            "arn:aws:transfer:region:account-id:workflow/workflow-id"
          ]
        }
      }
    }
  ]
}

```

AWS managed policies for AWS Transfer Family

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to [create AWS Identity and Access Management \(IAM\) customer managed policies](#) that provide your team with only the permissions that they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see [AWS managed policies](#) in the *IAM User Guide*. For a detailed listing of all AWS managed policies, see the [AWS managed policy reference guide](#).

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the `ReadOnlyAccess` AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see [AWS managed policies for job functions](#) in the *IAM User Guide*.

AWS managed policy: `AWSTransferConsoleFullAccess`

The `AWSTransferConsoleFullAccess` policy provides full access to Transfer Family through the AWS Management Console.

Permissions details

This policy includes the following permissions.

- `acm:ListCertificates` – Grants permission to retrieve a list of the certificate Amazon Resource Names (ARNs) and the domain name for each ARN.
- `ec2:DescribeAddresses` – Grants permission to describe one or more Elastic IP addresses.
- `ec2:DescribeAvailabilityZones` – Grants permission to describe one or more of the Availability Zones that are available to you.
- `ec2:DescribeNetworkInterfaces` – Grants permission to describe one or more elastic network interfaces.

- `ec2:DescribeSecurityGroups` – Grants permission to describe one or more security groups.
- `ec2:DescribeSubnets` – Grants permission to describe one or more subnets.
- `ec2:DescribeVpcs` – Grants permission to describe one or more virtual private clouds (VPCs).
- `ec2:DescribeVpcEndpoints` – Grants permission to describe one or more VPC endpoints.
- `health:DescribeEventAggregates` – Returns the number of events of each event type (issue, scheduled change, and account notification).
- `iam:GetPolicyVersion` – Grants permission to retrieve information about a version of the specified managed policy, including the policy document.
- `iam:ListPolicies` – Grants permission to list all managed policies.
- `iam:ListRoles` – Grants permission to list the IAM roles that have the specified path prefix.
- `iam:PassRole` – Grants permission to pass an IAM role to Transfer Family. For more details, see [Granting a user permissions to pass a role to an AWS service](#).
- `route53:ListHostedZones` – Grants permission to get a list of the public and private hosted zones that are associated with the current AWS account.
- `s3:ListAllMyBuckets` – Grants permission to list all buckets owned by the authenticated sender of the request.
- `transfer:*` – Grants access to Transfer Family resources. The asterisk (*) grants access to all resources.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "transfer.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "acm:ListCertificates",
```

```

        "ec2:DescribeAddresses",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcEndpoints",
        "health:DescribeEventAggregates",
        "iam:GetPolicyVersion",
        "iam:ListPolicies",
        "iam:ListRoles",
        "route53:ListHostedZones",
        "s3:ListAllMyBuckets",
        "transfer:*"
    ],
    "Resource": "*"
}
]
}

```

AWS managed policy: AWSTransferFullAccess

The `AWSTransferFullAccess` policy provides full access to Transfer Family services.

Permissions details

This policy includes the following permissions.

- `transfer:*` – Grants permission to access Transfer Family resources. The asterisk (*) grants access to all resources.
- `iam:PassRole` – Grants permission to pass an IAM role to Transfer Family. For more details, see [Granting a user permissions to pass a role to an AWS service](#).
- `ec2:DescribeAddresses` – Grants permission to describe one or more Elastic IP addresses.
- `ec2:DescribeNetworkInterfaces` – Grants permission to describe one or more network interfaces.
- `ec2:DescribeVpcEndpoints` – Grants permission to describe one or more VPC endpoints.

```

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

```

```
{
  "Effect": "Allow",
  "Action": "transfer:*",
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": "iam:PassRole",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "iam:PassedToService": "transfer.amazonaws.com"
    }
  }
},
{
  "Effect": "Allow",
  "Action": [
    "ec2:DescribeVpcEndpoints",
    "ec2:DescribeNetworkInterfaces",
    "ec2:DescribeAddresses"
  ],
  "Resource": "*"
}
]
```

AWS managed policy: AWSTransferLoggingAccess

The AWSTransferLoggingAccess policy grants AWS Transfer Family full access to create log streams and groups and put log events to your account.

Permissions details

This policy includes the following permissions for Amazon CloudWatch Logs.

- **CreateLogStream** – Grants permissions for principals to create a log stream.
- **DescribeLogStreams** – Grants permissions for principals to list the log streams for the log group.
- **CreateLogGroup** – Grants permissions for principals to create log groups.
- **PutLogEvents** – Grants permissions for principals to upload a batch of log events to a log stream.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:CreateLogGroup",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS managed policy: AWSTransferReadOnlyAccess

The AWSTransferReadOnlyAccess policy provides read-only access to Transfer Family services.

Permissions details

This policy includes the following permissions for Transfer Family.

- `DescribeUser` – Grants permissions for principals to view the descriptions for users.
- `DescribeServer` – Grants permissions for principals to view the descriptions for servers.
- `ListUsers` – Grants permissions for principals to list users for a server.
- `ListServers` – Grants permissions for principals to list the servers for the account.
- `TestIdentityProvider` – Grants permissions for principals to test whether the configured identity provider is set up correctly.
- `ListTagsForResource` – Grants permissions for principals to list the tags for a resource.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
```

```

    "Action": [
      "transfer:DescribeUser",
      "transfer:DescribeServer",
      "transfer:ListUsers",
      "transfer:ListServers",
      "transfer:TestIdentityProvider",
      "transfer:ListTagsForResource"
    ],
    "Resource": "*"
  }
]
}

```

AWS Transfer Family updates to AWS managed policies

View details about updates to AWS managed policies for AWS Transfer Family since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the [Document history for AWS Transfer Family](#) page.

Change	Description	Date
Documentation update	Added sections for each of the Transfer Family managed policies.	January 27, 2022
AWSTransferReadOnlyAccess – Update to an existing policy	AWS Transfer Family added new permissions to allow the policy to read AWS Managed Microsoft AD.	September 30, 2021
AWS Transfer Family started tracking changes	AWS Transfer Family started tracking changes for its AWS managed policies.	June 15, 2021

Troubleshooting AWS Transfer Family

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Transfer Family.

For issues with IAM in Transfer Family, see [Troubleshooting AWS Transfer Family identity and access](#).

Topics

- [Troubleshoot service-managed users](#)
- [Troubleshoot Amazon API Gateway issues](#)
- [Troubleshoot policies for encrypted Amazon S3 buckets](#)
- [Troubleshoot authentication issues](#)
- [Troubleshoot managed workflows issues](#)
- [Troubleshoot workflow decryption issues](#)
- [Troubleshoot Amazon EFS issues](#)
- [Troubleshoot testing your identity provider](#)
- [Troubleshoot adding trusted host keys for your SFTP connector](#)
- [Troubleshoot file upload issues](#)
- [Troubleshoot ResourceNotFound exception](#)
- [Troubleshoot SFTP connector issues](#)
- [Troubleshoot AS2 issues](#)

Troubleshoot service-managed users

This section describes possible solutions for the following issues.

Topics

- [Troubleshoot Amazon EFS service-managed users](#)
- [Troubleshoot public key body too long](#)
- [Troubleshoot failed to add SSH public key](#)

Troubleshoot Amazon EFS service-managed users

Description

You run the `sftp` command and the prompt doesn't appear, and instead you see the following message:

```
Couldn't canonicalize: Permission denied
Need cwd
```

Cause

Your AWS Identity and Access Management (IAM) user's role does not have permission to access Amazon Elastic File System (Amazon EFS).

Solution

Increase the policy permissions for your user's role. You can add an AWS managed policy, such as `AmazonElasticFileSystemClientFullAccess`.

Troubleshoot public key body too long

Description

When you try to create a service-managed user, you receive the following error:

```
Failed to create user (1 validation error detected:
'sshPublicKeyBody' failed to satisfy constraint: Member must have length less than or
equal to 2048)
```

Cause

You might be entering a PGP key for the public key body, and AWS Transfer Family does not support PGP keys for service-managed users.

Solution

If the PGP key is RSA-based, you can convert it to PEM format. For example, Ubuntu provides a conversion tool here: <https://manpages.ubuntu.com/manpages/xenial/man1/openpgp2ssh.1.html>

Troubleshoot failed to add SSH public key

Description

When you try to add a public key for a service-managed user, you receive the following error:

```
Failed to add SSH public key (Unsupported or invalid SSH public key format)
```

Cause

You might be attempting to import an SSH2-formatted public key, and AWS Transfer Family does not support SSH2-formatted public keys for service-managed users.

Solution

You need to convert the key into OpenSSH format. This process is described in [Convert an SSH2 public key to PEM format](#).

Troubleshoot Amazon API Gateway issues

This section describes possible solutions for the following API Gateway issues.

Topics

- [Too many authentication failures](#)
- [Connection closed](#)

Too many authentication failures

Description

When you try to connect to your server using Secure Shell (SSH) File Transfer Protocol (SFTP), you get the following error:

```
Received disconnect from 3.15.127.197 port 22:2: Too many authentication failures
Authentication failed.
Couldn't read packet: Connection reset by peer
```

Cause

You might have entered an incorrect password for your user. Try again to enter the correct password.

If the password is correct, the issue might be caused by a role Amazon Resource Name (ARN) that is not valid. To confirm that this is the issue, test the identity provider for your server. If you see

a response similar to the following, the role ARN is a placeholder only, as indicated by the role ID value of all zeros:

```
{
  "Response": "{\"Role\": \"arn:aws:iam::000000000000:role/MyUserS3AccessRole\",
  \"HomeDirectory\": \"/\",
  \"StatusCode\": 200,
  \"Message\": \"\",
  \"Url\": \"https://api-gateway-ID.execute-api.us-east-1.amazonaws.com/prod/
  servers/transfer-server-ID/users/myuser/config\"
}
```

Solution

Replace the placeholder role ARN with an actual role that has permission to access the server.

To update the role

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. In the left navigation pane, choose **Stacks**.
3. In the **Stacks** list, choose your stack, and then choose the **Parameters** tab.
4. Choose **Update**. On the **Update stack** page, choose **Use current template**, and then choose **Next**.
5. Replace **UserRoleArn** with a role ARN that has sufficient permissions for accessing your Transfer Family server.

Note

To grant the necessary permissions, you can add the AmazonAPIGatewayAdministrator and the AmazonS3FullAccess managed policies to your role.

6. Choose **Next**, and then choose **Next** again. On the **Review *stack*** page, select **I acknowledge that AWS CloudFormation might create IAM resources**, and then choose **Update stack**.

Connection closed

Description

When you try to connect to your server using Secure Shell (SSH) File Transfer Protocol (SFTP), you get the following error:

```
Connection closed
```

Cause

One possible cause for this issue is that your Amazon CloudWatch logging role does not have a trust relationship with Transfer Family.

Solution

Make sure that the logging role for the server has a trust relationship with Transfer Family. For more information, see [To establish a trust relationship](#).

Troubleshoot policies for encrypted Amazon S3 buckets

Description

You have an encrypted Amazon S3 bucket that you are using as storage for your Transfer Family server. If you try to upload a file to the server, you receive the error `Couldn't close file: Permission denied`.

And if you view the server logs, you see the following errors:

```
ERROR Message="Access denied" Operation=CLOSE Path=/bucket/user/test.txt BytesIn=13  
ERROR Message="Access denied"
```

Cause

The policy for your IAM user does not have permission to access the encrypted bucket.

Solution

You must specify additional permissions in your policy to grant the required AWS Key Management Service (AWS KMS) permissions. For details, see [Data encryption in Amazon S3](#).

Troubleshoot authentication issues

This section describes possible solutions for the following authentication issues.

Topics

- [Authentication failures—SSH/SFTP](#)
- [Managed AD mismatched realms issue](#)
- [Miscellaneous authentication issues](#)

Authentication failures—SSH/SFTP

Description

When you try to connect to your server using Secure Shell (SSH) File Transfer Protocol (SFTP), you receive a message similar to the following:

```
Received disconnect from 3.130.115.105 port 22:2: Too many authentication failures
Authentication failed.
```

Note

If you are using an API Gateway and receive this error, see [Too many authentication failures](#).

Cause

You have not added an RSA key pair for your user, so you must authenticate using a password instead.

Solution

When you run the `sftp` command, specify the `-o PubkeyAuthentication=no` option. This option forces the system to request your password. For example:

```
sftp -o PubkeyAuthentication=no sftp-user@server-id.server.transfer.region-id.amazonaws.com
```

Managed AD mismatched realms issue

Description

A user's realm and their group realm must match. They must both be in the default realm, or they must both be in the trusted realm.

Cause

If a user and their group do not match, the user cannot be authenticated by Transfer Family. If you test the identity provider for the user, you receive the error No associated access found for user's groups.

Solution

Reference a group in the user's realm that matches the group realm (either default or trusted).

Miscellaneous authentication issues

Description

You receive an authentication error and none of the other troubleshooting works

Cause

You might have specified a target for a logical directory that contains a leading or trailing slash (/).

Solution

Update your logical directory target, to make sure it begins with a slash, and does not contain a trailing slash. For example, /DOC-EXAMPLE-BUCKET/images is acceptable, but DOC-EXAMPLE-BUCKET/images and /DOC-EXAMPLE-BUCKET/images/ are not.

Troubleshoot managed workflows issues

This section describes possible solutions for the following workflow issues.

Topics

- [Troubleshoot workflow-related errors using Amazon CloudWatch](#)
- [Troubleshoot workflow copy errors](#)

Troubleshoot workflow-related errors using Amazon CloudWatch

Description

If you are having issues with your workflows, you can use Amazon CloudWatch to investigate the cause.

Cause

There can be several causes. Use Amazon CloudWatch Logs to investigate.

Solution

Transfer Family emits workflow execution status into CloudWatch Logs. The following types of workflow errors can appear in CloudWatch Logs:

- "type": "StepErrored"
- "type": "ExecutionErrored"
- "type": "ExecutionThrottled"
- "Service failure on starting workflow"

You can filter your workflow's execution logs using different filter and pattern syntax. For example, you can create a log filter in your CloudWatch logs to capture workflow execution logs that contain the **ExecutionErrored** message. For details, see [Real-time processing of log data with subscriptions](#) and [Filter and pattern syntax](#) in the *Amazon CloudWatch Logs User Guide*.

StepErrored

```
2021-10-29T12:57:26.272-05:00
    {"type":"StepErrored","details":
{"errorType":"BAD_REQUEST","errorMessage":"Cannot
tag Efs file","stepType":"TAG","stepName":"successful_tag_step"},
"workflowId":"w-
abcdef01234567890","executionId":"1234abcd-56ef-78gh-90ij-1234klmno567",
"transferDetails":
{"serverId":"s-1234567890abcdef0","username":"lhr","sessionId":"1234567890abcdef0"}}
```

Here, **StepErrored** indicates that a step within the workflow has generated an error. In a single workflow, you can have multiple steps configured. This error tells you in which step the error occurred and provides an error message. In this particular example, the step was configured to tag a file; however, tagging a file in an Amazon EFS file system is not supported, so the step generated an error.

ExecutionErrored

```
2021-10-29T12:57:26.618-05:00
```

```

{"type": "ExecutionErrored", "details": {}, "workflowId": "w-w-
abcdef01234567890",
  "executionId": "1234abcd-56ef-78gh-90ij-1234klmno567", "transferDetails":
{"serverId": "s-1234567890abcdef0",
  "username": "lhr", "sessionId": "1234567890abcdef0"}}

```

When a workflow is not able to execute any step, it generates an `ExecutionErrored` message. For example, if you have configured a single step in a given workflow, and if the step is not able to execute, the overall workflow fails.

Executionthrottled

Execution is throttled if a workflow is getting triggered at a rate that is faster than the system can support. This log message indicates that you must slow down your execution rate for workflows. If you are not able to scale down your workflow-execution rate, contact AWS Support at [Contact AWS](#).

Service failure on starting workflow

Anytime you remove a workflow from a server and replace it with a new one, or update server configuration (which impacts a workflow's execution role), you must wait approximately 10 minutes before executing the new workflow. The Transfer Family server caches the workflow details, and it takes 10 minutes for the server to refresh its cache.

Additionally, you must log out of any active SFTP sessions, and then log back in after the 10-minute waiting period to see the changes.

Troubleshoot workflow copy errors

Description

If you're executing a workflow that contains a step to copy the uploaded file, you could encounter the following error:

```

{
  "type": "StepErrored", "details": {
    "errorType": "BAD_REQUEST", "errorMessage": "Bad Request (Service: Amazon S3;
    Status Code: 400; Error Code: 400 Bad Request;
    Request ID: request-ID; S3 Extended Request ID: request-ID Proxy: null)",
    "stepType": "COPY", "stepName": "copy-step-name" },

```



```
"workflowId": "workflow-ID",
"executionId": "execution-ID",
"transferDetails": {
  "serverId": "server-ID",
  "username": "user-name",
  "sessionId": "session-ID"
}
}
```

Cause

The source file is in an Amazon S3 bucket that is in a different AWS Region than the destination bucket.

Solution

If you're executing a workflow that includes a copy step, make sure that the source and destination buckets are in the same AWS Region.

Troubleshoot workflow decryption issues

This section describes possible solutions for the following issues with encrypted workflows.

Topics

- [Troubleshoot error for signed encryption file](#)
- [Troubleshoot error for a FIPS algorithm](#)

Troubleshoot error for signed encryption file

Description

Your decrypt workflow fails and you receive the following error:

```
"Encrypted file with signed message unsupported"
```

Cause

Transfer Family does not currently support signing for encrypted files.

Solution

In your PGP client, if there is an option to sign the encrypted file, make sure to clear the selection, as Transfer Family does not currently support signing for encrypted files.

Troubleshoot error for a FIPS algorithm

Description

Your decrypt workflow fails, and the log message resembles the following:

```
{
  "type": "StepErrored",
  "details": {
    "errorType": "BAD_REQUEST",
    "errorMessage": "File encryption algorithm not supported with FIPS mode
enabled.",
    "stepType": "DECRYPT",
    "stepName": "step-name"
  },
  "workflowId": "workflow-ID",
  "executionId": "execution-ID",
  "transferDetails": {
    "serverId": "server-ID",
    "username": "user-name",
    "sessionId": "session-ID"
  }
}
```

Cause

Your Transfer Family server has FIPS mode enabled and an associated Decrypt workflow step. When encrypting the files before uploading to your Transfer Family server, the encryption client might generate encrypted files that use non-FIPS approved symmetric encryption algorithms. In such a scenario, the workflow is unable to decrypt files. In the following example, **GnuPG** version 2.4.0 is using OCB (a non-FIPS block cipher mode) to encrypt files: this causes the workflow to fail.

Solution

You must edit the GPG key that you used to encrypt your files, and then re-encrypt them. The following procedure describes the steps you must take.

To edit your PGP keys

1. Identify the key that you must edit by running `gpg --list-keys`

This returns a list of keys. Each key has details similar to the following:

```
pub  ed25519 2022-07-07 [SC]
     wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
uid  [ultimate] Mary Major <marymajor@example.com>
sub  cv25519 2022-07-07 [E]
```

2. Identify the key that you want to edit. In the example shown in the previous step, the ID is `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`.
3. Run `gpg --edit-key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`.

The system responds with details about the **GnuPG** program and the specified key.

4. At the `gpg>` prompt, enter `showpref`. The following details are returned:

```
[ultimate] (1). Mary Major <marymajor@example.com>
  Cipher: AES256, AES192, AES, 3DES
  AEAD: OCB
  Digest: SHA512, SHA384, SHA256, SHA224, SHA1
  Compression: ZLIB, BZIP2, ZIP, Uncompressed
  Features: MDC, AEAD, Keyserver no-modify
```

Note that the preferred algorithms that are stored on the key are listed.

5. We want to edit the key to retain all algorithms except for **OCB**. Run the `setpref` command, specifying all the algorithms to retain:

```
gpg> setpref AES256, AES192, AES, 3DES, SHA512, SHA384, SHA256, SHA224, SHA1, ZLIB,
  BZIP2, ZIP, Uncompressed
```

This returns the following details:

```
Set preference list to:
  Cipher: AES256, AES192, AES, 3DES
  AEAD:
  Digest: SHA512, SHA384, SHA256, SHA224, SHA1
  Compression: ZLIB, BZIP2, ZIP, Uncompressed
  Features: MDC, Keyserver no-modify
```

```
Really update the preferences? (y/N)
```

6. Enter `y` to update, then enter your password when prompted to confirm the change.
7. Save the changes.

```
gpg> save
```

Before re-running your decrypt workflow, you must re-encrypt your files, using the edited key.

Troubleshoot Amazon EFS issues

This section describes possible solutions for the following Amazon EFS issues.

Topics

- [Troubleshoot missing POSIX profile](#)
- [Troubleshoot logical directories with Amazon EFS](#)

Troubleshoot missing POSIX profile

Description

If you're using Amazon EFS storage for your server and you're using a custom identity provider, you must provide your AWS Lambda function with a POSIX profile.

Cause

One possible cause is that the templates that we provide for creating an AWS Lambda-backed Amazon API Gateway method do not currently contain POSIX information.

If you did provide POSIX information, the format that you used for providing the POSIX information might not be getting parsed correctly by Transfer Family.

Solution

Make sure that you are providing a JSON element to Transfer Family for the `PosixProfile` parameter.

For example, if you're using Python, you could add the following line where you parse the `PosixProfile` parameter:

```
if PosixProfile:
    response_data["PosixProfile"] = json.loads(PosixProfile)
```

Or, in JavaScript, you could add the following line, where the *uid-value* and *gid-value* are integers, 0 or greater, that represent the User ID (UID) and Group ID (GID) respectively:

```
PosixProfile: {"Uid": uid-value, "Gid": gid-value},
```

These code examples send the PosixProfile parameter to Transfer Family as a JSON object, rather than as a string.

Also, within AWS Secrets Manager, you must store the PosixProfile parameter as follows. Replace *your-uid* and *your-gid* with your actual values for the GID and UID.

```
{"Uid": your-uid, "Gid": your-gid, "SecondaryGids": []}
```

Troubleshoot logical directories with Amazon EFS

Description

If the user's home directory does not exist, and they run an `ls` command, the system responds as follows:

```
sftp> ls
remote readdir ("/"): No such file or directory
```

Cause

If your Transfer Family server uses Amazon EFS, the home directory for the user must be created with read and write access before the user can work in their logical home directory. The user cannot create this directory themselves, as they would lack permissions for `mkdir` on their logical home directory.

Solution

A user with administrative access to the parent directory needs to create the user's logical home directory.

Troubleshoot testing your identity provider

Description

If you test your identity provider using the console or the `TestIdentityProvider` API call, the `Response` field is empty. For example:

```
{
  "Response": "{}",
  "StatusCode": 200,
  "Message": ""
}
```

Cause

The most likely cause is that the authentication failed because of an incorrect user name or password.

Solution

Make sure that you are using the correct credentials for your user, and make updates to the username or password, if necessary.

Troubleshoot adding trusted host keys for your SFTP connector

Description

When you are creating or editing an SFTP connector, and you are adding a trusted host key, you receive the following error: `Failed to edit connector details (Invalid host key format.)`

Cause

If you paste in a correct public key, the issue might be that you included the comment portion of the key. AWS Transfer Family does not currently accept the comment portion of the key.

Solution

Delete the comment portion of the key, when you paste it into the text field. For example, assume your key looks similar to the following:

```
ssh-rsa AAAA...== marymajor@dev-dsk-marymajor-1d-c1234567.us-east-1.amazon.com
```

Remove the text that follows the == characters and only paste in the portion of the key up to and including the ==.

```
ssh-rsa AAAA...==
```

Troubleshoot file upload issues

This section describes possible solutions for the following file upload issues.

Topics

- [Troubleshoot Amazon S3 file upload errors](#)
- [Troubleshoot unreadable file names](#)

Troubleshoot Amazon S3 file upload errors

Description

When you are attempting to upload a file to Amazon S3 storage using Transfer Family, you receive the following error message: AWS Transfer does not support random access writes to S3 objects.

Cause

When you're using Amazon S3 for your server's storage, Transfer Family does not support multiple connections for a single transfer.

Solution

If your Transfer Family server is using Amazon S3 for its storage, disable any options for your client software that mention using multiple connections for a single transfer.

Troubleshoot unreadable file names

Description

You see corrupted file names in some of your uploaded files. Users sometimes encounter problems with FTP and SFTP transfers that garble certain characters in file names, such as umlauts, accented letters, or certain scripts, such as Chinese or Arabic.

Cause

Although the FTP and SFTP protocols can allow for character encoding of file names to be negotiated by clients, Amazon S3 and Amazon EFS do not. Instead, they require UTF-8 character encoding. As a result, certain characters are not rendered correctly.

Solution

To solve this problem, review your client application for file name character encoding and make sure it is set to UTF-8.

Troubleshoot ResourceNotFoundException

Description

You receive an error where the resource cannot be found. For example, if you run `UpdateServer`, you might get the following error:

```
An error occurred (ResourceNotFoundException) when calling the UpdateServer operation:  
Unknown server
```

Cause

There are several reasons for receiving a `ResourceNotFoundException` message. In most cases, the resource that you specified in your API command does not exist. If you did specify an existing resource, then the most probable cause is that your default region is different than the region for your resource. For example, if your default region is **us-east-1**, and your Transfer Family server is in **us-east-2**, you will receive an Unknown resource exception.

For details about setting a default region, see [Quick configuration with `aws configure`](#).

Solution

Add a region parameter to your API command to explicitly specify where to find a particular resource.

```
aws transfer -describe-server --server-id server-id --region us-east-2
```

Troubleshoot SFTP connector issues

This section describes possible solutions for the following SFTP connector issues.

Topics

- [Key negotiation fails](#)
- [Miscellaneous SFTP connector issues](#)

Key negotiation fails

Description

You receive an error where the key exchange negotiation fails. For example:

```
Key exchange negotiation failed due to incompatible host key algorithms.  
Client offered: [ecdsa-sha2-nistp256, ecdsa-sha2-nistp384,  
ecdsa-sha2-nistp521, rsa-sha2-512, rsa-sha2-256] Server offered: [ssh-rsa]
```

Cause

This error is because there's no overlap between the host key algorithms supported by the server and those supported by the connector.

Solution

Ensure that the remote server supports at least one of the Client host key algorithms listed in the error message. For the list of supported algorithms, see [Security policies for AWS Transfer Family SFTP connectors](#).

Miscellaneous SFTP connector issues

Description

You receive an error after you run `StartFileTransfer`, but do not know the cause of the issue, and only the connector ID is returned after the API call.

Cause

This error can have several causes. To troubleshoot, we recommend that you test your connector and search your CloudWatch logs.

Solution

- **Test your connector:** See [Test an SFTP connector](#). If the test fails, the system provides an error message based on the reason the test failed. That section describes how to test your connector from either the console or by using the [TestConnection](#) API command.
- **View CloudWatch logs for your connector:** See [Example log entries for SFTP connectors](#). This topic provides examples for SFTP connector log entries, and the naming convention to help you find the appropriate logs.

Troubleshoot AS2 issues

Error messages and troubleshooting tips for Applicability Statement 2 (AS2)-enabled servers are described here: [AS2 error codes](#).

API reference

The following sections document the AWS Transfer Family API service calls, data types, parameters, and errors.

Topics

- [Welcome to the AWS Transfer Family API](#)
- [Actions](#)
- [Data Types](#)
- [Making API requests](#)
- [Common Parameters](#)
- [Common Errors](#)

Welcome to the AWS Transfer Family API

AWS Transfer Family is a secure transfer service that you can use to transfer files into and out of Amazon Simple Storage Service (Amazon S3) storage over the following protocols:

- Secure Shell (SSH) File Transfer Protocol (SFTP)
- File Transfer Protocol Secure (FTPS)
- File Transfer Protocol (FTP)
- Applicability Statement 2 (AS2)

File transfer protocols are used in data exchange workflows across different industries such as financial services, healthcare, advertising, and retail, among others. AWS Transfer Family simplifies the migration of file transfer workflows to AWS.

To use the AWS Transfer Family service, you instantiate a server in the AWS Region of your choice. You can create the server, list available servers, and update and delete servers. The server is the entity that requests file operations from AWS Transfer Family. Servers have a number of important properties. The server is a named instance as identified by a system assigned `ServerId` identifier. You can optionally assign a hostname, or even a custom hostname to a server. The service bills for any instantiated servers (even ones `OFFLINE`), and for the amount of data transferred.

Users must be known to the server that requests file operations. A user as identified by their username is assigned to a server. Usernames are used to authenticate requests. A server can have only one authentication method: `AWS_DIRECTORY_SERVICE`, `SERVICE_MANAGED`, `AWS_LAMBDA`, or `API_GATEWAY`.

You can use any of the following identity provider types to authenticate users:

- For `SERVICE_MANAGED`, an SSH public key is stored with the user's properties on a server. A user can have one or more SSH public keys on file for the `SERVICE_MANAGED` authentication method. When a client requests a file operation for `SERVICE_MANAGED` method, the client provides the username and SSH private key, which is authenticated, and access is provided.
- You can manage user authentication and access with your Microsoft Active Directory groups by selecting the `AWS_DIRECTORY_SERVICE` authentication method.
- You can connect to a custom identity provider by using AWS Lambda. Choose the `AWS_LAMBDA` authentication method.
- You can also authenticate user requests using a custom authentication method that provides both user authentication and access. This method relies on the Amazon API Gateway to use your API call from your identity provider to validate user requests. This method is referred to as `API_GATEWAY` in API calls, and as **Custom** in the console. You might use this custom method to authenticate users against a directory service, a database name/password pair, or some other mechanism.

Users are assigned a policy with a trust relationship between themselves and an Amazon S3 bucket. They might be able to access all or part of a bucket. For a server to act on a user's behalf, the server must inherit the trust relationship from the user. An AWS Identity and Access Management (IAM) role is created that contains the trust relationship, and that role is assigned an `AssumeRole` action. The server then can perform file operations as if it were the user.

Users who have a home directory property set will have that directory (or folder) act as the target and source of file operations. When no home directory is set, the bucket's `root` directory becomes the landing directory.

Servers, users, and roles are all identified by their Amazon Resource Name (ARN). You can assign tags, which are key-value pairs, to entities with an ARN. Tags are metadata that can be used to group or search for these entities. One example where tags are useful is for accounting purposes.

The following conventions are observed in AWS Transfer Family ID formats:

- `ServerId` values take the form `s-01234567890abcdef`.
- `SshPublicKeyId` values take the form `key-01234567890abcdef`.

Amazon Resource Name (ARN) formats take the following form:

- For servers, ARNs take the form `arn:aws:transfer:region:account-id:server/server-id`.

An example of a server ARN is: `arn:aws:transfer:us-east-1:123456789012:server/s-01234567890abcdef`.

- For users, ARNs take the form `arn:aws:transfer:region:account-id:user/server-id/username`.

An example is `arn:aws:transfer:us-east-1:123456789012:user/s-01234567890abcdef/user1`.

DNS entries (endpoints) in use are as follows:

- API endpoints take the form `transfer.region.amazonaws.com`.
- Server endpoints take the form `server.transfer.region.amazonaws.com`.

For a list of Transfer Family endpoints by AWS Region, see the [AWS Transfer Family endpoints and quotas](#) in the *AWS General Reference*.

This API interface reference for AWS Transfer Family contains documentation for a programming interface that you can use to manage AWS Transfer Family. The reference structure is as follows:

- For the alphabetical list of API actions, see [Actions](#).
- For the alphabetical list of data types, see [Data Types](#).
- For a list of common query parameters, see [Common Parameters](#).
- For descriptions of the error codes, see [Common Errors](#).

Tip

Rather than actually running a command, you can use the `--generate-cli-skeleton` parameter with any API call to generate and display a parameter template. You can then

use the generated template to customize and use as input on a later command. For details, see [Generate and use a parameter skeleton file](#).

Actions

The following actions are supported:

- [CreateAccess](#)
- [CreateAgreement](#)
- [CreateConnector](#)
- [CreateProfile](#)
- [CreateServer](#)
- [CreateUser](#)
- [CreateWorkflow](#)
- [DeleteAccess](#)
- [DeleteAgreement](#)
- [DeleteCertificate](#)
- [DeleteConnector](#)
- [DeleteHostKey](#)
- [DeleteProfile](#)
- [DeleteServer](#)
- [DeleteSshPublicKey](#)
- [DeleteUser](#)
- [DeleteWorkflow](#)
- [DescribeAccess](#)
- [DescribeAgreement](#)
- [DescribeCertificate](#)
- [DescribeConnector](#)
- [DescribeExecution](#)
- [DescribeHostKey](#)
- [DescribeProfile](#)

- [DescribeSecurityPolicy](#)
- [DescribeServer](#)
- [DescribeUser](#)
- [DescribeWorkflow](#)
- [ImportCertificate](#)
- [ImportHostKey](#)
- [ImportSshPublicKey](#)
- [ListAccesses](#)
- [ListAgreements](#)
- [ListCertificates](#)
- [ListConnectors](#)
- [ListExecutions](#)
- [ListHostKeys](#)
- [ListProfiles](#)
- [ListSecurityPolicies](#)
- [ListServers](#)
- [ListTagsForResource](#)
- [ListUsers](#)
- [ListWorkflows](#)
- [SendWorkflowStepState](#)
- [StartDirectoryListing](#)
- [StartFileTransfer](#)
- [StartServer](#)
- [StopServer](#)
- [TagResource](#)
- [TestConnection](#)
- [TestIdentityProvider](#)
- [UntagResource](#)
- [UpdateAccess](#)
- [UpdateAgreement](#)

- [UpdateCertificate](#)
- [UpdateConnector](#)
- [UpdateHostKey](#)
- [UpdateProfile](#)
- [UpdateServer](#)
- [UpdateUser](#)

CreateAccess

Used by administrators to choose which groups in the directory should have access to upload and download files over the enabled protocols using AWS Transfer Family. For example, a Microsoft Active Directory might contain 50,000 users, but only a small fraction might need the ability to transfer files to the server. An administrator can use `CreateAccess` to limit the access to the correct set of users who need this ability.

Request Syntax

```
{
  "ExternalId": "string",
  "HomeDirectory": "string",
  "HomeDirectoryMappings": [
    {
      "Entry": "string",
      "Target": "string",
      "Type": "string"
    }
  ],
  "HomeDirectoryType": "string",
  "Policy": "string",
  "PosixProfile": {
    "Gid": number,
    "SecondaryGids": [ number ],
    "Uid": number
  },
  "Role": "string",
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ExternalId

A unique identifier that is required to identify specific groups within your directory. The users of the group that you associate have access to your Amazon S3 or Amazon EFS resources over the

enabled protocols using AWS Transfer Family. If you know the group name, you can view the SID values by running the following command using Windows PowerShell.

```
Get-ADGroup -Filter {samAccountName -like "YourGroupName*"} -Properties * | Select SamAccountName, ObjectSid
```

In that command, replace *YourGroupName* with the name of your Active Directory group.

The regular expression used to validate this parameter is a string of characters consisting of uppercase and lowercase alphanumeric characters with no spaces. You can also include underscores or any of the following characters: =,.,@:/-

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: S-1-[\d-]+

Required: Yes

HomeDirectory

The landing directory (folder) for a user when they log in to the server using the client.

A HomeDirectory example is `/bucket_name/home/mydirectory`.

Note

The HomeDirectory parameter is only used if HomeDirectoryType is set to PATH.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: (|/.*)

Required: No

HomeDirectoryMappings

Logical directory mappings that specify what Amazon S3 or Amazon EFS paths and keys should be visible to your user and how you want to make them visible. You must specify the Entry and Target pair, where Entry shows how the path is made visible and Target is the actual

Amazon S3 or Amazon EFS path. If you only specify a target, it is displayed as is. You also must ensure that your AWS Identity and Access Management (IAM) role provides access to paths in Target. This value can be set only when `HomeDirectoryType` is set to `LOGICAL`.

The following is an Entry and Target pair example.

```
[ { "Entry": "/directory1", "Target": "/bucket_name/home/mydirectory" } ]
```

In most cases, you can use this value instead of the session policy to lock down your user to the designated home directory ("chroot"). To do this, you can set Entry to / and set Target to the `HomeDirectory` parameter value.

The following is an Entry and Target pair example for chroot.

```
[ { "Entry": "/", "Target": "/bucket_name/home/mydirectory" } ]
```

Type: Array of [HomeDirectoryMapEntry](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50000 items.

Required: No

[HomeDirectoryType](#)

The type of landing directory (folder) that you want your users' home directory to be when they log in to the server. If you set it to `PATH`, the user will see the absolute Amazon S3 bucket or Amazon EFS path as is in their file transfer protocol clients. If you set it to `LOGICAL`, you need to provide mappings in the `HomeDirectoryMappings` for how you want to make Amazon S3 or Amazon EFS paths visible to your users.

Note

If `HomeDirectoryType` is `LOGICAL`, you must provide mappings, using the `HomeDirectoryMappings` parameter. If, on the other hand, `HomeDirectoryType` is `PATH`, you provide an absolute path using the `HomeDirectory` parameter. You cannot have both `HomeDirectory` and `HomeDirectoryMappings` in your template.

Type: String

Valid Values: `PATH` | `LOGICAL`

Required: No

Policy

A session policy for your user so that you can use the same AWS Identity and Access Management (IAM) role across multiple users. This policy scopes down a user's access to portions of their Amazon S3 bucket. Variables that you can use inside this policy include `${Transfer:UserName}`, `${Transfer:HomeDirectory}`, and `${Transfer:HomeBucket}`.

Note

This policy applies only when the domain of `ServerId` is Amazon S3. Amazon EFS does not use session policies.

For session policies, AWS Transfer Family stores the policy as a JSON blob, instead of the Amazon Resource Name (ARN) of the policy. You save the policy as a JSON blob and pass it in the `Policy` argument.

For an example of a session policy, see [Example session policy](#).

For more information, see [AssumeRole](#) in the *AWS Security Token Service API Reference*.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Required: No

PosixProfile

The full POSIX identity, including user ID (`Uid`), group ID (`Gid`), and any secondary groups IDs (`SecondaryGids`), that controls your users' access to your Amazon EFS file systems. The POSIX permissions that are set on files and directories in your file system determine the level of access your users get when transferring files into and out of your Amazon EFS file systems.

Type: [PosixProfile](#) object

Required: No

Role

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that controls your users' access to your Amazon S3 bucket or Amazon EFS file system. The policies attached to this role determine the level of access that you want to provide your users when

transferring files into and out of your Amazon S3 bucket or Amazon EFS file system. The IAM role should also contain a trust relationship that allows the server to access your resources when servicing your users' transfer requests.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: Yes

ServerId

A system-assigned unique identifier for a server instance. This is the specific server that you added your user to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Required: Yes

Response Syntax

```
{
  "ExternalId": "string",
  "ServerId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ExternalId

The external identifier of the group whose users have access to your Amazon S3 or Amazon EFS resources over the enabled protocols using AWS Transfer Family.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: S-1-[\d-]+

ServerId

The identifier of the server that the user is attached to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateAgreement

Creates an agreement. An agreement is a bilateral trading partner agreement, or partnership, between an AWS Transfer Family server and an AS2 process. The agreement defines the file and message transfer relationship between the server and the AS2 process. To define an agreement, Transfer Family combines a server, local profile, partner profile, certificate, and other attributes.

The partner is identified with the `PartnerProfileId`, and the AS2 process is identified with the `LocalProfileId`.

Request Syntax

```
{
  "AccessRole": "string",
  "BaseDirectory": "string",
  "Description": "string",
  "LocalProfileId": "string",
  "PartnerProfileId": "string",
  "ServerId": "string",
  "Status": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

AccessRole

Connectors are used to send files using either the AS2 or SFTP protocol. For the access role, provide the Amazon Resource Name (ARN) of the AWS Identity and Access Management role to use.

For AS2 connectors

With AS2, you can send files by calling `StartFileTransfer` and specifying the file paths in the request parameter, `SendFilePaths`. We use the file's parent directory (for example, for `--send-file-paths /bucket/dir/file.txt`, parent directory is `/bucket/dir/`) to temporarily store a processed AS2 message file, store the MDN when we receive them from the partner, and write a final JSON file containing relevant metadata of the transmission. So, the `AccessRole` needs to provide read and write access to the parent directory of the file location used in the `StartFileTransfer` request. Additionally, you need to provide read and write access to the parent directory of the files that you intend to send with `StartFileTransfer`.

If you are using Basic authentication for your AS2 connector, the access role requires the `secretsmanager:GetSecretValue` permission for the secret. If the secret is encrypted using a customer-managed key instead of the AWS managed key in Secrets Manager, then the role also needs the `kms:Decrypt` permission for that key.

For SFTP connectors

Make sure that the access role provides read and write access to the parent directory of the file location that's used in the `StartFileTransfer` request. Additionally, make sure that the role provides `secretsmanager:GetSecretValue` permission to AWS Secrets Manager.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: Yes

BaseDirectory

The landing directory (folder) for files transferred by using the AS2 protocol.

A `BaseDirectory` example is `/DOC-EXAMPLE-BUCKET/home/mydirectory`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `(|/.*)`

Required: Yes

Description

A name or short description to identify the agreement.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[\p{Graph}]+`

Required: No

LocalProfileId

A unique identifier for the AS2 local profile.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `p-([0-9a-f]{17})`

Required: Yes

PartnerProfileId

A unique identifier for the partner profile used in the agreement.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `p-([0-9a-f]{17})`

Required: Yes

ServerId

A system-assigned unique identifier for a server instance. This is the specific server that the agreement uses.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Required: Yes

Status

The status of the agreement. The agreement can be either ACTIVE or INACTIVE.

Type: String

Valid Values: ACTIVE | INACTIVE

Required: No

Tags

Key-value pairs that can be used to group and search for agreements.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

Response Syntax

```
{  
  "AgreementId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

AgreementId

The unique identifier for the agreement. Use this ID for deleting, or updating an agreement, as well as in any other API calls that require that you specify the agreement ID.

Type: String

Length Constraints: Fixed length of 19.

Pattern: a-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example creates an agreement, and returns the agreement ID.

```
aws transfer create-agreement --server-id s-021345abcdef6789 --local-profile-id p-1234567890abcdef0 --partner-profile-id p-abcdef01234567890 --base-folder /DOC-EXAMPLE-BUCKET/AS2-files --access-role arn:aws:iam::111122223333:role/AS2-role
```

Sample Response

The API call returns the agreement ID for the new agreement.

```
{
  "AgreementId": "a-11112222333344444"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateConnector

Creates the connector, which captures the parameters for a connection for the AS2 or SFTP protocol. For AS2, the connector is required for sending files to an externally hosted AS2 server. For SFTP, the connector is required when sending files to an SFTP server or receiving files from an SFTP server. For more details about connectors, see [Configure AS2 connectors](#) and [Create SFTP connectors](#).

Note

You must specify exactly one configuration object: either for AS2 (As2Config) or SFTP (SftpConfig).

Request Syntax

```
{
  "AccessRole": "string",
  "As2Config": {
    "BasicAuthSecretId": "string",
    "Compression": "string",
    "EncryptionAlgorithm": "string",
    "LocalProfileId": "string",
    "MdnResponse": "string",
    "MdnSigningAlgorithm": "string",
    "MessageSubject": "string",
    "PartnerProfileId": "string",
    "SigningAlgorithm": "string"
  },
  "LoggingRole": "string",
  "SecurityPolicyName": "string",
  "SftpConfig": {
    "TrustedHostKeys": [ "string" ],
    "UserSecretId": "string"
  },
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
}
```

```
"Url": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

[AccessRole](#)

Connectors are used to send files using either the AS2 or SFTP protocol. For the access role, provide the Amazon Resource Name (ARN) of the AWS Identity and Access Management role to use.

For AS2 connectors

With AS2, you can send files by calling `StartFileTransfer` and specifying the file paths in the request parameter, `SendFilePaths`. We use the file's parent directory (for example, for `--send-file-paths /bucket/dir/file.txt`, parent directory is `/bucket/dir/`) to temporarily store a processed AS2 message file, store the MDN when we receive them from the partner, and write a final JSON file containing relevant metadata of the transmission. So, the `AccessRole` needs to provide read and write access to the parent directory of the file location used in the `StartFileTransfer` request. Additionally, you need to provide read and write access to the parent directory of the files that you intend to send with `StartFileTransfer`.

If you are using Basic authentication for your AS2 connector, the access role requires the `secretsmanager:GetSecretValue` permission for the secret. If the secret is encrypted using a customer-managed key instead of the AWS managed key in Secrets Manager, then the role also needs the `kms:Decrypt` permission for that key.

For SFTP connectors

Make sure that the access role provides read and write access to the parent directory of the file location that's used in the `StartFileTransfer` request. Additionally, make sure that the role provides `secretsmanager:GetSecretValue` permission to AWS Secrets Manager.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: Yes

As2Config

A structure that contains the parameters for an AS2 connector object.

Type: [As2ConnectorConfig](#) object

Required: No

LoggingRole

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that allows a connector to turn on CloudWatch logging for Amazon S3 events. When set, you can view connector activity in your CloudWatch logs.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

SecurityPolicyName

Specifies the name of the security policy for the connector.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 100.

Pattern: `TransferSFTPConnectorSecurityPolicy-[A-Za-z0-9-]+`

Required: No

SftpConfig

A structure that contains the parameters for an SFTP connector object.

Type: [SftpConnectorConfig](#) object

Required: No

Tags

Key-value pairs that can be used to group and search for connectors. Tags are metadata attached to connectors for any purpose.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

Url

The URL of the partner's AS2 or SFTP endpoint.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Required: Yes

Response Syntax

```
{
  "ConnectorId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ConnectorId

The unique identifier for the connector, returned after the API call succeeds.

Type: String

Length Constraints: Fixed length of 19.

Pattern: c-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example creates an AS2 connector. In the command, replace items as follows:

- `url`: provide the URL for the trading partner's AS2 server.
- `your-IAM-role-for-bucket-access`: an IAM role that has access to the Amazon S3 bucket you are using to store your files.
- Use the ARN for your logging role, which includes your AWS account ID.
- Provide a path to a file that contains the AS2 connector configuration parameters. The AS2 connector configuration object is described in [As2ConnectorConfig](#).

```
// Listing for testAs2Config.json
{
  "LocalProfileId": "your-profile-id",
  "PartnerProfileId": "partner-profile-id",
  "MdnResponse": "SYNC",
  "Compression": "ZLIB",
  "EncryptionAlgorithm": "AES256_CBC",
  "SigningAlgorithm": "SHA256",
  "MdnSigningAlgorithm": "DEFAULT",
  "MessageSubject": "Your Message Subject"
}
```

```
aws transfer create-connector --url "http://partner-as2-server-url" \
  --access-role your-IAM-role-for-bucket-access \
  --logging-role arn:aws:iam:your-account-id:role/service-role/
AWSTransferLoggingAccess \
  --as2-config file://path/to/testAS2Config.json
```

Example

The following example creates an SFTP connector. In the command, replace items as follows:

- `sftp-server-url`: provide the URL for the SFTP server with which you are exchanging files.
- `your-IAM-role-for-bucket-access`: an IAM role that has access to the Amazon S3 bucket you are using to store your files.
- Use the ARN for your logging role, which includes your AWS account ID.
- Provide a path to a file that contains the SFTP connector configuration parameters. The SFTP connector configuration object is described in [SftpConnectorConfig](#).

```
// Listing for testSFTPConfig.json
{
  "UserSecretId": "arn:aws:secretsmanager:us-east-2:123456789012:secret:aws/transfer/
example-username-key",
  "TrustedHostKeys": [
    "sftp.example.com ssh-rsa AAAAbbbb...EEEE="
  ]
}
```

```
aws transfer create-connector --url "sftp://sftp-server-url" \
--access-role your-IAM-role-for-bucket-access \
--logging-role arn:aws:iam::your-account-id:role/service-role/AWSTransferLoggingAccess
\
--sftp-config file:///path/to/testSFTPConfig.json
```

Example

The API call returns the connector ID for the new connector.

Sample Response

```
{
  "ConnectorId": "a-11112222333344444"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateProfile

Creates the local or partner profile to use for AS2 transfers.

Request Syntax

```
{
  "As2Id": "string",
  "CertificateIds": [ "string" ],
  "ProfileType": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

As2Id

The As2Id is the *AS2-name*, as defined in the [RFC 4130](#). For inbound transfers, this is the AS2-From header for the AS2 messages sent from the partner. For outbound connectors, this is the AS2-To header for the AS2 messages sent to the partner using the StartFileTransfer API operation. This ID cannot include spaces.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[\p{Print}\s]*`

Required: Yes

CertificateIds

An array of identifiers for the imported certificates. You use this identifier for working with profiles and partner profiles.

Type: Array of strings

Length Constraints: Fixed length of 22.

Pattern: `cert-([0-9a-f]{17})`

Required: No

ProfileType

Determines the type of profile to create:

- Specify LOCAL to create a local profile. A local profile represents the AS2-enabled Transfer Family server organization or party.
- Specify PARTNER to create a partner profile. A partner profile represents a remote organization, external to Transfer Family.

Type: String

Valid Values: LOCAL | PARTNER

Required: Yes

Tags

Key-value pairs that can be used to group and search for AS2 profiles.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

Response Syntax

```
{  
  "ProfileId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ProfileId

The unique identifier for the AS2 profile, returned after the API call succeeds.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example creates a profile, and returns the profile ID.

The certificate IDs are created when you run `import-certificate`, one for the signing certificate, and one for the encryption certificate.

```
aws transfer create-profile --as2-id MYCORP --certificate-ids c-abcdefgh123456hijk  
c-987654aaaa321bbbb
```

Sample Response

The API call returns the profile ID for the new profile.

```
{  
  "ProfileId": "p-11112222333344444"  
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateServer

Instantiates an auto-scaling virtual server based on the selected file transfer protocol in AWS. When you make updates to your file transfer protocol-enabled server or when you work with users, use the service-generated `ServerId` property that is assigned to the newly created server.

Request Syntax

```
{
  "Certificate": "string",
  "Domain": "string",
  "EndpointDetails": {
    "AddressAllocationIds": [ "string" ],
    "SecurityGroupIds": [ "string" ],
    "SubnetIds": [ "string" ],
    "VpcEndpointId": "string",
    "VpcId": "string"
  },
  "EndpointType": "string",
  "HostKey": "string",
  "IdentityProviderDetails": {
    "DirectoryId": "string",
    "Function": "string",
    "InvocationRole": "string",
    "SftpAuthenticationMethods": "string",
    "Url": "string"
  },
  "IdentityProviderType": "string",
  "LoggingRole": "string",
  "PostAuthenticationLoginBanner": "string",
  "PreAuthenticationLoginBanner": "string",
  "ProtocolDetails": {
    "As2Transports": [ "string" ],
    "PassiveIp": "string",
    "SetStatOption": "string",
    "TlsSessionResumptionMode": "string"
  },
  "Protocols": [ "string" ],
  "S3StorageOptions": {
    "DirectoryListingOptimization": "string"
  },
  "SecurityPolicyName": "string",
  "StructuredLogDestinations": [ "string" ],
```

```
"Tags": [
  {
    "Key": "string",
    "Value": "string"
  }
],
"WorkflowDetails": {
  "OnPartialUpload": [
    {
      "ExecutionRole": "string",
      "WorkflowId": "string"
    }
  ],
  "OnUpload": [
    {
      "ExecutionRole": "string",
      "WorkflowId": "string"
    }
  ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

Certificate

The Amazon Resource Name (ARN) of the AWS Certificate Manager (ACM) certificate. Required when `Protocols` is set to `FTPS`.

To request a new public certificate, see [Request a public certificate](#) in the *AWS Certificate Manager User Guide*.

To import an existing certificate into ACM, see [Importing certificates into ACM](#) in the *AWS Certificate Manager User Guide*.

To request a private certificate to use FTPS through private IP addresses, see [Request a private certificate](#) in the *AWS Certificate Manager User Guide*.

Certificates with the following cryptographic algorithms and key sizes are supported:

- 2048-bit RSA (RSA_2048)
- 4096-bit RSA (RSA_4096)
- Elliptic Prime Curve 256 bit (EC_prime256v1)
- Elliptic Prime Curve 384 bit (EC_secp384r1)
- Elliptic Prime Curve 521 bit (EC_secp521r1)

Note

The certificate must be a valid SSL/TLS X.509 version 3 certificate with FQDN or IP address specified and information about the issuer.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1600.

Required: No

Domain

The domain of the storage system that is used for file transfers. There are two domains available: Amazon Simple Storage Service (Amazon S3) and Amazon Elastic File System (Amazon EFS). The default value is S3.

Note

After the server is created, the domain cannot be changed.

Type: String

Valid Values: S3 | EFS

Required: No

EndpointDetails

The virtual private cloud (VPC) endpoint settings that are configured for your server. When you host your endpoint within your VPC, you can make your endpoint accessible only to resources within your VPC, or you can attach Elastic IP addresses and make your endpoint accessible to

clients over the internet. Your VPC's default security groups are automatically assigned to your endpoint.

Type: [EndpointDetails](#) object

Required: No

[EndpointType](#)

The type of endpoint that you want your server to use. You can choose to make your server's endpoint publicly accessible (PUBLIC) or host it inside your VPC. With an endpoint that is hosted in a VPC, you can restrict access to your server and resources only within your VPC or choose to make it internet facing by attaching Elastic IP addresses directly to it.

Note

After May 19, 2021, you won't be able to create a server using `EndpointType=VPC_ENDPOINT` in your AWS account if your account hasn't already done so before May 19, 2021. If you have already created servers with `EndpointType=VPC_ENDPOINT` in your AWS account on or before May 19, 2021, you will not be affected. After this date, use `EndpointType=VPC`.

For more information, see [Discontinuing the use of VPC_ENDPOINT](#).

It is recommended that you use VPC as the `EndpointType`. With this endpoint type, you have the option to directly associate up to three Elastic IPv4 addresses (BYO IP included) with your server's endpoint and use VPC security groups to restrict traffic by the client's public IP address. This is not possible with `EndpointType` set to `VPC_ENDPOINT`.

Type: String

Valid Values: PUBLIC | VPC | VPC_ENDPOINT

Required: No

[HostKey](#)

The RSA, ECDSA, or ED25519 private key to use for your SFTP-enabled server. You can add multiple host keys, in case you want to rotate keys, or have a set of active keys that use different algorithms.

Use the following command to generate an RSA 2048 bit key with no passphrase:

```
ssh-keygen -t rsa -b 2048 -N "" -m PEM -f my-new-server-key.
```

Use a minimum value of 2048 for the `-b` option. You can create a stronger key by using 3072 or 4096.

Use the following command to generate an ECDSA 256 bit key with no passphrase:

```
ssh-keygen -t ecdsa -b 256 -N "" -m PEM -f my-new-server-key.
```

Valid values for the `-b` option for ECDSA are 256, 384, and 521.

Use the following command to generate an ED25519 key with no passphrase:

```
ssh-keygen -t ed25519 -N "" -f my-new-server-key.
```

For all of these commands, you can replace `my-new-server-key` with a string of your choice.

Important

If you aren't planning to migrate existing users from an existing SFTP-enabled server to a new server, don't update the host key. Accidentally changing a server's host key can be disruptive.

For more information, see [Update host keys for your SFTP-enabled server](#) in the *AWS Transfer Family User Guide*.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Required: No

IdentityProviderDetails

Required when `IdentityProviderType` is set to `AWS_DIRECTORY_SERVICE`, `AWS_LAMBDA` or `API_GATEWAY`. Accepts an array containing all of the information required to use a directory in `AWS_DIRECTORY_SERVICE` or invoke a customer-supplied authentication API, including the API Gateway URL. Not required when `IdentityProviderType` is set to `SERVICE_MANAGED`.

Type: [IdentityProviderDetails](#) object

Required: No

IdentityProviderType

The mode of authentication for a server. The default value is `SERVICE_MANAGED`, which allows you to store and access user credentials within the AWS Transfer Family service.

Use `AWS_DIRECTORY_SERVICE` to provide access to Active Directory groups in AWS Directory Service for Microsoft Active Directory or Microsoft Active Directory in your on-premises environment or in AWS using AD Connector. This option also requires you to provide a Directory ID by using the `IdentityProviderDetails` parameter.

Use the `API_GATEWAY` value to integrate with an identity provider of your choosing. The `API_GATEWAY` setting requires you to provide an Amazon API Gateway endpoint URL to call for authentication by using the `IdentityProviderDetails` parameter.

Use the `AWS_LAMBDA` value to directly use an AWS Lambda function as your identity provider. If you choose this value, you must specify the ARN for the Lambda function in the `Function` parameter for the `IdentityProviderDetails` data type.

Type: String

Valid Values: `SERVICE_MANAGED` | `API_GATEWAY` | `AWS_DIRECTORY_SERVICE` | `AWS_LAMBDA`

Required: No

LoggingRole

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that allows a server to turn on Amazon CloudWatch logging for Amazon S3 or Amazon EFS events. When set, you can view user activity in your CloudWatch logs.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Pattern: `(|arn:.*role/\S+)`

Required: No

PostAuthenticationLoginBanner

Specifies a string to display when users connect to a server. This string is displayed after the user authenticates.

Note

The SFTP protocol does not support post-authentication display banners.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Pattern: `[\x09-\x0D\x20-\x7E]*`

Required: No

PreAuthenticationLoginBanner

Specifies a string to display when users connect to a server. This string is displayed before the user authenticates. For example, the following banner displays details about using the system:

```
This system is for the use of authorized users only. Individuals using
this computer system without authority, or in excess of their authority,
are subject to having all of their activities on this system monitored
and recorded by system personnel.
```

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Pattern: `[\x09-\x0D\x20-\x7E]*`

Required: No

ProtocolDetails

The protocol settings that are configured for your server.

- To indicate passive mode (for FTP and FTPS protocols), use the `PassiveIp` parameter. Enter a single dotted-quad IPv4 address, such as the external IP address of a firewall, router, or load balancer.
- To ignore the error that is generated when the client attempts to use the `SETSTAT` command on a file that you are uploading to an Amazon S3 bucket, use the `SetStatOption` parameter. To have the AWS Transfer Family server ignore the `SETSTAT` command and upload files without needing to make any changes to your SFTP client, set the value to

ENABLE_NO_OP. If you set the `SetStatOption` parameter to `ENABLE_NO_OP`, Transfer Family generates a log entry to Amazon CloudWatch Logs, so that you can determine when the client is making a SETSTAT call.

- To determine whether your AWS Transfer Family server resumes recent, negotiated sessions through a unique session ID, use the `TlsSessionResumptionMode` parameter.
- `As2Transports` indicates the transport method for the AS2 messages. Currently, only HTTP is supported.

Type: [ProtocolDetails](#) object

Required: No

Protocols

Specifies the file transfer protocol or protocols over which your file transfer protocol client can connect to your server's endpoint. The available protocols are:

- SFTP (Secure Shell (SSH) File Transfer Protocol): File transfer over SSH
- FTPS (File Transfer Protocol Secure): File transfer with TLS encryption
- FTP (File Transfer Protocol): Unencrypted file transfer
- AS2 (Applicability Statement 2): used for transporting structured business-to-business data

Note

- If you select FTPS, you must choose a certificate stored in AWS Certificate Manager (ACM) which is used to identify your server when clients connect to it over FTPS.
- If `Protocol` includes either FTP or FTPS, then the `EndpointType` must be `VPC` and the `IdentityProviderType` must be either `AWS_DIRECTORY_SERVICE`, `AWS_LAMBDA`, or `API_GATEWAY`.
- If `Protocol` includes FTP, then `AddressAllocationIds` cannot be associated.
- If `Protocol` is set only to SFTP, the `EndpointType` can be set to `PUBLIC` and the `IdentityProviderType` can be set any of the supported identity types: `SERVICE_MANAGED`, `AWS_DIRECTORY_SERVICE`, `AWS_LAMBDA`, or `API_GATEWAY`.
- If `Protocol` includes AS2, then the `EndpointType` must be `VPC`, and domain must be Amazon S3.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 4 items.

Valid Values: SFTP | FTP | FTPS | AS2

Required: No

S3StorageOptions

Specifies whether or not performance for your Amazon S3 directories is optimized. This is disabled by default.

By default, home directory mappings have a TYPE of DIRECTORY. If you enable this option, you would then need to explicitly set the HomeDirectoryMapEntry Type to FILE if you want a mapping to have a file target.

Type: [S3StorageOptions](#) object

Required: No

SecurityPolicyName

Specifies the name of the security policy for the server.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 100.

Pattern: Transfer[A-Za-z0-9]*SecurityPolicy-[A-Za-z0-9-]+

Required: No

StructuredLogDestinations

Specifies the log groups to which your server logs are sent.

To specify a log group, you must provide the ARN for an existing log group. In this case, the format of the log group is as follows:

```
arn:aws:logs:region-name:amazon-account-id:log-group:log-group-name:*
```

For example, `arn:aws:logs:us-east-1:111122223333:log-group:mytestgroup:*`

If you have previously specified a log group for a server, you can clear it, and in effect turn off structured logging, by providing an empty value for this parameter in an `update-server` call. For example:

```
update-server --server-id s-1234567890abcdef0 --structured-log-destinations
```

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 1 item.

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: No

Tags

Key-value pairs that can be used to group and search for servers.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

WorkflowDetails

Specifies the workflow ID for the workflow to assign and the execution role that's used for executing the workflow.

In addition to a workflow to execute when a file is uploaded completely, `WorkflowDetails` can also contain a workflow ID (and execution role) for a workflow to execute on partial upload. A partial upload occurs when the server session disconnects while the file is still being uploaded.

Type: [WorkflowDetails](#) object

Required: No

Response Syntax

```
{  
  "ServerId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ServerId

The service-assigned identifier of the server that is created.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example creates a new server using a VPC_ENDPOINT.

Sample Request

```
{
  "EndpointType": "VPC",
  "EndpointDetails": "...",
  "HostKey": "Your RSA private key",
  "IdentityProviderDetails": "IdentityProvider",
  "IdentityProviderType": "SERVICE_MANAGED",
  "LoggingRole": "CloudWatchLoggingRole",
  "Tags": [
    {
      "Key": "Name",
      "Value": "MyServer"
    }
  ]
}
```

Example

This is a sample response for this API call.

Sample Response

```
{  
  "ServerId": "s-01234567890abcdef"  
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateUser

Creates a user and associates them with an existing file transfer protocol-enabled server. You can only create and associate users with servers that have the `IdentityProviderType` set to `SERVICE_MANAGED`. Using parameters for `CreateUser`, you can specify the user name, set the home directory, store the user's public key, and assign the user's AWS Identity and Access Management (IAM) role. You can also optionally add a session policy, and assign metadata with tags that can be used to group and search for users.

Request Syntax

```
{
  "HomeDirectory": "string",
  "HomeDirectoryMappings": [
    {
      "Entry": "string",
      "Target": "string",
      "Type": "string"
    }
  ],
  "HomeDirectoryType": "string",
  "Policy": "string",
  "PosixProfile": {
    "Gid": number,
    "SecondaryGids": [ number ],
    "Uid": number
  },
  "Role": "string",
  "ServerId": "string",
  "SshPublicKeyBody": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
  "UserName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

HomeDirectory

The landing directory (folder) for a user when they log in to the server using the client.

A HomeDirectory example is `/bucket_name/home/mydirectory`.

Note

The HomeDirectory parameter is only used if HomeDirectoryType is set to PATH.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: (|/.*)

Required: No

HomeDirectoryMappings

Logical directory mappings that specify what Amazon S3 or Amazon EFS paths and keys should be visible to your user and how you want to make them visible. You must specify the Entry and Target pair, where Entry shows how the path is made visible and Target is the actual Amazon S3 or Amazon EFS path. If you only specify a target, it is displayed as is. You also must ensure that your AWS Identity and Access Management (IAM) role provides access to paths in Target. This value can be set only when HomeDirectoryType is set to *LOGICAL*.

The following is an Entry and Target pair example.

```
[ { "Entry": "/directory1", "Target": "/bucket_name/home/mydirectory" } ]
```

In most cases, you can use this value instead of the session policy to lock your user down to the designated home directory ("chroot"). To do this, you can set Entry to `/` and set Target to the value the user should see for their home directory when they log in.

The following is an Entry and Target pair example for chroot.

```
[ { "Entry": "/", "Target": "/bucket_name/home/mydirectory" } ]
```


Type: Array of [HomeDirectoryMapEntry](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50000 items.

Required: No

[HomeDirectoryType](#)

The type of landing directory (folder) that you want your users' home directory to be when they log in to the server. If you set it to `PATH`, the user will see the absolute Amazon S3 bucket or Amazon EFS path as is in their file transfer protocol clients. If you set it to `LOGICAL`, you need to provide mappings in the `HomeDirectoryMappings` for how you want to make Amazon S3 or Amazon EFS paths visible to your users.

Note

If `HomeDirectoryType` is `LOGICAL`, you must provide mappings, using the `HomeDirectoryMappings` parameter. If, on the other hand, `HomeDirectoryType` is `PATH`, you provide an absolute path using the `HomeDirectory` parameter. You cannot have both `HomeDirectory` and `HomeDirectoryMappings` in your template.

Type: String

Valid Values: `PATH` | `LOGICAL`

Required: No

[Policy](#)

A session policy for your user so that you can use the same AWS Identity and Access Management (IAM) role across multiple users. This policy scopes down a user's access to portions of their Amazon S3 bucket. Variables that you can use inside this policy include `${Transfer:UserName}`, `${Transfer:HomeDirectory}`, and `${Transfer:HomeBucket}`.

Note

This policy applies only when the domain of `ServerId` is Amazon S3. Amazon EFS does not use session policies.

For session policies, AWS Transfer Family stores the policy as a JSON blob, instead of the Amazon Resource Name (ARN) of the policy. You save the policy as a JSON blob and pass it in the `Policy` argument.

For an example of a session policy, see [Example session policy](#).

For more information, see [AssumeRole](#) in the *AWS Security Token Service API Reference*.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Required: No

[PosixProfile](#)

Specifies the full POSIX identity, including user ID (Uid), group ID (Gid), and any secondary groups IDs (SecondaryGids), that controls your users' access to your Amazon EFS file systems. The POSIX permissions that are set on files and directories in Amazon EFS determine the level of access your users get when transferring files into and out of your Amazon EFS file systems.

Type: [PosixProfile](#) object

Required: No

[Role](#)

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that controls your users' access to your Amazon S3 bucket or Amazon EFS file system. The policies attached to this role determine the level of access that you want to provide your users when transferring files into and out of your Amazon S3 bucket or Amazon EFS file system. The IAM role should also contain a trust relationship that allows the server to access your resources when servicing your users' transfer requests.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: Yes

[ServerId](#)

A system-assigned unique identifier for a server instance. This is the specific server that you added your user to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Required: Yes

SshPublicKeyBody

The public portion of the Secure Shell (SSH) key used to authenticate the user to the server.

The three standard SSH public key format elements are `<key type>`, `<body base64>`, and an optional `<comment>`, with spaces between each element.

AWS Transfer Family accepts RSA, ECDSA, and ED25519 keys.

- For RSA keys, the key type is `ssh-rsa`.
- For ED25519 keys, the key type is `ssh-ed25519`.
- For ECDSA keys, the key type is either `ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp384`, or `ecdsa-sha2-nistp521`, depending on the size of the key you generated.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Required: No

Tags

Key-value pairs that can be used to group and search for users. Tags are metadata attached to users for any purpose.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

UserName

A unique string that identifies a user and is associated with a `ServerId`. This user name must be a minimum of 3 and a maximum of 100 characters long. The following are valid characters: a-z, A-Z, 0-9, underscore '_', hyphen '-', period '.', and at sign '@'. The user name can't start with a hyphen, period, or at sign.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\w][\w@.-]{2,99}`

Required: Yes

Response Syntax

```
{  
  "ServerId": "string",  
  "UserName": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ServerId

The identifier of the server that the user is attached to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

UserName

A unique string that identifies a Transfer Family user.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\w][\w@.-]{2,99}`

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

To create a user, you can first save the parameters into a JSON file, for example `createUserParameters`, then run the `create-user` API command.

```
{
  "HomeDirectory": "/DOC-EXAMPLE-BUCKET",
  "HomeDirectoryType": "PATH",
  "Role": "arn:aws:iam::111122223333:role/bob-role",
  "ServerId": "s-1111aaaa2222bbbb3",
  "SshPublicKeyBody": "ecdsa-sha2-nistp521 AAAAE2VjZHNhLXNoYTItbmlzdHA...
  bobusa@mycomputer.us-east-1.amazon.com",
```

```
"UserName": "bobusa-API"
}
```

Sample Request

```
aws transfer create-user --cli-input-json file://createUserParameters
```

Sample Response

```
{
  "ServerId": "'s-1111aaaa2222bbbb3",
  "UserName": "bobusa-API"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateWorkflow

Allows you to create a workflow with specified steps and step details the workflow invokes after file transfer completes. After creating a workflow, you can associate the workflow created with any transfer servers by specifying the workflow-details field in CreateServer and UpdateServer operations.

Request Syntax

```
{
  "Description": "string",
  "OnExceptionSteps": [
    {
      "CopyStepDetails": {
        "DestinationFileLocation": {
          "EfsFileLocation": {
            "FileSystemId": "string",
            "Path": "string"
          },
          "S3FileLocation": {
            "Bucket": "string",
            "Key": "string"
          }
        },
        "Name": "string",
        "OverwriteExisting": "string",
        "SourceFileLocation": "string"
      },
      "CustomStepDetails": {
        "Name": "string",
        "SourceFileLocation": "string",
        "Target": "string",
        "TimeoutSeconds": number
      },
      "DecryptStepDetails": {
        "DestinationFileLocation": {
          "EfsFileLocation": {
            "FileSystemId": "string",
            "Path": "string"
          },
          "S3FileLocation": {
            "Bucket": "string",
            "Key": "string"
          }
        }
      }
    }
  ]
}
```

```

    }
  },
  "Name": "string",
  "OverwriteExisting": "string",
  "SourceFileLocation": "string",
  "Type": "string"
},
"DeleteStepDetails": {
  "Name": "string",
  "SourceFileLocation": "string"
},
"TagStepDetails": {
  "Name": "string",
  "SourceFileLocation": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ]
},
"Type": "string"
}
],
"Steps": [
  {
    "CopyStepDetails": {
      "DestinationFileLocation": {
        "EfsFileLocation": {
          "FileSystemId": "string",
          "Path": "string"
        },
        "S3FileLocation": {
          "Bucket": "string",
          "Key": "string"
        }
      },
      "Name": "string",
      "OverwriteExisting": "string",
      "SourceFileLocation": "string"
    },
    "CustomStepDetails": {
      "Name": "string",
      "SourceFileLocation": "string",

```



```
    "Target": "string",
    "TimeoutSeconds": number
  },
  "DecryptStepDetails": {
    "DestinationFileLocation": {
      "EfsFileLocation": {
        "FileSystemId": "string",
        "Path": "string"
      },
      "S3FileLocation": {
        "Bucket": "string",
        "Key": "string"
      }
    },
    "Name": "string",
    "OverwriteExisting": "string",
    "SourceFileLocation": "string",
    "Type": "string"
  },
  "DeleteStepDetails": {
    "Name": "string",
    "SourceFileLocation": "string"
  },
  "TagStepDetails": {
    "Name": "string",
    "SourceFileLocation": "string",
    "Tags": [
      {
        "Key": "string",
        "Value": "string"
      }
    ]
  },
  "Type": "string"
}
],
"Tags": [
  {
    "Key": "string",
    "Value": "string"
  }
]
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

Description

A textual description for the workflow.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `[\w-]*`

Required: No

OnExceptionSteps

Specifies the steps (actions) to take if errors are encountered during execution of the workflow.

Note

For custom steps, the Lambda function needs to send FAILURE to the call back API to kick off the exception steps. Additionally, if the Lambda does not send SUCCESS before it times out, the exception steps are executed.

Type: Array of [WorkflowStep](#) objects

Array Members: Minimum number of 0 items. Maximum number of 8 items.

Required: No

Steps

Specifies the details for the steps that are in the specified workflow.

The TYPE specifies which of the following actions is being taken for this step.

- **COPY** - Copy the file to another location.
- **CUSTOM** - Perform a custom step with an AWS Lambda function target.

- **DECRYPT** - Decrypt a file that was encrypted before it was uploaded.
- **DELETE** - Delete the file.
- **TAG** - Add a tag to the file.

Note

Currently, copying and tagging are supported only on S3.

For file location, you specify either the Amazon S3 bucket and key, or the Amazon EFS file system ID and path.

Type: Array of [WorkflowStep](#) objects

Array Members: Minimum number of 0 items. Maximum number of 8 items.

Required: Yes

Tags

Key-value pairs that can be used to group and search for workflows. Tags are metadata attached to workflows for any purpose.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

Response Syntax

```
{  
  "WorkflowId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: w-([a-z0-9]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

You can save workflow step information into a text file, and then use that file to create a workflow, as in the following example. The following example assumes you have saved your workflow steps into *example-file.json* (in the same folder from where you run the command), and that you wish to create the workflow in the N. Virginia (us-east-1) region.

```
aws transfer create-workflow --description "example workflow from a file" --steps
file://example-file.json --region us-east-1
```

```
// Example file containing workflow steps
[
  {
    "Type": "TAG",
    "TagStepDetails": {
      "Name": "TagStep",
      "Tags": [
        {
          "Key": "name",
          "Value": "testTag"
        }
      ]
    }
  },
  {
    "Type": "COPY",
    "CopyStepDetails": {
      "Name": "CopyStep",
      "DestinationFileLocation": {
        "S3FileLocation": {
          "Bucket": "DOC-EXAMPLE-BUCKET",
          "Key": "DOC-EXAMPLE-KEY/"
        }
      }
    }
  }
]
```

```
    },
    "OverwriteExisting": "TRUE",
    "SourceFileLocation": "${original.file}"
  }
},
{
  "Type": "DELETE",
  "DeleteStepDetails":{
    "Name":"DeleteStep",
    "SourceFileLocation": "${original.file}"
  }
}
]
```

Example

The `CreateWorkflow` call returns the workflow ID for the new workflow.

Sample Response

```
{
  "WorkflowId": "w-1234abcd5678efghi"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteAccess

Allows you to delete the access specified in the `ServerID` and `ExternalID` parameters.

Request Syntax

```
{  
  "ExternalId": "string",  
  "ServerId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ExternalId

A unique identifier that is required to identify specific groups within your directory. The users of the group that you associate have access to your Amazon S3 or Amazon EFS resources over the enabled protocols using AWS Transfer Family. If you know the group name, you can view the SID values by running the following command using Windows PowerShell.

```
Get-ADGroup -Filter {samAccountName -like "YourGroupName*"} -Properties  
* | Select SamAccountName, ObjectSid
```

In that command, replace *YourGroupName* with the name of your Active Directory group.

The regular expression used to validate this parameter is a string of characters consisting of uppercase and lowercase alphanumeric characters with no spaces. You can also include underscores or any of the following characters: `=, @: / -`

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: S-1-[\d-]+

Required: Yes

ServerId

A system-assigned unique identifier for a server that has this user assigned.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteAgreement

Delete the agreement that's specified in the provided AgreementId.

Request Syntax

```
{  
  "AgreementId": "string",  
  "ServerId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

AgreementId

A unique identifier for the agreement. This identifier is returned when you create an agreement.

Type: String

Length Constraints: Fixed length of 19.

Pattern: a-([0-9a-f]{17})

Required: Yes

ServerId

The server identifier associated with the agreement that you are deleting.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)

- [AWS SDK for Ruby V3](#)

DeleteCertificate

Deletes the certificate that's specified in the `CertificateId` parameter.

Request Syntax

```
{  
  "CertificateId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

CertificateId

The identifier of the certificate object that you are deleting.

Type: String

Length Constraints: Fixed length of 22.

Pattern: `cert-([0-9a-f]{17})`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteConnector

Deletes the connector that's specified in the provided `ConnectorId`.

Request Syntax

```
{  
  "ConnectorId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ConnectorId

The unique identifier for the connector.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `c-([0-9a-f]{17})`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteHostKey

Deletes the host key that's specified in the HostKeyId parameter.

Request Syntax

```
{  
  "HostKeyId": "string",  
  "ServerId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

HostKeyId

The identifier of the host key that you are deleting.

Type: String

Length Constraints: Fixed length of 25.

Pattern: hostkey-[0-9a-f]{17}

Required: Yes

ServerId

The identifier of the server that contains the host key that you are deleting.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteProfile

Deletes the profile that's specified in the `ProfileId` parameter.

Request Syntax

```
{  
  "ProfileId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ProfileId

The identifier of the profile that you are deleting.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteServer

Deletes the file transfer protocol-enabled server that you specify.

No response returns from this operation.

Request Syntax

```
{
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ServerId

A unique system-assigned identifier for a server instance.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example deletes a server.

Sample Request

```
{
  "ServerId": "s-01234567890abcdef"
}
```

Example

If successful, nothing is returned.

Sample Response

```
{
```



```
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteSshPublicKey

Deletes a user's Secure Shell (SSH) public key.

Request Syntax

```
{  
  "ServerId": "string",  
  "SshPublicKeyId": "string",  
  "UserName": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ServerId

A system-assigned unique identifier for a file transfer protocol-enabled server instance that has the user assigned to it.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

SshPublicKeyId

A unique identifier used to reference your user's specific SSH key.

Type: String

Length Constraints: Fixed length of 21.

Pattern: key-[0-9a-f]{17}

Required: Yes

UserName

A unique string that identifies a user whose public key is being deleted.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\w][\w@.-]{2,99}`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example deletes a user's SSH public key.

Sample Request

```
{
  "ServerId": "s-01234567890abcdef",
  "SshPublicKeyId": "MyPublicKey",
  "UserName": "my_user"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteUser

Deletes the user belonging to a file transfer protocol-enabled server you specify.

No response returns from this operation.

Note

When you delete a user from a server, the user's information is lost.

Request Syntax

```
{  
  "ServerId": "string",  
  "UserName": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ServerId

A system-assigned unique identifier for a server instance that has the user assigned to it.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

UserName

A unique string that identifies a user that is being deleted from a server.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\w][\w@.-]{2,99}`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example deletes a Transfer Family user.

Sample Request

```
{
  "ServerId": "s-01234567890abcdef",
  "UserNames": "my_user"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteWorkflow

Deletes the specified workflow.

Request Syntax

```
{  
  "WorkflowId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: w-([a-z0-9]{17})

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeAccess

Describes the access that is assigned to the specific file transfer protocol-enabled server, as identified by its `ServerId` property and its `ExternalId`.

The response from this call returns the properties of the access that is associated with the `ServerId` value that was specified.

Request Syntax

```
{
  "ExternalId": "string",
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ExternalId

A unique identifier that is required to identify specific groups within your directory. The users of the group that you associate have access to your Amazon S3 or Amazon EFS resources over the enabled protocols using AWS Transfer Family. If you know the group name, you can view the SID values by running the following command using Windows PowerShell.

```
Get-ADGroup -Filter {samAccountName -like "YourGroupName*"} -Properties
* | Select SamAccountName, ObjectSid
```

In that command, replace *YourGroupName* with the name of your Active Directory group.

The regular expression used to validate this parameter is a string of characters consisting of uppercase and lowercase alphanumeric characters with no spaces. You can also include underscores or any of the following characters: `.,@:/-`

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: S-1-[\d-]+

Required: Yes

ServerId

A system-assigned unique identifier for a server that has this access assigned.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{
  "Access": {
    "ExternalId": "string",
    "HomeDirectory": "string",
    "HomeDirectoryMappings": [
      {
        "Entry": "string",
        "Target": "string",
        "Type": "string"
      }
    ],
    "HomeDirectoryType": "string",
    "Policy": "string",
    "PosixProfile": {
      "Gid": number,
      "SecondaryGids": [ number ],
      "Uid": number
    },
    "Role": "string"
  },
  "ServerId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Access

The external identifier of the server that the access is attached to.

Type: [DescribedAccess](#) object

ServerId

A system-assigned unique identifier for a server that has this access assigned.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeAgreement

Describes the agreement that's identified by the AgreementId.

Request Syntax

```
{  
  "AgreementId": "string",  
  "ServerId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

AgreementId

A unique identifier for the agreement. This identifier is returned when you create an agreement.

Type: String

Length Constraints: Fixed length of 19.

Pattern: a-([0-9a-f]{17})

Required: Yes

ServerId

The server identifier that's associated with the agreement.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{
```

```
"Agreement": {
  "AccessRole": "string",
  "AgreementId": "string",
  "Arn": "string",
  "BaseDirectory": "string",
  "Description": "string",
  "LocalProfileId": "string",
  "PartnerProfileId": "string",
  "ServerId": "string",
  "Status": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Agreement

The details for the specified agreement, returned as a `DescribedAgreement` object.

Type: [DescribedAgreement](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeCertificate

Describes the certificate that's identified by the `CertificateId`.

Request Syntax

```
{  
  "CertificateId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

CertificateId

An array of identifiers for the imported certificates. You use this identifier for working with profiles and partner profiles.

Type: String

Length Constraints: Fixed length of 22.

Pattern: `cert-([0-9a-f]{17})`

Required: Yes

Response Syntax

```
{  
  "Certificate": {  
    "ActiveDate": number,  
    "Arn": "string",  
    "Certificate": "string",  
    "CertificateChain": "string",  
    "CertificateId": "string",  
    "Description": "string",  
    "InactiveDate": number,  
    "NotAfterDate": number,  
    "NotBeforeDate": number,  
  }
```

```
  "Serial": "string",
  "Status": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
  "Type": "string",
  "Usage": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Certificate

The details for the specified certificate, returned as an object.

Type: [DescribedCertificate](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeConnector

Describes the connector that's identified by the ConnectorId.

Request Syntax

```
{  
  "ConnectorId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ConnectorId

The unique identifier for the connector.

Type: String

Length Constraints: Fixed length of 19.

Pattern: c-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{  
  "Connector": {  
    "AccessRole": "string",  
    "Arn": "string",  
    "As2Config": {  
      "BasicAuthSecretId": "string",  
      "Compression": "string",  
      "EncryptionAlgorithm": "string",  
      "LocalProfileId": "string",  
      "MdnResponse": "string",  
      "MdnSigningAlgorithm": "string",
```

```

    "MessageSubject": "string",
    "PartnerProfileId": "string",
    "SigningAlgorithm": "string"
  },
  "ConnectorId": "string",
  "LoggingRole": "string",
  "SecurityPolicyName": "string",
  "ServiceManagedEgressIpAddresses": [ "string" ],
  "SftpConfig": {
    "TrustedHostKeys": [ "string" ],
    "UserSecretId": "string"
  },
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
  "Url": "string"
}

```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Connector

The structure that contains the details of the connector.

Type: [DescribedConnector](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeExecution

You can use `DescribeExecution` to check the details of the execution of the specified workflow.

Note

This API call only returns details for in-progress workflows. If you provide an ID for an execution that is not in progress, or if the execution doesn't match the specified workflow ID, you receive a `ResourceNotFound` exception.

Request Syntax

```
{
  "ExecutionId": "string",
  "WorkflowId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ExecutionId

A unique identifier for the execution of a workflow.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}`

Required: Yes

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: w-([a-z0-9]{17})

Required: Yes

Response Syntax

```
{
  "Execution": {
    "ExecutionId": "string",
    "ExecutionRole": "string",
    "InitialFileLocation": {
      "EfsFileLocation": {
        "FileSystemId": "string",
        "Path": "string"
      },
      "S3FileLocation": {
        "Bucket": "string",
        "Etag": "string",
        "Key": "string",
        "VersionId": "string"
      }
    },
    "LoggingConfiguration": {
      "LoggingRole": "string",
      "LogGroupName": "string"
    },
    "PosixProfile": {
      "Gid": number,
      "SecondaryGids": [ number ],
      "Uid": number
    },
    "Results": {
      "OnExceptionSteps": [
        {
          "Error": {
            "Message": "string",
            "Type": "string"
          },
          "Outputs": "string",
          "StepType": "string"
        }
      ]
    }
  }
}
```



```
    ],
    "Steps": [
      {
        "Error": {
          "Message": "string",
          "Type": "string"
        },
        "Outputs": "string",
        "StepType": "string"
      }
    ]
  },
  "ServiceMetadata": {
    "UserDetails": {
      "ServerId": "string",
      "SessionId": "string",
      "UserName": "string"
    }
  },
  "Status": "string"
},
"WorkflowId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Execution

The structure that contains the details of the workflow' execution.

Type: [DescribedExecution](#) object

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: w-([a-z0-9]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)

- [AWS SDK for Ruby V3](#)

DescribeHostKey

Returns the details of the host key that's specified by the `HostKeyId` and `ServerId`.

Request Syntax

```
{  
  "HostKeyId": "string",  
  "ServerId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

HostKeyId

The identifier of the host key that you want described.

Type: String

Length Constraints: Fixed length of 25.

Pattern: `hostkey-[0-9a-f]{17}`

Required: Yes

ServerId

The identifier of the server that contains the host key that you want described.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Required: Yes

Response Syntax

```
{
```

```
"HostKey": {
  "Arn": "string",
  "DateImported": number,
  "Description": "string",
  "HostKeyFingerprint": "string",
  "HostKeyId": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
  "Type": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

HostKey

Returns the details for the specified host key.

Type: [DescribedHostKey](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeProfile

Returns the details of the profile that's specified by the ProfileId.

Request Syntax

```
{
  "ProfileId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ProfileId

The identifier of the profile that you want described.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{
  "Profile": {
    "Arn": "string",
    "As2Id": "string",
    "CertificateIds": [ "string" ],
    "ProfileId": "string",
    "ProfileType": "string",
    "Tags": [
      {
        "Key": "string",
        "Value": "string"
      }
    ]
  }
}
```

```
    }  
  ]  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Profile

The details of the specified profile, returned as an object.

Type: [DescribedProfile](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeSecurityPolicy

Describes the security policy that is attached to your server or SFTP connector. The response contains a description of the security policy's properties. For more information about security policies, see [Working with security policies for servers](#) or [Working with security policies for SFTP connectors](#).

Request Syntax

```
{  
  "SecurityPolicyName": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

[SecurityPolicyName](#)

Specify the text name of the security policy for which you want the details.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 100.

Pattern: Transfer[A-Za-z0-9]*SecurityPolicy-[A-Za-z0-9-]+

Required: Yes

Response Syntax

```
{  
  "SecurityPolicy": {  
    "Fips": boolean,  
    "Protocols": [ "string" ],  
    "SecurityPolicyName": "string",  
    "SshCiphers": [ "string" ],  
    "SshHostKeyAlgorithms": [ "string" ],  
    "SshKexs": [ "string" ],  
  }
```

```
"SshMacs": [ "string" ],  
"TlsCiphers": [ "string" ],  
"Type": "string"  
}  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

SecurityPolicy

An array containing the properties of the security policy.

Type: [DescribedSecurityPolicy](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example command takes the security policy name as an argument, and returns the algorithms for the specified security policy.

Sample Request

```
aws transfer describe-security-policy --security-policy-name "TransferSecurityPolicy-FIPS-2023-05"
```

Sample Response

```
{
  "SecurityPolicy": {
    "Fips": true,
    "SecurityPolicyName": "TransferSecurityPolicy-FIPS-2023-05",
    "SshCiphers": [
      "aes256-gcm@openssh.com",
      "aes128-gcm@openssh.com",
      "aes256-ctr",
      "aes192-ctr"
    ],
    "SshKexs": [
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group-exchange-sha256"
    ],
    "SshMacs": [
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512-etm@openssh.com"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
    ]
  }
}
```

```
}  
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeServer

Describes a file transfer protocol-enabled server that you specify by passing the `ServerId` parameter.

The response contains a description of a server's properties. When you set `EndpointType` to `VPC`, the response will contain the `EndpointDetails`.

Request Syntax

```
{
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ServerId

A system-assigned unique identifier for a server.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{
  "Server": {
    "Arn": "string",
    "As2ServiceManagedEgressIpAddresses": [ "string" ],
    "Certificate": "string",
    "Domain": "string",
    "EndpointDetails": {
      "AddressAllocationIds": [ "string" ],
```

```

    "SecurityGroupIds": [ "string" ],
    "SubnetIds": [ "string" ],
    "VpcEndpointId": "string",
    "VpcId": "string"
  },
  "EndpointType": "string",
  "HostKeyFingerprint": "string",
  "IdentityProviderDetails": {
    "DirectoryId": "string",
    "Function": "string",
    "InvocationRole": "string",
    "SftpAuthenticationMethods": "string",
    "Url": "string"
  },
  "IdentityProviderType": "string",
  "LoggingRole": "string",
  "PostAuthenticationLoginBanner": "string",
  "PreAuthenticationLoginBanner": "string",
  "ProtocolDetails": {
    "As2Transports": [ "string" ],
    "PassiveIp": "string",
    "SetStatOption": "string",
    "TlsSessionResumptionMode": "string"
  },
  "Protocols": [ "string" ],
  "S3StorageOptions": {
    "DirectoryListingOptimization": "string"
  },
  "SecurityPolicyName": "string",
  "ServerId": "string",
  "State": "string",
  "StructuredLogDestinations": [ "string" ],
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
  "UserCount": number,
  "WorkflowDetails": {
    "OnPartialUpload": [
      {
        "ExecutionRole": "string",
        "WorkflowId": "string"
      }
    ]
  }
}

```

```
    }
  ],
  "OnUpload": [
    {
      "ExecutionRole": "string",
      "WorkflowId": "string"
    }
  ]
}
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Server

An array containing the properties of a server with the `ServerID` you specified.

Type: [DescribedServer](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example returns the properties assigned to a server.

Sample Request

```
{
  "ServerId": "s-01234567890abcdef"
}
```

Example

This example illustrates one usage of DescribeServer.

Sample Response

```
{
  "Server": {
    "Arn": "arn:aws:transfer:us-east-1:176354371281:server/s-01234567890abcdef",
    "EndpointDetails": {
      "AddressAllocationIds": [
        "eipalloc-01a2eabe3c04d5678",
        "eipalloc-102345be"
      ],
      "SubnetIds": [
        "subnet-047eaa7f0187a7cde",
        "subnet-0a2d0f474daffde18"
      ],
      "VpcEndpointId": "vpce-03fe0080e7cb008b8",
      "VpcId": "vpc-09047a51f1c8e1634"
    },
  },
}
```

```
    "EndpointType": "VPC",
    "HostKeyFingerprint": "your host key",
    "IdentityProviderType": "SERVICE_MANAGED",
    "ServerId": "s-01234567890abcdef",
    "State": "ONLINE",
    "Tags": [],
    "UserCount": 0
  }
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeUser

Describes the user assigned to the specific file transfer protocol-enabled server, as identified by its `ServerId` property.

The response from this call returns the properties of the user associated with the `ServerId` value that was specified.

Request Syntax

```
{
  "ServerId": "string",
  "UserName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ServerId

A system-assigned unique identifier for a server that has this user assigned.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Required: Yes

UserName

The name of the user assigned to one or more servers. User names are part of the sign-in credentials to use the AWS Transfer Family service and perform file transfer tasks.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\\w][\\w@.-]{2,99}`

Required: Yes

Response Syntax

```
{
  "ServerId": "string",
  "User": {
    "Arn": "string",
    "HomeDirectory": "string",
    "HomeDirectoryMappings": [
      {
        "Entry": "string",
        "Target": "string",
        "Type": "string"
      }
    ],
    "HomeDirectoryType": "string",
    "Policy": "string",
    "PosixProfile": {
      "Gid": number,
      "SecondaryGids": [ number ],
      "Uid": number
    },
    "Role": "string",
    "SshPublicKeys": [
      {
        "DateImported": number,
        "SshPublicKeyBody": "string",
        "SshPublicKeyId": "string"
      }
    ],
    "Tags": [
      {
        "Key": "string",
        "Value": "string"
      }
    ],
    "UserName": "string"
  }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ServerId

A system-assigned unique identifier for a server that has this user assigned.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

User

An array containing the properties of the Transfer Family user for the `ServerID` value that you specified.

Type: [DescribedUser](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example shows the details for an existing user.

Sample Request

```
aws transfer describe-user --server-id s-1111aaaa2222bbbb3 --user-name bob-test
```

Sample Response

```
{
  "ServerId": "s-1111aaaa2222bbbb3",
  "User": {
    "Arn": "arn:aws:transfer:us-east-1:111122223333:user/s-1111aaaa2222bbbb3/bob-test",
    "HomeDirectory": "/DOC-EXAMPLE-BUCKET",
    "HomeDirectoryType": "PATH",
    "Role": "arn:aws:iam::111122223333:role/bob-role",
    "SshPublicKeys": [
      {
        "DateImported": "2022-03-31T12:27:52.614000-04:00",
        "SshPublicKeyBody": "ssh-rsa AAAAB3NzaC1yc..... bobusa@mycomputer.us-east-1.amazon.com",
        "SshPublicKeyId": "key-abcde12345fghik67"
      }
    ],
    "Tags": [],
    "UserName": "bob-test"
  }
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeWorkflow

Describes the specified workflow.

Request Syntax

```
{  
  "WorkflowId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: w-([a-z0-9]{17})

Required: Yes

Response Syntax

```
{  
  "Workflow": {  
    "Arn": "string",  
    "Description": "string",  
    "OnExceptionSteps": [  
      {  
        "CopyStepDetails": {  
          "DestinationFileLocation": {  
            "EfsFileLocation": {  
              "FileSystemId": "string",  
              "Path": "string"  
            },  
            "S3FileLocation": {
```



```

        "Bucket": "string",
        "Key": "string"
    }
},
"Name": "string",
"OverwriteExisting": "string",
"SourceFileLocation": "string"
},
"CustomStepDetails": {
    "Name": "string",
    "SourceFileLocation": "string",
    "Target": "string",
    "TimeoutSeconds": number
},
"DecryptStepDetails": {
    "DestinationFileLocation": {
        "EfsFileLocation": {
            "FileSystemId": "string",
            "Path": "string"
        },
        "S3FileLocation": {
            "Bucket": "string",
            "Key": "string"
        }
    },
    "Name": "string",
    "OverwriteExisting": "string",
    "SourceFileLocation": "string",
    "Type": "string"
},
"DeleteStepDetails": {
    "Name": "string",
    "SourceFileLocation": "string"
},
"TagStepDetails": {
    "Name": "string",
    "SourceFileLocation": "string",
    "Tags": [
        {
            "Key": "string",
            "Value": "string"
        }
    ]
},

```

```

    "Type": "string"
  }
],
"Steps": [
  {
    "CopyStepDetails": {
      "DestinationFileLocation": {
        "EfsFileLocation": {
          "FileSystemId": "string",
          "Path": "string"
        },
        "S3FileLocation": {
          "Bucket": "string",
          "Key": "string"
        }
      },
      "Name": "string",
      "OverwriteExisting": "string",
      "SourceFileLocation": "string"
    },
    "CustomStepDetails": {
      "Name": "string",
      "SourceFileLocation": "string",
      "Target": "string",
      "TimeoutSeconds": number
    },
    "DecryptStepDetails": {
      "DestinationFileLocation": {
        "EfsFileLocation": {
          "FileSystemId": "string",
          "Path": "string"
        },
        "S3FileLocation": {
          "Bucket": "string",
          "Key": "string"
        }
      },
      "Name": "string",
      "OverwriteExisting": "string",
      "SourceFileLocation": "string",
      "Type": "string"
    },
    "DeleteStepDetails": {
      "Name": "string",

```

```

        "SourceFileLocation": "string"
    },
    "TagStepDetails": {
        "Name": "string",
        "SourceFileLocation": "string",
        "Tags": [
            {
                "Key": "string",
                "Value": "string"
            }
        ]
    },
    "Type": "string"
}
],
"Tags": [
    {
        "Key": "string",
        "Value": "string"
    }
],
"WorkflowId": "string"
}
}

```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Workflow

The structure that contains the details of the workflow.

Type: [DescribedWorkflow](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ImportCertificate

Imports the signing and encryption certificates that you need to create local (AS2) profiles and partner profiles.

Request Syntax

```
{
  "ActiveDate": number,
  "Certificate": "string",
  "CertificateChain": "string",
  "Description": "string",
  "InactiveDate": number,
  "PrivateKey": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
  "Usage": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ActiveDate

An optional date that specifies when the certificate becomes active.

Type: Timestamp

Required: No

Certificate

- For the CLI, provide a file path for a certificate in URI format. For example, `--certificate file://encryption-cert.pem`. Alternatively, you can provide the raw content.
- For the SDK, specify the raw content of a certificate file. For example, `--certificate "`cat encryption-cert.pem`"`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 16384.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF]*`

Required: Yes

CertificateChain

An optional list of certificates that make up the chain for the certificate that's being imported.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2097152.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF]*`

Required: No

Description

A short description that helps identify the certificate.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[\p{Graph}]+`

Required: No

InactiveDate

An optional date that specifies when the certificate becomes inactive.

Type: Timestamp

Required: No

PrivateKey

- For the CLI, provide a file path for a private key in URI format. For example, `--private-key file://encryption-key.pem`. Alternatively, you can provide the raw content of the private key file.
- For the SDK, specify the raw content of a private key file. For example, `--private-key "`cat encryption-key.pem`"`

Type: String

Length Constraints: Minimum length of 1. Maximum length of 16384.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF]*`

Required: No

Tags

Key-value pairs that can be used to group and search for certificates.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

Usage

Specifies how this certificate is used. It can be used in the following ways:

- SIGNING: For signing AS2 messages
- ENCRYPTION: For encrypting AS2 messages
- TLS: For securing AS2 communications sent over HTTPS

Type: String

Valid Values: SIGNING | ENCRYPTION

Required: Yes

Response Syntax

```
{  
  "CertificateId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CertificateId

An array of identifiers for the imported certificates. You use this identifier for working with profiles and partner profiles.

Type: String

Length Constraints: Fixed length of 22.

Pattern: `cert-([0-9a-f]{17})`

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example imports a certificate to use for encryption. In the first command, we provide the contents of the certificate and certificate chain files. Use this format for SDK commands.


```
aws transfer import-certificate --usage ENCRYPTION --certificate "`cat encryption-
cert.pem`" \
  --private-key "`cat encryption-key.pem`" --certificate-chain "`cat root-ca.pem`"
```

Example

The following example is identical to the preceding command, except that we provide the file locations for the private key, certificate, and certificate chain files. This version of the command doesn't work if you are using an SDK.

```
aws transfer import-certificate --usage ENCRYPTION --certificate file://encryption-
cert.pem \
  --private-key file://encryption-key.pem --certificate-chain file://root-ca.pem
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ImportHostKey

Adds a host key to the server that's specified by the `ServerId` parameter.

Request Syntax

```
{
  "Description": "string",
  "HostKeyBody": "string",
  "ServerId": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

Description

The text description that identifies this host key.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Pattern: `[\p{Print}]*`

Required: No

HostKeyBody

The private key portion of an SSH key pair.

AWS Transfer Family accepts RSA, ECDSA, and ED25519 keys.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Required: Yes

ServerId

The identifier of the server that contains the host key that you are importing.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Tags

Key-value pairs that can be used to group and search for host keys.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

Response Syntax

```
{
  "HostKeyId": "string",
  "ServerId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

HostKeyId

Returns the host key identifier for the imported key.

Type: String

Length Constraints: Fixed length of 25.

Pattern: `hostkey-[0-9a-f]{17}`

ServerId

Returns the server identifier that contains the imported key.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ImportSshPublicKey

Adds a Secure Shell (SSH) public key to a Transfer Family user identified by a `UserName` value assigned to the specific file transfer protocol-enabled server, identified by `ServerId`.

The response returns the `UserName` value, the `ServerId` value, and the name of the `SshPublicKeyId`.

Request Syntax

```
{
  "ServerId": "string",
  "SshPublicKeyBody": "string",
  "UserName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ServerId

A system-assigned unique identifier for a server.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

SshPublicKeyBody

The public key portion of an SSH key pair.

AWS Transfer Family accepts RSA, ECDSA, and ED25519 keys.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Required: Yes

UserName

The name of the Transfer Family user that is assigned to one or more servers.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\w][\w@.-]{2,99}`

Required: Yes

Response Syntax

```
{
  "ServerId": "string",
  "SshPublicKeyId": "string",
  "UserName": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ServerId

A system-assigned unique identifier for a server.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

SshPublicKeyId

The name given to a public key by the system that was imported.

Type: String

Length Constraints: Fixed length of 21.

Pattern: key- [0-9a-f]{17}

UserName

A user name assigned to the ServerID value that you specified.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: [\w][\w@.-]{2,99}

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

This command imports an ECDSA key stored in the `id_ecdsa.pub` file.

```
aws transfer import-ssh-public-key --server-id s-021345abcdef6789 --ssh-public-key-body
file://id_ecdsa.pub --user-name jane-doe
```

Example

If you run the previous command, the system returns the following information.

```
{
  "ServerId": "s-021345abcdef6789",
  "SshPublicKeyId": "key-1234567890abcdef0",
  "UserName": "jane-doe"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListAccesses

Lists the details for all the accesses you have on your server.

Request Syntax

```
{
  "MaxResults": number,
  "NextToken": "string",
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

MaxResults

Specifies the maximum number of access SIDs to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

When you can get additional results from the ListAccesses call, a NextToken parameter is returned in the output. You can then pass in a subsequent command to the NextToken parameter to continue listing additional accesses.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

ServerId

A system-assigned unique identifier for a server that has users assigned to it.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{
  "Accesses": [
    {
      "ExternalId": "string",
      "HomeDirectory": "string",
      "HomeDirectoryType": "string",
      "Role": "string"
    }
  ],
  "NextToken": "string",
  "ServerId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Accesses

Returns the accesses and their properties for the `ServerId` value that you specify.

Type: Array of [ListedAccess](#) objects

NextToken

When you can get additional results from the `ListAccesses` call, a `NextToken` parameter is returned in the output. You can then pass in a subsequent command to the `NextToken` parameter to continue listing additional accesses.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

ServerId

A system-assigned unique identifier for a server that has users assigned to it.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The NextToken parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListAgreements

Returns a list of the agreements for the server that's identified by the `ServerId` that you supply. If you want to limit the results to a certain number, supply a value for the `MaxResults` parameter. If you ran the command previously and received a value for `NextToken`, you can supply that value to continue listing agreements from where you left off.

Request Syntax

```
{
  "MaxResults": number,
  "NextToken": "string",
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

[MaxResults](#)

The maximum number of agreements to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

[NextToken](#)

When you can get additional results from the `ListAgreements` call, a `NextToken` parameter is returned in the output. You can then pass in a subsequent command to the `NextToken` parameter to continue listing additional agreements.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

ServerId

The identifier of the server for which you want a list of agreements.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{
  "Agreements": [
    {
      "AgreementId": "string",
      "Arn": "string",
      "Description": "string",
      "LocalProfileId": "string",
      "PartnerProfileId": "string",
      "ServerId": "string",
      "Status": "string"
    }
  ],
  "NextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Agreements

Returns an array, where each item contains the details of an agreement.

Type: Array of [ListedAgreement](#) objects

NextToken

Returns a token that you can use to call `ListAgreements` again and receive additional results, if there are any.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The `NextToken` parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListCertificates

Returns a list of the current certificates that have been imported into AWS Transfer Family. If you want to limit the results to a certain number, supply a value for the `MaxResults` parameter. If you ran the command previously and received a value for the `NextToken` parameter, you can supply that value to continue listing certificates from where you left off.

Request Syntax

```
{  
  "MaxResults": number,  
  "NextToken": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

MaxResults

The maximum number of certificates to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

When you can get additional results from the `ListCertificates` call, a `NextToken` parameter is returned in the output. You can then pass in a subsequent command to the `NextToken` parameter to continue listing additional certificates.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

Response Syntax

```
{
  "Certificates": [
    {
      "ActiveDate": number,
      "Arn": "string",
      "CertificateId": "string",
      "Description": "string",
      "InactiveDate": number,
      "Status": "string",
      "Type": "string",
      "Usage": "string"
    }
  ],
  "NextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Certificates

Returns an array of the certificates that are specified in the `ListCertificates` call.

Type: Array of [ListedCertificate](#) objects

NextToken

Returns the next token, which you can use to list the next certificate.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The NextToken parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListConnectors

Lists the connectors for the specified Region.

Request Syntax

```
{  
  "MaxResults": number,  
  "NextToken": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

MaxResults

The maximum number of connectors to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

When you can get additional results from the ListConnectors call, a NextToken parameter is returned in the output. You can then pass in a subsequent command to the NextToken parameter to continue listing additional connectors.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

Response Syntax

```
{
```

```
"Connectors": [  
  {  
    "Arn": "string",  
    "ConnectorId": "string",  
    "Url": "string"  
  }  
],  
"NextToken": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Connectors

Returns an array, where each item contains the details of a connector.

Type: Array of [ListedConnector](#) objects

NextToken

Returns a token that you can use to call `ListConnectors` again and receive additional results, if there are any.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The `NextToken` parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListExecutions

Lists all in-progress executions for the specified workflow.

Note

If the specified workflow ID cannot be found, `ListExecutions` returns a `ResourceNotFound` exception.

Request Syntax

```
{
  "MaxResults": number,
  "NextToken": "string",
  "WorkflowId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

MaxResults

Specifies the maximum number of executions to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

`ListExecutions` returns the `NextToken` parameter in the output. You can then pass the `NextToken` parameter in a subsequent command to continue listing additional executions.

This is useful for pagination, for instance. If you have 100 executions for a workflow, you might only want to list first 10. If so, call the API by specifying the `max-results`:

```
aws transfer list-executions --max-results 10
```

This returns details for the first 10 executions, as well as the pointer (NextToken) to the eleventh execution. You can now call the API again, supplying the NextToken value you received:

```
aws transfer list-executions --max-results 10 --next-token  
$somePointerReturnedFromPreviousListResult
```

This call returns the next 10 executions, the 11th through the 20th. You can then repeat the call until the details for all 100 executions have been returned.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: w-([a-z0-9]{17})

Required: Yes

Response Syntax

```
{  
  "Executions": [  
    {  
      "ExecutionId": "string",  
      "InitialFileLocation": {  
        "EfsFileLocation": {  
          "FileSystemId": "string",  
          "Path": "string"  
        },  
        "S3FileLocation": {  
          "Bucket": "string",
```

```
        "Etag": "string",
        "Key": "string",
        "VersionId": "string"
    }
},
"ServiceMetadata": {
    "UserDetails": {
        "ServerId": "string",
        "SessionId": "string",
        "UserName": "string"
    }
},
"Status": "string"
}
],
"NextToken": "string",
"WorkflowId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Executions

Returns the details for each execution, in a `ListedExecution` array.

Type: Array of [ListedExecution](#) objects

NextToken

`ListExecutions` returns the `NextToken` parameter in the output. You can then pass the `NextToken` parameter in a subsequent command to continue listing additional executions.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: w-([a-z0-9]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The NextToken parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)

- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListHostKeys

Returns a list of host keys for the server that's specified by the `ServerId` parameter.

Request Syntax

```
{
  "MaxResults": number,
  "NextToken": "string",
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

[MaxResults](#)

The maximum number of host keys to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

[NextToken](#)

When there are additional results that were not returned, a `NextToken` parameter is returned. You can use that value for a subsequent call to `ListHostKeys` to continue listing results.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

[ServerId](#)

The identifier of the server that contains the host keys that you want to view.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{
  "HostKeys": [
    {
      "Arn": "string",
      "DateImported": number,
      "Description": "string",
      "Fingerprint": "string",
      "HostKeyId": "string",
      "Type": "string"
    }
  ],
  "NextToken": "string",
  "ServerId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

HostKeys

Returns an array, where each item contains the details of a host key.

Type: Array of [ListedHostKey](#) objects

NextToken

Returns a token that you can use to call `ListHostKeys` again and receive additional results, if there are any.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

ServerId

Returns the server identifier that contains the listed host keys.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The NextToken parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListProfiles

Returns a list of the profiles for your system. If you want to limit the results to a certain number, supply a value for the `MaxResults` parameter. If you ran the command previously and received a value for `NextToken`, you can supply that value to continue listing profiles from where you left off.

Request Syntax

```
{  
  "MaxResults": number,  
  "NextToken": "string",  
  "ProfileType": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

MaxResults

The maximum number of profiles to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

When there are additional results that were not returned, a `NextToken` parameter is returned. You can use that value for a subsequent call to `ListProfiles` to continue listing results.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

ProfileType

Indicates whether to list only `LOCAL` type profiles or only `PARTNER` type profiles. If not supplied in the request, the command lists all types of profiles.

Type: String

Valid Values: LOCAL | PARTNER

Required: No

Response Syntax

```
{
  "NextToken": "string",
  "Profiles": [
    {
      "Arn": "string",
      "As2Id": "string",
      "ProfileId": "string",
      "ProfileType": "string"
    }
  ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

Returns a token that you can use to call `ListProfiles` again and receive additional results, if there are any.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Profiles

Returns an array, where each item contains the details of a profile.

Type: Array of [ListedProfile](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The NextToken parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListSecurityPolicies

Lists the security policies that are attached to your servers and SFTP connectors. For more information about security policies, see [Working with security policies for servers](#) or [Working with security policies for SFTP connectors](#).

Request Syntax

```
{
  "MaxResults": number,
  "NextToken": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

MaxResults

Specifies the number of security policies to return as a response to the ListSecurityPolicies query.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

When additional results are obtained from the ListSecurityPolicies command, a NextToken parameter is returned in the output. You can then pass the NextToken parameter in a subsequent command to continue listing additional security policies.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

Response Syntax

```
{  
  "NextToken": "string",  
  "SecurityPolicyNames": [ "string" ]  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

When you can get additional results from the `ListSecurityPolicies` operation, a `NextToken` parameter is returned in the output. In a following command, you can pass in the `NextToken` parameter to continue listing security policies.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

SecurityPolicyNames

An array of security policies that were listed.

Type: Array of strings

Length Constraints: Minimum length of 0. Maximum length of 100.

Pattern: `Transfer[A-Za-z0-9]*SecurityPolicy-[A-Za-z0-9-]+`

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The NextToken parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example lists the names for all available security policies.

Sample Request

```
aws transfer list-security-policies
```

Sample Response

```
{
  "SecurityPolicyNames": [
    "TransferSecurityPolicy-2023-05",
    "TransferSecurityPolicy-2022-03",
    "TransferSecurityPolicy-FIPS-2024-01",
    "TransferSecurityPolicy-2024-01",
    "TransferSecurityPolicy-PQ-SSH-FIPS-Experimental-2023-04",
    "TransferSecurityPolicy-PQ-SSH-Experimental-2023-04",
    "TransferSecurityPolicy-FIPS-2020-06",
    "TransferSecurityPolicy-2020-06",
    "TransferSecurityPolicy-2018-11",
    "TransferSecurityPolicy-FIPS-2023-05"
  ]
}
```

```
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListServers

Lists the file transfer protocol-enabled servers that are associated with your AWS account.

Request Syntax

```
{  
  "MaxResults": number,  
  "NextToken": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

MaxResults

Specifies the number of servers to return as a response to the ListServers query.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

When additional results are obtained from the ListServers command, a NextToken parameter is returned in the output. You can then pass the NextToken parameter in a subsequent command to continue listing additional servers.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

Response Syntax

```
{  
  "NextToken": "string",  
}
```

```
"Servers": [  
  {  
    "Arn": "string",  
    "Domain": "string",  
    "EndpointType": "string",  
    "IdentityProviderType": "string",  
    "LoggingRole": "string",  
    "ServerId": "string",  
    "State": "string",  
    "UserCount": number  
  }  
]
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

When you can get additional results from the `ListServers` operation, a `NextToken` parameter is returned in the output. In a following command, you can pass in the `NextToken` parameter to continue listing additional servers.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Servers

An array of servers that were listed.

Type: Array of [ListedServer](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The NextToken parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example lists the servers that exist in your AWS account.

Note that the example NextToken values are not real: they are meant to indicate how to use the parameter.

Sample Request

```
{
  "MaxResults": 1,
  "NextToken": "token-from-previous-API-call"
}
```

Sample Response

```
{
  "NextToken": "another-token-to-continue-listing",
  "Servers": [
    {
      "Arn": "arn:aws:transfer:us-east-1:111112222222:server/s-01234567890abcdef",
      "Domain": "S3",

```

```
    "IdentityProviderType": "SERVICE_MANAGED",
    "EndpointType": "PUBLIC",
    "LoggingRole": "arn:aws:iam::111112222222:role/my-role",
    "ServerId": "s-01234567890abcdef",
    "State": "ONLINE",
    "UserCount": 3
  }
]
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListTagsForResource

Lists all of the tags associated with the Amazon Resource Name (ARN) that you specify. The resource can be a user, server, or role.

Request Syntax

```
{  
  "Arn": "string",  
  "MaxResults": number,  
  "NextToken": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

Arn

Requests the tags associated with a particular Amazon Resource Name (ARN). An ARN is an identifier for a specific AWS resource, such as a server, user, or role.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

MaxResults

Specifies the number of tags to return as a response to the ListTagsForResource request.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

When you request additional results from the `ListTagsForResource` operation, a `NextToken` parameter is returned in the input. You can then pass in a subsequent command to the `NextToken` parameter to continue listing additional tags.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

Response Syntax

```
{
  "Arn": "string",
  "NextToken": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Arn

The ARN you specified to list the tags of.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

NextToken

When you can get additional results from the `ListTagsForResource` call, a `NextToken` parameter is returned in the output. You can then pass in a subsequent command to the `NextToken` parameter to continue listing additional tags.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Tags

Key-value pairs that are assigned to a resource, usually for the purpose of grouping and searching for items. Tags are metadata that you define.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The `NextToken` parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example lists the tags for the resource with the ARN you specified.

Sample Request

```
{
  "Arn": "arn:aws:transfer:us-east-1:176354371281:server/s-01234567890abcdef"
}
```

Example

This example illustrates one usage of ListTagsForResource.

Sample Response

```
{
  "Tags": [
    {
      "Key": "Name",
      "Value": "MyServer"
    }
  ]
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListUsers

Lists the users for a file transfer protocol-enabled server that you specify by passing the `ServerId` parameter.

Request Syntax

```
{
  "MaxResults": number,
  "NextToken": "string",
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

MaxResults

Specifies the number of users to return as a response to the `ListUsers` request.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

If there are additional results from the `ListUsers` call, a `NextToken` parameter is returned in the output. You can then pass the `NextToken` to a subsequent `ListUsers` command, to continue listing additional users.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

ServerId

A system-assigned unique identifier for a server that has users assigned to it.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{
  "NextToken": "string",
  "ServerId": "string",
  "Users": [
    {
      "Arn": "string",
      "HomeDirectory": "string",
      "HomeDirectoryType": "string",
      "Role": "string",
      "SshPublicKeyCount": number,
      "UserName": "string"
    }
  ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

When you can get additional results from the `ListUsers` call, a `NextToken` parameter is returned in the output. You can then pass in a subsequent command to the `NextToken` parameter to continue listing additional users.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

ServerId

A system-assigned unique identifier for a server that the users are assigned to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Users

Returns the Transfer Family users and their properties for the `ServerId` value that you specify.

Type: Array of [ListedUser](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The `NextToken` parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The `ListUsers` API call returns a list of users associated with a server you specify.

Sample Request

```
{
  "MaxResults": 100,
  "NextToken": "eyJNYXJrZXIiOiBudWxsLCAiYm90b1X0cnVuU2F0ZV9hbW91bnQiOiAyfQ==",
  "ServerId": "s-01234567890abcdef"
}
```

Example

This is a sample response for this API call.

Sample Response

```
{
  "NextToken": "eyJNYXJrZXIiOiBudWxsLCAiYm90b1X0cnVuU2F0ZV9hbW91bnQiOiAyfQ==",
  "ServerId": "s-01234567890abcdef",
  "Users": [
    {
      "Arn": "arn:aws:transfer:us-east-1:176354371281:user/s-01234567890abcdef/charlie",
      "HomeDirectory": "/tests/home/charlie",
      "SshPublicKeyCount": 1,
      "Role": "arn:aws:iam::176354371281:role/transfer-role1",
      "Tags": [
        {
          "Key": "Name",
          "Value": "user1"
        }
      ],
      "UserName": "my_user"
    }
  ]
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListWorkflows

Lists all workflows associated with your AWS account for your current region.

Request Syntax

```
{  
  "MaxResults": number,  
  "NextToken": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

MaxResults

Specifies the maximum number of workflows to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

NextToken

ListWorkflows returns the NextToken parameter in the output. You can then pass the NextToken parameter in a subsequent command to continue listing additional workflows.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

Response Syntax

```
{  
  "NextToken": "string",  
}
```

```
"Workflows": [  
  {  
    "Arn": "string",  
    "Description": "string",  
    "WorkflowId": "string"  
  }  
]
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

ListWorkflows returns the NextToken parameter in the output. You can then pass the NextToken parameter in a subsequent command to continue listing additional workflows.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 6144.

Workflows

Returns the Arn, WorkflowId, and Description for each workflow.

Type: Array of [ListedWorkflow](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidNextTokenException

The NextToken parameter that was passed is invalid.

HTTP Status Code: 400

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

SendWorkflowStepState

Sends a callback for asynchronous custom steps.

The `ExecutionId`, `WorkflowId`, and `Token` are passed to the target resource during execution of a custom step of a workflow. You must include those with their callback as well as providing a status.

Request Syntax

```
{
  "ExecutionId": "string",
  "Status": "string",
  "Token": "string",
  "WorkflowId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ExecutionId

A unique identifier for the execution of a workflow.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}`

Required: Yes

Status

Indicates whether the specified step succeeded or failed.

Type: String

Valid Values: SUCCESS | FAILURE

Required: Yes

Token

Used to distinguish between multiple callbacks for multiple Lambda steps within the same execution.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `\w+`

Required: Yes

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `w-([a-z0-9]{17})`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

StartDirectoryListing

Retrieves a list of the contents of a directory from a remote SFTP server. You specify the connector ID, the output path, and the remote directory path. You can also specify the optional `MaxItems` value to control the maximum number of items that are listed from the remote directory. This API returns a list of all files and directories in the remote directory (up to the maximum value), but does not return files or folders in sub-directories. That is, it only returns a list of files and directories one-level deep.

After you receive the listing file, you can provide the files that you want to transfer to the `RetrieveFilePaths` parameter of the `StartFileTransfer` API call.

The naming convention for the output file is `connector-ID-listing-ID.json`. The output file contains the following information:

- `filePath`: the complete path of a remote file, relative to the directory of the listing request for your SFTP connector on the remote server.
- `modifiedTimestamp`: the last time the file was modified, in UTC time format. This field is optional. If the remote file attributes don't contain a timestamp, it is omitted from the file listing.
- `size`: the size of the file, in bytes. This field is optional. If the remote file attributes don't contain a file size, it is omitted from the file listing.
- `path`: the complete path of a remote directory, relative to the directory of the listing request for your SFTP connector on the remote server.
- `truncated`: a flag indicating whether the list output contains all of the items contained in the remote directory or not. If your `Truncated` output value is true, you can increase the value provided in the optional `max-items` input attribute to be able to list more items (up to the maximum allowed list size of 10,000 items).

Request Syntax

```
{
  "ConnectorId": "string",
  "MaxItems": number,
  "OutputDirectoryPath": "string",
  "RemoteDirectoryPath": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ConnectorId

The unique identifier for the connector.

Type: String

Length Constraints: Fixed length of 19.

Pattern: c-([0-9a-f]{17})

Required: Yes

MaxItems

An optional parameter where you can specify the maximum number of file/directory names to retrieve. The default value is 1,000.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 10000.

Required: No

OutputDirectoryPath

Specifies the path (bucket and prefix) in Amazon S3 storage to store the results of the directory listing.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: (.)+

Required: Yes

RemoteDirectoryPath

Specifies the directory on the remote SFTP server for which you want to list its contents.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: (.)+

Required: Yes

Response Syntax

```
{  
  "ListingId": "string",  
  "OutputFileName": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ListingId

Returns a unique identifier for the directory listing call.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 512.

Pattern: [0-9a-zA-Z./-]+

OutputFileName

Returns the file name where the results are stored. This is a combination of the connector ID and the listing ID: <connector-id>-<listing-id>.json.

Type: String

Length Constraints: Minimum length of 26. Maximum length of 537.

Pattern: c-([0-9a-f]{17})-[0-9a-zA-Z./-]+.json

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example lists the contents of the home folder on the remote SFTP server, which is identified by the specified connector. The results are placed into the Amazon S3 location `/DOC-EXAMPLE-BUCKET/connector-files`, and into a file named `c-AAAA1111BBBB2222C-6666abcd-11aa-22bb-cc33-0000aaaa3333.json`.

Sample Request

```
{
  "ConnectorId": "c-AAAA1111BBBB2222C",
  "MaxItems": "10",
```

```
"OutputDirectoryPath": "/DOC-EXAMPLE-BUCKET/connector-files",
"RemoteDirectoryPath": "/home"
}
```

Sample Response

```
{
  "ListingId": "6666abcd-11aa-22bb-cc33-0000aaaa3333",
  "OutputFileName": "c-AAAA1111BBBB2222C-6666abcd-11aa-22bb-cc33-0000aaaa3333.json"
}
```

```
// under bucket "DOC-EXAMPLE-BUCKET"
connector-files/c-AAAA1111BBBB2222C-6666abcd-11aa-22bb-cc33-0000aaaa3333.json
{
  "files": [
    {
      "filePath": "/home/what.txt",
      "modifiedTimestamp": "2024-01-30T20:34:54Z",
      "size" : 2323
    },
    {
      "filePath": "/home/how.pgp",
      "modifiedTimestamp": "2024-01-30T20:34:54Z",
      "size" : 51238
    }
  ],
  "paths": [
    {
      "path": "/home/magic"
    },
    {
      "path": "/home/aws"
    }
  ],
  "truncated": false
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

StartFileTransfer

Begins a file transfer between local AWS storage and a remote AS2 or SFTP server.

- For an AS2 connector, you specify the `ConnectorId` and one or more `SendFilePaths` to identify the files you want to transfer.
- For an SFTP connector, the file transfer can be either outbound or inbound. In both cases, you specify the `ConnectorId`. Depending on the direction of the transfer, you also specify the following items:
 - If you are transferring file from a partner's SFTP server to Amazon Web Services storage, you specify one or more `RetrieveFilePaths` to identify the files you want to transfer, and a `LocalDirectoryPath` to specify the destination folder.
 - If you are transferring file to a partner's SFTP server from AWS storage, you specify one or more `SendFilePaths` to identify the files you want to transfer, and a `RemoteDirectoryPath` to specify the destination folder.

Request Syntax

```
{  
  "ConnectorId": "string",  
  "LocalDirectoryPath": "string",  
  "RemoteDirectoryPath": "string",  
  "RetrieveFilePaths": [ "string" ],  
  "SendFilePaths": [ "string" ]  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ConnectorId

The unique identifier for the connector.

Type: String

Length Constraints: Fixed length of 19.

Pattern: c-([0-9a-f]{17})

Required: Yes

LocalDirectoryPath

For an inbound transfer, the `LocalDirectoryPath` specifies the destination for one or more files that are transferred from the partner's SFTP server.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: (.)+

Required: No

RemoteDirectoryPath

For an outbound transfer, the `RemoteDirectoryPath` specifies the destination for one or more files that are transferred to the partner's SFTP server. If you don't specify a `RemoteDirectoryPath`, the destination for transferred files is the SFTP user's home directory.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: (.)+

Required: No

RetrieveFilePaths

One or more source paths for the partner's SFTP server. Each string represents a source file path for one inbound file transfer.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: (.)+

Required: No

SendFilePaths

One or more source paths for the Amazon S3 storage. Each string represents a source file path for one outbound file transfer. For example, `DOC-EXAMPLE-BUCKET/myfile.txt` .

Note

Replace `DOC-EXAMPLE-BUCKET` with one of your actual buckets.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `(.)+`

Required: No

Response Syntax

```
{
  "TransferId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

TransferId

Returns the unique identifier for the file transfer.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 512.

Pattern: `[0-9a-zA-Z./-]+`

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example starts an AS2 file transfer from a Transfer Family server to a remote trading partner's endpoint. Replace `DOC-EXAMPLE-BUCKET` with one of your actual buckets.

Sample Request

```
{
  "ConnectorId": "c-AAAA1111BBBB2222C",
```



```
"SendFilePaths": [  
  "/DOC-EXAMPLE-BUCKET/myfile-1.txt",  
  "/DOC-EXAMPLE-BUCKET/myfile-2.txt",  
  "/DOC-EXAMPLE-BUCKET/myfile-3.txt"  
]  
}
```

Sample Response

```
{  
  "TransferId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"  
}
```

Example

The following example starts a file transfer from local AWS storage to a remote SFTP server.

Sample Request

```
{  
  "ConnectorId": "c-01234567890abcdef",  
  "SendFilePaths": [  
    "/DOC-EXAMPLE-BUCKET/myfile-1.txt",  
    "/DOC-EXAMPLE-BUCKET/myfile-2.txt",  
    "/DOC-EXAMPLE-BUCKET/myfile-3.txt"  
  ],  
  "RemoteDirectoryPath": "/MySFTPRootFolder/fromTransferFamilyServer"  
}
```

Sample Response

```
{  
  "TransferId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"  
}
```

Example

The following example starts a file transfer from a remote SFTP server to local AWS storage.

Sample Request

```
{
```

```
"ConnectorId": "c-111122223333AAAAA",
"RetrieveFilePaths": [
  "/MySFTPFolder/toTransferFamily/myfile-1.txt",
  "/MySFTPFolder/toTransferFamily/myfile-2.txt",
  "/MySFTPFolder/toTransferFamily/myfile-3.txt"
],
"LocalDirectoryPath": "/DOC-EXAMPLE-BUCKET/mySourceFiles"
}
```

Sample Response

```
{
  "TransferId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

StartServer

Changes the state of a file transfer protocol-enabled server from OFFLINE to ONLINE. It has no impact on a server that is already ONLINE. An ONLINE server can accept and process file transfer jobs.

The state of STARTING indicates that the server is in an intermediate state, either not fully able to respond, or not fully online. The values of START_FAILED can indicate an error condition.

No response is returned from this call.

Request Syntax

```
{
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ServerId

A system-assigned unique identifier for a server that you start.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example starts a server.

Sample Request

```
{
  "ServerId": "s-01234567890abcdef"
}
```

Example

This is a sample response for this API call.

Sample Response

```
{
  "ServerId": "s-01234567890abcdef"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

StopServer

Changes the state of a file transfer protocol-enabled server from ONLINE to OFFLINE. An OFFLINE server cannot accept and process file transfer jobs. Information tied to your server, such as server and user properties, are not affected by stopping your server.

Note

Stopping the server does not reduce or impact your file transfer protocol endpoint billing; you must delete the server to stop being billed.

The state of STOPPING indicates that the server is in an intermediate state, either not fully able to respond, or not fully offline. The values of STOP_FAILED can indicate an error condition.

No response is returned from this call.

Request Syntax

```
{
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ServerId

A system-assigned unique identifier for a server that you stopped.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example stops a server.

Sample Request

```
{
  "ServerId": "s-01234567890abcdef"
}
```

Example

This is a sample response for this API call.

Sample Response

```
{
  "ServerId": "s-01234567890abcdef"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

TagResource

Attaches a key-value pair to a resource, as identified by its Amazon Resource Name (ARN). Resources are users, servers, roles, and other entities.

There is no response returned from this call.

Request Syntax

```
{
  "Arn": "string",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

Arn

An Amazon Resource Name (ARN) for a specific AWS resource, such as a server, user, or role.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

Tags

Key-value pairs assigned to ARNs that you can use to group and search for resources by type. You can attach this metadata to resources (servers, users, workflows, and so on) for any purpose.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example adds a tag to a file transfer protocol-enabled server.

Sample Request

```
{
  "Arn": "arn:aws:transfer:us-east-1:176354371281:server/s-01234567890abcdef",
  "Tags": [
    {
      "Key": "Group",
      "Value": "Europe"
    }
  ]
}
```

Example

This example illustrates one usage of TagResource.

Sample Response

HTTP 200 response with an empty HTTP body.

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

TestConnection

Tests whether your SFTP connector is set up successfully. We highly recommend that you call this operation to test your ability to transfer files between local AWS storage and a trading partner's SFTP server.

Request Syntax

```
{  
  "ConnectorId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ConnectorId

The unique identifier for the connector.

Type: String

Length Constraints: Fixed length of 19.

Pattern: c-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{  
  "ConnectorId": "string",  
  "Status": "string",  
  "StatusMessage": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ConnectorId

Returns the identifier of the connector object that you are testing.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `c-([0-9a-f]{17})`

Status

Returns OK for successful test, or ERROR if the test fails.

Type: String

StatusMessage

Returns `Connection succeeded` if the test is successful. Or, returns a descriptive error message if the test fails. The following list provides troubleshooting details, depending on the error message that you receive.

- Verify that your secret name aligns with the one in Transfer Role permissions.
- Verify the server URL in the connector configuration , and verify that the login credentials work successfully outside of the connector.
- Verify that the secret exists and is formatted correctly.
- Verify that the trusted host key in the connector configuration matches the `ssh-keyscan` output.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example tests the connection to a remote server.

```
aws transfer test-connection --connector-id c-abcd1234567890fff
```

Sample Response

If successful the API call returns the following details.

```
{
  "Status": "OK",
  "StatusMessage": "Connection succeeded"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

TestIdentityProvider

If the `IdentityProviderType` of a file transfer protocol-enabled server is `AWS_DIRECTORY_SERVICE` or `API_Gateway`, tests whether your identity provider is set up successfully. We highly recommend that you call this operation to test your authentication method as soon as you create your server. By doing so, you can troubleshoot issues with the identity provider integration to ensure that your users can successfully use the service.

The `ServerId` and `UserName` parameters are required. The `ServerProtocol`, `SourceIp`, and `UserPassword` are all optional.

Note the following:

- You cannot use `TestIdentityProvider` if the `IdentityProviderType` of your server is `SERVICE_MANAGED`.
- `TestIdentityProvider` does not work with keys: it only accepts passwords.
- `TestIdentityProvider` can test the password operation for a custom Identity Provider that handles keys and passwords.
- If you provide any incorrect values for any parameters, the `Response` field is empty.
- If you provide a server ID for a server that uses service-managed users, you get an error:

```
An error occurred (InvalidRequestException) when calling the
TestIdentityProvider operation: s-server-ID not configured for external
auth
```

- If you enter a Server ID for the `--server-id` parameter that does not identify an actual Transfer server, you receive the following error:

```
An error occurred (ResourceNotFoundException) when calling the
TestIdentityProvider operation: Unknown server.
```

It is possible your sever is in a different region. You can specify a region by adding the following: `--region region-code`, such as `--region us-east-2` to specify a server in **US East (Ohio)**.

Request Syntax

```
{
  "ServerId": "string",
  "ServerProtocol": "string",
```



```
"SourceIp": "string",  
"UserName": "string",  
"UserPassword": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ServerId

A system-assigned identifier for a specific server. That server's user authentication method is tested with a user name and password.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

ServerProtocol

The type of file transfer protocol to be tested.

The available protocols are:

- Secure Shell (SSH) File Transfer Protocol (SFTP)
- File Transfer Protocol Secure (FTPS)
- File Transfer Protocol (FTP)
- Applicability Statement 2 (AS2)

Type: String

Valid Values: SFTP | FTP | FTPS | AS2

Required: No

SourceIp

The source IP address of the account to be tested.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 32.

Pattern: `\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}`

Required: No

[UserName](#)

The name of the account to be tested.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\w][\w@.-]{2,99}`

Required: Yes

[UserPassword](#)

The password of the account to be tested.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Required: No

Response Syntax

```
{
  "Message": "string",
  "Response": "string",
  "StatusCode": number,
  "Url": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Message

A message that indicates whether the test was successful or not.

Note

If an empty string is returned, the most likely cause is that the authentication failed due to an incorrect username or password.

Type: String

Response

The response that is returned from your API Gateway or your Lambda function.

Type: String

StatusCode

The HTTP status code that is the response from your API Gateway or your Lambda function.

Type: Integer

Url

The endpoint of the service used to authenticate a user.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following request returns a message from an identity provider that a user name and password combination is a valid identity to use with AWS Transfer Family.

Sample Request

```
{
  "ServerID": "s-01234567890abcdef",
  "UserName": "my_user",
  "UserPassword": "MyPassword-1"
}
```

Example

The following response shows a sample response for a successful test.

Sample Response

```
"Response":
  {"homeDirectory\":"\mybucket001\", \"homeDirectoryDetails\":null,
  \"homeDirectoryType\":"\PATH\", \"posixProfile\":null,
  \"publicKeys\":"\[ssh-rsa-key]\", \"role\":"\arn:aws:iam::123456789012:role/
  my_role\", \"policy\":null, \"username\":"\transferuser002\",
  \"identityProviderType\":null, \"userConfigMessage\":null)"}}
```

```
"StatusCode": "200",
"Message": ""
```

Example

The following response indicates that the specified user belongs to more than one group that has access.

```
"Response": "",
"StatusCode": 200,
"Message": "More than one associated access found for user's groups."
```

Example

If you have created and configured a custom identity provider by using an API Gateway, you can enter the following command to test your user:

```
aws transfer test-identity-provider --server-id s-0123456789abcdefg --user-
name myuser
```

where *s-0123456789abcdefg* is your transfer server, and *myuser* is the username for your custom user.

If the command succeeds, your response is similar to the following, where:

- AWS account ID is *012345678901*
- User role is *user-role-api-gateway*
- Home directory is *myuser-bucket*
- Public key is *public-key*
- Invocation URL is *invocation-URL*

```
{
  "Response": "{\"Role\": \"arn:aws:iam::012345678901:role/user-role-api-gateway\",
  \"HomeDirectory\": \"/myuser-bucket\", \"PublicKeys\": \"[public-key]\"}\",
  "StatusCode": 200,
  "Message": "",
  "Url": "https://invocation-URL/servers/s-0123456789abcdefg/users/myuser/config"
```

```
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UntagResource

Detaches a key-value pair from a resource, as identified by its Amazon Resource Name (ARN). Resources are users, servers, roles, and other entities.

No response is returned from this call.

Request Syntax

```
{
  "Arn": "string",
  "TagKeys": [ "string" ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

Arn

The value of the resource that will have the tag removed. An Amazon Resource Name (ARN) is an identifier for a specific AWS resource, such as a server, user, or role.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

TagKeys

TagKeys are key-value pairs assigned to ARNs that can be used to group and search for resources by type. This metadata can be attached to resources for any purpose.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Length Constraints: Minimum length of 0. Maximum length of 128.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

Examples

Example

The following example removes a tag of a file transfer protocol-enabled server.

Sample Request


```
{
  "Arn": "arn:aws:transfer:us-east-1:176354371281:server/s-01234567890abcdef",
  "TagKeys": "Europe" ]
}
```

Example

This example illustrates one usage of UntagResource.

Sample Response

HTTP 200 response with an empty HTTP body.

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateAccess

Allows you to update parameters for the access specified in the `ServerID` and `ExternalID` parameters.

Request Syntax

```
{
  "ExternalId": "string",
  "HomeDirectory": "string",
  "HomeDirectoryMappings": [
    {
      "Entry": "string",
      "Target": "string",
      "Type": "string"
    }
  ],
  "HomeDirectoryType": "string",
  "Policy": "string",
  "PosixProfile": {
    "Gid": number,
    "SecondaryGids": [ number ],
    "Uid": number
  },
  "Role": "string",
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ExternalId

A unique identifier that is required to identify specific groups within your directory. The users of the group that you associate have access to your Amazon S3 or Amazon EFS resources over the enabled protocols using AWS Transfer Family. If you know the group name, you can view the SID values by running the following command using Windows PowerShell.

```
Get-ADGroup -Filter {samAccountName -like "YourGroupName*"} -Properties  
* | Select SamAccountName, ObjectSid
```

In that command, replace *YourGroupName* with the name of your Active Directory group.

The regular expression used to validate this parameter is a string of characters consisting of uppercase and lowercase alphanumeric characters with no spaces. You can also include underscores or any of the following characters: =, ., @, /, -

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: S-1-[\d-]+

Required: Yes

HomeDirectory

The landing directory (folder) for a user when they log in to the server using the client.

A HomeDirectory example is `/bucket_name/home/mydirectory`.

Note

The HomeDirectory parameter is only used if HomeDirectoryType is set to PATH.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: (|/.*)

Required: No

HomeDirectoryMappings

Logical directory mappings that specify what Amazon S3 or Amazon EFS paths and keys should be visible to your user and how you want to make them visible. You must specify the Entry and Target pair, where Entry shows how the path is made visible and Target is the actual Amazon S3 or Amazon EFS path. If you only specify a target, it is displayed as is. You also must

ensure that your AWS Identity and Access Management (IAM) role provides access to paths in Target. This value can be set only when `HomeDirectoryType` is set to `LOGICAL`.

The following is an Entry and Target pair example.

```
[ { "Entry": "/directory1", "Target": "/bucket_name/home/mydirectory" } ]
```

In most cases, you can use this value instead of the session policy to lock down your user to the designated home directory ("chroot"). To do this, you can set Entry to / and set Target to the `HomeDirectory` parameter value.

The following is an Entry and Target pair example for chroot.

```
[ { "Entry": "/", "Target": "/bucket_name/home/mydirectory" } ]
```

Type: Array of [HomeDirectoryMapEntry](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50000 items.

Required: No

[HomeDirectoryType](#)

The type of landing directory (folder) that you want your users' home directory to be when they log in to the server. If you set it to `PATH`, the user will see the absolute Amazon S3 bucket or Amazon EFS path as is in their file transfer protocol clients. If you set it to `LOGICAL`, you need to provide mappings in the `HomeDirectoryMappings` for how you want to make Amazon S3 or Amazon EFS paths visible to your users.

Note

If `HomeDirectoryType` is `LOGICAL`, you must provide mappings, using the `HomeDirectoryMappings` parameter. If, on the other hand, `HomeDirectoryType` is `PATH`, you provide an absolute path using the `HomeDirectory` parameter. You cannot have both `HomeDirectory` and `HomeDirectoryMappings` in your template.

Type: String

Valid Values: `PATH` | `LOGICAL`

Required: No

Policy

A session policy for your user so that you can use the same AWS Identity and Access Management (IAM) role across multiple users. This policy scopes down a user's access to portions of their Amazon S3 bucket. Variables that you can use inside this policy include `${Transfer:UserName}`, `${Transfer:HomeDirectory}`, and `${Transfer:HomeBucket}`.

Note

This policy applies only when the domain of `ServerId` is Amazon S3. Amazon EFS does not use session policies.

For session policies, AWS Transfer Family stores the policy as a JSON blob, instead of the Amazon Resource Name (ARN) of the policy. You save the policy as a JSON blob and pass it in the `Policy` argument.

For an example of a session policy, see [Example session policy](#).

For more information, see [AssumeRole](#) in the *AWS Security Token Service API Reference*.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Required: No

PosixProfile

The full POSIX identity, including user ID (`Uid`), group ID (`Gid`), and any secondary groups IDs (`SecondaryGids`), that controls your users' access to your Amazon EFS file systems. The POSIX permissions that are set on files and directories in your file system determine the level of access your users get when transferring files into and out of your Amazon EFS file systems.

Type: [PosixProfile](#) object

Required: No

Role

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that controls your users' access to your Amazon S3 bucket or Amazon EFS file system. The policies attached to this role determine the level of access that you want to provide your users when

transferring files into and out of your Amazon S3 bucket or Amazon EFS file system. The IAM role should also contain a trust relationship that allows the server to access your resources when servicing your users' transfer requests.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

ServerId

A system-assigned unique identifier for a server instance. This is the specific server that you added your user to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Required: Yes

Response Syntax

```
{
  "ExternalId": "string",
  "ServerId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ExternalId

The external identifier of the group whose users have access to your Amazon S3 or Amazon EFS resources over the enabled protocols using AWSTransfer Family.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: S-1-[\d-]+

ServerId

The identifier of the server that the user is attached to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateAgreement

Updates some of the parameters for an existing agreement. Provide the `AgreementId` and the `ServerId` for the agreement that you want to update, along with the new values for the parameters to update.

Request Syntax

```
{
  "AccessRole": "string",
  "AgreementId": "string",
  "BaseDirectory": "string",
  "Description": "string",
  "LocalProfileId": "string",
  "PartnerProfileId": "string",
  "ServerId": "string",
  "Status": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

[AccessRole](#)

Connectors are used to send files using either the AS2 or SFTP protocol. For the access role, provide the Amazon Resource Name (ARN) of the AWS Identity and Access Management role to use.

For AS2 connectors

With AS2, you can send files by calling `StartFileTransfer` and specifying the file paths in the request parameter, `SendFilePaths`. We use the file's parent directory (for example, for `--send-file-paths /bucket/dir/file.txt`, parent directory is `/bucket/dir/`) to temporarily store a processed AS2 message file, store the MDN when we receive them from the partner, and write a final JSON file containing relevant metadata of the transmission. So, the `AccessRole` needs to provide read and write access to the parent directory of the file location used in the `StartFileTransfer` request. Additionally, you need to provide read and write access to the parent directory of the files that you intend to send with `StartFileTransfer`.

If you are using Basic authentication for your AS2 connector, the access role requires the `secretsmanager:GetSecretValue` permission for the secret. If the secret is encrypted using a customer-managed key instead of the AWS managed key in Secrets Manager, then the role also needs the `kms:Decrypt` permission for that key.

For SFTP connectors

Make sure that the access role provides read and write access to the parent directory of the file location that's used in the `StartFileTransfer` request. Additionally, make sure that the role provides `secretsmanager:GetSecretValue` permission to AWS Secrets Manager.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

AgreementId

A unique identifier for the agreement. This identifier is returned when you create an agreement.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `a-([0-9a-f]{17})`

Required: Yes

BaseDirectory

To change the landing directory (folder) for files that are transferred, provide the bucket folder that you want to use; for example, `/DOC-EXAMPLE-BUCKET/home/mydirectory` .

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `(|/.*)`

Required: No

Description

To replace the existing description, provide a short description for the agreement.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[\p{Graph}]+`

Required: No

LocalProfileId

A unique identifier for the AS2 local profile.

To change the local profile identifier, provide a new value here.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `p-([0-9a-f]{17})`

Required: No

PartnerProfileId

A unique identifier for the partner profile. To change the partner profile identifier, provide a new value here.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `p-([0-9a-f]{17})`

Required: No

ServerId

A system-assigned unique identifier for a server instance. This is the specific server that the agreement uses.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Status

You can update the status for the agreement, either activating an inactive agreement or the reverse.

Type: String

Valid Values: ACTIVE | INACTIVE

Required: No

Response Syntax

```
{  
  "AgreementId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

AgreementId

A unique identifier for the agreement. This identifier is returned when you create an agreement.

Type: String

Length Constraints: Fixed length of 19.

Pattern: a-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)

- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateCertificate

Updates the active and inactive dates for a certificate.

Request Syntax

```
{
  "ActiveDate": number,
  "CertificateId": "string",
  "Description": "string",
  "InactiveDate": number
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

ActiveDate

An optional date that specifies when the certificate becomes active.

Type: Timestamp

Required: No

CertificateId

The identifier of the certificate object that you are updating.

Type: String

Length Constraints: Fixed length of 22.

Pattern: cert-([0-9a-f]{17})

Required: Yes

Description

A short description to help identify the certificate.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[\p{Graph}]+`

Required: No

InactiveDate

An optional date that specifies when the certificate becomes inactive.

Type: Timestamp

Required: No

Response Syntax

```
{
  "CertificateId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CertificateId

Returns the identifier of the certificate object that you are updating.

Type: String

Length Constraints: Fixed length of 22.

Pattern: `cert-([0-9a-f]{17})`

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example updates the active date for a certificate, setting the active date to January 16, 2022 at 16:12:07 UTC -5 hours.

Sample Request

```
aws transfer update-certificate --certificate-id c-abcdefg123456hijk --active-date  
2022-01-16T16:12:07-05:00
```

Example

The following is a sample response for this API call.

Sample Response

```
"CertificateId": "c-abcdefg123456hijk"
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateConnector

Updates some of the parameters for an existing connector. Provide the `ConnectorId` for the connector that you want to update, along with the new values for the parameters to update.

Request Syntax

```
{
  "AccessRole": "string",
  "As2Config": {
    "BasicAuthSecretId": "string",
    "Compression": "string",
    "EncryptionAlgorithm": "string",
    "LocalProfileId": "string",
    "MdnResponse": "string",
    "MdnSigningAlgorithm": "string",
    "MessageSubject": "string",
    "PartnerProfileId": "string",
    "SigningAlgorithm": "string"
  },
  "ConnectorId": "string",
  "LoggingRole": "string",
  "SecurityPolicyName": "string",
  "SftpConfig": {
    "TrustedHostKeys": [ "string" ],
    "UserSecretId": "string"
  },
  "Url": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

[AccessRole](#)

Connectors are used to send files using either the AS2 or SFTP protocol. For the access role, provide the Amazon Resource Name (ARN) of the AWS Identity and Access Management role to use.

For AS2 connectors

With AS2, you can send files by calling `StartFileTransfer` and specifying the file paths in the request parameter, `SendFilePaths`. We use the file's parent directory (for example, for `--send-file-paths /bucket/dir/file.txt`, parent directory is `/bucket/dir/`) to temporarily store a processed AS2 message file, store the MDN when we receive them from the partner, and write a final JSON file containing relevant metadata of the transmission. So, the `AccessRole` needs to provide read and write access to the parent directory of the file location used in the `StartFileTransfer` request. Additionally, you need to provide read and write access to the parent directory of the files that you intend to send with `StartFileTransfer`.

If you are using Basic authentication for your AS2 connector, the access role requires the `secretsmanager:GetSecretValue` permission for the secret. If the secret is encrypted using a customer-managed key instead of the AWS managed key in Secrets Manager, then the role also needs the `kms:Decrypt` permission for that key.

For SFTP connectors

Make sure that the access role provides read and write access to the parent directory of the file location that's used in the `StartFileTransfer` request. Additionally, make sure that the role provides `secretsmanager:GetSecretValue` permission to AWS Secrets Manager.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

[As2Config](#)

A structure that contains the parameters for an AS2 connector object.

Type: [As2ConnectorConfig](#) object

Required: No

[ConnectorId](#)

The unique identifier for the connector.

Type: String

Length Constraints: Fixed length of 19.

Pattern: c-([0-9a-f]{17})

Required: Yes

LoggingRole

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that allows a connector to turn on CloudWatch logging for Amazon S3 events. When set, you can view connector activity in your CloudWatch logs.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:.*role/\S+

Required: No

SecurityPolicyName

Specifies the name of the security policy for the connector.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 100.

Pattern: TransferSFTPConnectorSecurityPolicy-[A-Za-z0-9-]+

Required: No

SftpConfig

A structure that contains the parameters for an SFTP connector object.

Type: [SftpConnectorConfig](#) object

Required: No

Url

The URL of the partner's AS2 or SFTP endpoint.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Required: No

Response Syntax

```
{
  "ConnectorId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ConnectorId

Returns the identifier of the connector object that you are updating.

Type: String

Length Constraints: Fixed length of 19.

Pattern: c-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateHostKey

Updates the description for the host key that's specified by the `ServerId` and `HostKeyId` parameters.

Request Syntax

```
{
  "Description": "string",
  "HostKeyId": "string",
  "ServerId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

Description

An updated description for the host key.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Pattern: `[\p{Print}]*`

Required: Yes

HostKeyId

The identifier of the host key that you are updating.

Type: String

Length Constraints: Fixed length of 25.

Pattern: `hostkey-[0-9a-f]{17}`

Required: Yes

ServerId

The identifier of the server that contains the host key that you are updating.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

Response Syntax

```
{
  "HostKeyId": "string",
  "ServerId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

HostKeyId

Returns the host key identifier for the updated host key.

Type: String

Length Constraints: Fixed length of 25.

Pattern: hostkey-[0-9a-f]{17}

ServerId

Returns the server identifier for the server that contains the updated host key.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateProfile

Updates some of the parameters for an existing profile. Provide the `ProfileId` for the profile that you want to update, along with the new values for the parameters to update.

Request Syntax

```
{  
  "CertificateIds": [ "string" ],  
  "ProfileId": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

CertificateIds

An array of identifiers for the imported certificates. You use this identifier for working with profiles and partner profiles.

Type: Array of strings

Length Constraints: Fixed length of 22.

Pattern: `cert-([0-9a-f]{17})`

Required: No

ProfileId

The identifier of the profile object that you are updating.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `p-([0-9a-f]{17})`

Required: Yes

Response Syntax

```
{  
  "ProfileId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ProfileId

Returns the identifier for the profile that's being updated.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateServer

Updates the file transfer protocol-enabled server's properties after that server has been created.

The UpdateServer call returns the ServerId of the server you updated.

Request Syntax

```
{
  "Certificate": "string",
  "EndpointDetails": {
    "AddressAllocationIds": [ "string" ],
    "SecurityGroupIds": [ "string" ],
    "SubnetIds": [ "string" ],
    "VpcEndpointId": "string",
    "VpcId": "string"
  },
  "EndpointType": "string",
  "HostKey": "string",
  "IdentityProviderDetails": {
    "DirectoryId": "string",
    "Function": "string",
    "InvocationRole": "string",
    "SftpAuthenticationMethods": "string",
    "Url": "string"
  },
  "LoggingRole": "string",
  "PostAuthenticationLoginBanner": "string",
  "PreAuthenticationLoginBanner": "string",
  "ProtocolDetails": {
    "As2Transports": [ "string" ],
    "PassiveIp": "string",
    "SetStatOption": "string",
    "TlsSessionResumptionMode": "string"
  },
  "Protocols": [ "string" ],
  "S3StorageOptions": {
    "DirectoryListingOptimization": "string"
  },
  "SecurityPolicyName": "string",
  "ServerId": "string",
  "StructuredLogDestinations": [ "string" ],
  "WorkflowDetails": {
```

```
    "OnPartialUpload": [
      {
        "ExecutionRole": "string",
        "WorkflowId": "string"
      }
    ],
    "OnUpload": [
      {
        "ExecutionRole": "string",
        "WorkflowId": "string"
      }
    ]
  }
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

Certificate

The Amazon Resource Name (ARN) of the AWSCertificate Manager (ACM) certificate. Required when `Protocols` is set to `FTPS`.

To request a new public certificate, see [Request a public certificate](#) in the *AWSCertificate Manager User Guide*.

To import an existing certificate into ACM, see [Importing certificates into ACM](#) in the *AWSCertificate Manager User Guide*.

To request a private certificate to use `FTPS` through private IP addresses, see [Request a private certificate](#) in the *AWSCertificate Manager User Guide*.

Certificates with the following cryptographic algorithms and key sizes are supported:

- 2048-bit RSA (RSA_2048)
- 4096-bit RSA (RSA_4096)
- Elliptic Prime Curve 256 bit (EC_prime256v1)
- Elliptic Prime Curve 384 bit (EC_secp384r1)
- Elliptic Prime Curve 521 bit (EC_secp521r1)

Note

The certificate must be a valid SSL/TLS X.509 version 3 certificate with FQDN or IP address specified and information about the issuer.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1600.

Required: No

EndpointDetails

The virtual private cloud (VPC) endpoint settings that are configured for your server. When you host your endpoint within your VPC, you can make your endpoint accessible only to resources within your VPC, or you can attach Elastic IP addresses and make your endpoint accessible to clients over the internet. Your VPC's default security groups are automatically assigned to your endpoint.

Type: [EndpointDetails](#) object

Required: No

EndpointType

The type of endpoint that you want your server to use. You can choose to make your server's endpoint publicly accessible (PUBLIC) or host it inside your VPC. With an endpoint that is hosted in a VPC, you can restrict access to your server and resources only within your VPC or choose to make it internet facing by attaching Elastic IP addresses directly to it.

Note

After May 19, 2021, you won't be able to create a server using `EndpointType=VPC_ENDPOINT` in your AWS account if your account hasn't already done so before May 19, 2021. If you have already created servers with `EndpointType=VPC_ENDPOINT` in your AWS account on or before May 19, 2021, you will not be affected. After this date, use `EndpointType=VPC`.

For more information, see [Discontinuing the use of VPC_ENDPOINT](#).

It is recommended that you use VPC as the `EndpointType`. With this endpoint type, you have the option to directly associate up to three Elastic IPv4 addresses (BYO IP

included) with your server's endpoint and use VPC security groups to restrict traffic by the client's public IP address. This is not possible with `EndpointType` set to `VPC_ENDPOINT`.

Type: String

Valid Values: PUBLIC | VPC | VPC_ENDPOINT

Required: No

HostKey

The RSA, ECDSA, or ED25519 private key to use for your SFTP-enabled server. You can add multiple host keys, in case you want to rotate keys, or have a set of active keys that use different algorithms.

Use the following command to generate an RSA 2048 bit key with no passphrase:

```
ssh-keygen -t rsa -b 2048 -N "" -m PEM -f my-new-server-key.
```

Use a minimum value of 2048 for the `-b` option. You can create a stronger key by using 3072 or 4096.

Use the following command to generate an ECDSA 256 bit key with no passphrase:

```
ssh-keygen -t ecdsa -b 256 -N "" -m PEM -f my-new-server-key.
```

Valid values for the `-b` option for ECDSA are 256, 384, and 521.

Use the following command to generate an ED25519 key with no passphrase:

```
ssh-keygen -t ed25519 -N "" -f my-new-server-key.
```

For all of these commands, you can replace *my-new-server-key* with a string of your choice.

Important

If you aren't planning to migrate existing users from an existing SFTP-enabled server to a new server, don't update the host key. Accidentally changing a server's host key can be disruptive.

For more information, see [Update host keys for your SFTP-enabled server](#) in the *AWS Transfer Family User Guide*.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Required: No

IdentityProviderDetails

An array containing all of the information required to call a customer's authentication API method.

Type: [IdentityProviderDetails](#) object

Required: No

LoggingRole

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that allows a server to turn on Amazon CloudWatch logging for Amazon S3 or Amazon EFSEvents. When set, you can view user activity in your CloudWatch logs.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Pattern: (`|arn:.*role/\S+`)

Required: No

PostAuthenticationLoginBanner

Specifies a string to display when users connect to a server. This string is displayed after the user authenticates.

Note

The SFTP protocol does not support post-authentication display banners.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Pattern: `[\x09-\x0D\x20-\x7E]*`

Required: No

PreAuthenticationLoginBanner

Specifies a string to display when users connect to a server. This string is displayed before the user authenticates. For example, the following banner displays details about using the system:

```
This system is for the use of authorized users only. Individuals using
this computer system without authority, or in excess of their authority,
are subject to having all of their activities on this system monitored
and recorded by system personnel.
```

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Pattern: `[\x09-\x0D\x20-\x7E]*`

Required: No

ProtocolDetails

The protocol settings that are configured for your server.

- To indicate passive mode (for FTP and FTPS protocols), use the `PassiveIp` parameter. Enter a single dotted-quad IPv4 address, such as the external IP address of a firewall, router, or load balancer.
- To ignore the error that is generated when the client attempts to use the `SETSTAT` command on a file that you are uploading to an Amazon S3 bucket, use the `SetStatOption` parameter. To have the AWS Transfer Family server ignore the `SETSTAT` command and upload files without needing to make any changes to your SFTP client, set the value to `ENABLE_NO_OP`. If you set the `SetStatOption` parameter to `ENABLE_NO_OP`, Transfer Family generates a log entry to Amazon CloudWatch Logs, so that you can determine when the client is making a `SETSTAT` call.
- To determine whether your AWS Transfer Family server resumes recent, negotiated sessions through a unique session ID, use the `TlsSessionResumptionMode` parameter.
- `As2Transports` indicates the transport method for the AS2 messages. Currently, only HTTP is supported.

Type: [ProtocolDetails](#) object

Required: No

Protocols

Specifies the file transfer protocol or protocols over which your file transfer protocol client can connect to your server's endpoint. The available protocols are:

- SFTP (Secure Shell (SSH) File Transfer Protocol): File transfer over SSH
- FTPS (File Transfer Protocol Secure): File transfer with TLS encryption
- FTP (File Transfer Protocol): Unencrypted file transfer
- AS2 (Applicability Statement 2): used for transporting structured business-to-business data

Note

- If you select FTPS, you must choose a certificate stored in AWS Certificate Manager (ACM) which is used to identify your server when clients connect to it over FTPS.
- If `Protocol` includes either FTP or FTPS, then the `EndpointType` must be VPC and the `IdentityProviderType` must be either `AWS_DIRECTORY_SERVICE`, `AWS_LAMBDA`, or `API_GATEWAY`.
- If `Protocol` includes FTP, then `AddressAllocationIds` cannot be associated.
- If `Protocol` is set only to SFTP, the `EndpointType` can be set to PUBLIC and the `IdentityProviderType` can be set any of the supported identity types: `SERVICE_MANAGED`, `AWS_DIRECTORY_SERVICE`, `AWS_LAMBDA`, or `API_GATEWAY`.
- If `Protocol` includes AS2, then the `EndpointType` must be VPC, and domain must be Amazon S3.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 4 items.

Valid Values: SFTP | FTP | FTPS | AS2

Required: No

S3StorageOptions

Specifies whether or not performance for your Amazon S3 directories is optimized. This is disabled by default.

By default, home directory mappings have a TYPE of DIRECTORY. If you enable this option, you would then need to explicitly set the HomeDirectoryMapEntry Type to FILE if you want a mapping to have a file target.

Type: [S3StorageOptions](#) object

Required: No

SecurityPolicyName

Specifies the name of the security policy for the server.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 100.

Pattern: Transfer[A-Za-z0-9]*SecurityPolicy-[A-Za-z0-9-]+

Required: No

ServerId

A system-assigned unique identifier for a server instance that the Transfer Family user is assigned to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: Yes

StructuredLogDestinations

Specifies the log groups to which your server logs are sent.

To specify a log group, you must provide the ARN for an existing log group. In this case, the format of the log group is as follows:

```
arn:aws:logs:region-name:amazon-account-id:log-group:log-group-name:*
```

For example, `arn:aws:logs:us-east-1:111122223333:log-group:mytestgroup:*`

If you have previously specified a log group for a server, you can clear it, and in effect turn off structured logging, by providing an empty value for this parameter in an `update-server` call. For example:

```
update-server --server-id s-1234567890abcdef0 --structured-log-destinations
```

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 1 item.

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: No

WorkflowDetails

Specifies the workflow ID for the workflow to assign and the execution role that's used for executing the workflow.

In addition to a workflow to execute when a file is uploaded completely, `WorkflowDetails` can also contain a workflow ID (and execution role) for a workflow to execute on partial upload. A partial upload occurs when the server session disconnects while the file is still being uploaded.

To remove an associated workflow from a server, you can provide an empty `OnUpload` object, as in the following example.

```
aws transfer update-server --server-id s-01234567890abcdef --workflow-details '{"OnUpload":[]}'
```

Type: [WorkflowDetails](#) object

Required: No

Response Syntax

```
{  
  "ServerId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ServerId

A system-assigned unique identifier for a server that the Transfer Family user is assigned to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

ConflictException

This exception is thrown when the `UpdateServer` is called for a file transfer protocol-enabled server that has VPC as the endpoint type and the server's `VpcEndpointID` is not in the available state.

HTTP Status Code: 400

InternalServiceError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceExistsException

The requested resource does not exist, or exists in a region other than the one specified for the command.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example updates the role of a server.

Sample Request

```
{
  "EndpointDetails": {
    "VpcEndpointId": "vpce-01234f056f3g13",
    "LoggingRole": "CloudWatchS3Events",
    "ServerId": "s-01234567890abcdef"
  }
}
```

Example

The following example removes any associated workflows from the server.

Sample Request

```
aws transfer update-server --server-id s-01234567890abcdef --workflow-details
 '{"OnUpload":[]}'
```

Example

This is a sample response for this API call.

Sample Response

```
{
  "ServerId": "s-01234567890abcdef"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateUser

Assigns new properties to a user. Parameters you pass modify any or all of the following: the home directory, role, and policy for the `UserName` and `ServerId` you specify.

The response returns the `ServerId` and the `UserName` for the updated user.

In the console, you can select *Restricted* when you create or update a user. This ensures that the user can't access anything outside of their home directory. The programmatic way to configure this behavior is to update the user. Set their `HomeDirectoryType` to `LOGICAL`, and specify `HomeDirectoryMappings` with `Entry` as root (`/`) and `Target` as their home directory.

For example, if the user's home directory is `/test/admin-user`, the following command updates the user so that their configuration in the console shows the *Restricted* flag as selected.

```
aws transfer update-user --server-id <server-id> --user-name admin-user --home-directory-type LOGICAL --home-directory-mappings "[{\\"Entry\\":\\"/\\", \\"Target\\":\\"/test/admin-user\\"}]"
```

Request Syntax

```
{
  "HomeDirectory": "string",
  "HomeDirectoryMappings": [
    {
      "Entry": "string",
      "Target": "string",
      "Type": "string"
    }
  ],
  "HomeDirectoryType": "string",
  "Policy": "string",
  "PosixProfile": {
    "Gid": number,
    "SecondaryGids": [ number ],
    "Uid": number
  },
  "Role": "string",
  "ServerId": "string",
  "UserName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

HomeDirectory

The landing directory (folder) for a user when they log in to the server using the client.

A HomeDirectory example is `/bucket_name/home/mydirectory`.

Note

The HomeDirectory parameter is only used if HomeDirectoryType is set to PATH.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: (|/.*)

Required: No

HomeDirectoryMappings

Logical directory mappings that specify what Amazon S3 or Amazon EFS paths and keys should be visible to your user and how you want to make them visible. You must specify the Entry and Target pair, where Entry shows how the path is made visible and Target is the actual Amazon S3 or Amazon EFS path. If you only specify a target, it is displayed as is. You also must ensure that your AWS Identity and Access Management (IAM) role provides access to paths in Target. This value can be set only when HomeDirectoryType is set to *LOGICAL*.

The following is an Entry and Target pair example.

```
[ { "Entry": "/directory1", "Target": "/bucket_name/home/mydirectory" } ]
```

In most cases, you can use this value instead of the session policy to lock down your user to the designated home directory ("chroot"). To do this, you can set Entry to '/' and set Target to the HomeDirectory parameter value.

The following is an Entry and Target pair example for chroot.

```
[ { "Entry": "/", "Target": "/bucket_name/home/mydirectory" } ]
```

Type: Array of [HomeDirectoryMapEntry](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50000 items.

Required: No

[HomeDirectoryType](#)

The type of landing directory (folder) that you want your users' home directory to be when they log in to the server. If you set it to PATH, the user will see the absolute Amazon S3 bucket or Amazon EFS path as is in their file transfer protocol clients. If you set it to LOGICAL, you need to provide mappings in the HomeDirectoryMappings for how you want to make Amazon S3 or Amazon EFS paths visible to your users.

Note

If HomeDirectoryType is LOGICAL, you must provide mappings, using the HomeDirectoryMappings parameter. If, on the other hand, HomeDirectoryType is PATH, you provide an absolute path using the HomeDirectory parameter. You cannot have both HomeDirectory and HomeDirectoryMappings in your template.

Type: String

Valid Values: PATH | LOGICAL

Required: No

[Policy](#)

A session policy for your user so that you can use the same AWS Identity and Access Management (IAM) role across multiple users. This policy scopes down a user's access to portions of their Amazon S3 bucket. Variables that you can use inside this policy include `${Transfer:UserName}`, `${Transfer:HomeDirectory}`, and `${Transfer:HomeBucket}`.

Note

This policy applies only when the domain of ServerId is Amazon S3. Amazon EFS does not use session policies.

For session policies, AWS Transfer Family stores the policy as a JSON blob, instead of the Amazon Resource Name (ARN) of the policy. You save the policy as a JSON blob and pass it in the `Policy` argument.

For an example of a session policy, see [Example session policy](#).

For more information, see [AssumeRole](#) in the *AWS Security Token Service API Reference*.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Required: No

PosixProfile

Specifies the full POSIX identity, including user ID (`Uid`), group ID (`Gid`), and any secondary groups IDs (`SecondaryGids`), that controls your users' access to your Amazon Elastic File Systems (Amazon EFS). The POSIX permissions that are set on files and directories in your file system determines the level of access your users get when transferring files into and out of your Amazon EFS file systems.

Type: [PosixProfile](#) object

Required: No

Role

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that controls your users' access to your Amazon S3 bucket or Amazon EFS file system. The policies attached to this role determine the level of access that you want to provide your users when transferring files into and out of your Amazon S3 bucket or Amazon EFS file system. The IAM role should also contain a trust relationship that allows the server to access your resources when servicing your users' transfer requests.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

ServerId

A system-assigned unique identifier for a Transfer Family server instance that the user is assigned to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Required: Yes

UserName

A unique string that identifies a user and is associated with a server as specified by the `ServerId`. This user name must be a minimum of 3 and a maximum of 100 characters long. The following are valid characters: a-z, A-Z, 0-9, underscore '_', hyphen '-', period '.', and at sign '@'. The user name can't start with a hyphen, period, or at sign.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\w][\w@.-]{2,99}`

Required: Yes

Response Syntax

```
{
  "ServerId": "string",
  "UserName": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ServerId

A system-assigned unique identifier for a Transfer Family server instance that the account is assigned to.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

UserName

The unique identifier for a user that is assigned to a server instance that was specified in the request.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: [\\w][\\w@.-]{2,99}

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

InternalServerError

This exception is thrown when an error occurs in the AWS Transfer Family service.

HTTP Status Code: 500

InvalidRequestException

This exception is thrown when the client submits a malformed request.

HTTP Status Code: 400

ResourceNotFoundException

This exception is thrown when a resource is not found by the AWSTransfer Family service.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed because the AWSTransfer Family service is not available.

HTTP Status Code: 500

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

Examples

Example

The following example updates a Transfer Family user.

Sample Request

```
{
  "HomeDirectory": "/bucket2/documentation",
  "HomeDirectoryMappings": [
    {
      "Entry": "/directory1",
      "Target": "/bucket_name/home/mydirectory"
    }
  ],
  "HomeDirectoryType": "PATH",
  "Role": "AssumeRole",
  "ServerId": "s-01234567890abcdef",
  "UserName": "my_user"
}
```

Example

This is a sample response for this API call.

Sample Response

```
{
  "ServerId": "s-01234567890abcdef",
  "UserName": "my_user"
}
```

```
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript V3](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

Data Types

The following data types are supported:

- [As2ConnectorConfig](#)
- [CopyStepDetails](#)
- [CustomStepDetails](#)
- [DecryptStepDetails](#)
- [DeleteStepDetails](#)
- [DescribedAccess](#)
- [DescribedAgreement](#)
- [DescribedCertificate](#)
- [DescribedConnector](#)
- [DescribedExecution](#)
- [DescribedHostKey](#)
- [DescribedProfile](#)

- [DescribedSecurityPolicy](#)
- [DescribedServer](#)
- [DescribedUser](#)
- [DescribedWorkflow](#)
- [EfsFileLocation](#)
- [EndpointDetails](#)
- [ExecutionError](#)
- [ExecutionResults](#)
- [ExecutionStepResult](#)
- [FileLocation](#)
- [HomeDirectoryMapEntry](#)
- [IdentityProviderDetails](#)
- [InputFileLocation](#)
- [ListedAccess](#)
- [ListedAgreement](#)
- [ListedCertificate](#)
- [ListedConnector](#)
- [ListedExecution](#)
- [ListedHostKey](#)
- [ListedProfile](#)
- [ListedServer](#)
- [ListedUser](#)
- [ListedWorkflow](#)
- [LoggingConfiguration](#)
- [PosixProfile](#)
- [ProtocolDetails](#)
- [S3FileLocation](#)
- [S3InputFileLocation](#)
- [S3StorageOptions](#)
- [S3Tag](#)

- [ServiceMetadata](#)
- [SftpConnectorConfig](#)
- [SshPublicKey](#)
- [Tag](#)
- [TagStepDetails](#)
- [UserDetails](#)
- [WorkflowDetail](#)
- [WorkflowDetails](#)
- [WorkflowStep](#)

As2ConnectorConfig

Contains the details for an AS2 connector object. The connector object is used for AS2 outbound processes, to connect the AWS Transfer Family customer with the trading partner.

Contents

BasicAuthSecretId

Provides Basic authentication support to the AS2 Connectors API. To use Basic authentication, you must provide the name or Amazon Resource Name (ARN) of a secret in AWS Secrets Manager.

The default value for this parameter is `null`, which indicates that Basic authentication is not enabled for the connector.

If the connector should use Basic authentication, the secret needs to be in the following format:

```
{ "Username": "user-name", "Password": "user-password" }
```

Replace `user-name` and `user-password` with the credentials for the actual user that is being authenticated.

Note the following:

- You are storing these credentials in Secrets Manager, *not passing them directly* into this API.
- If you are using the API, SDKs, or CloudFormation to configure your connector, then you must create the secret before you can enable Basic authentication. However, if you are using the AWS management console, you can have the system create the secret for you.

If you have previously enabled Basic authentication for a connector, you can disable it by using the `UpdateConnector` API call. For example, if you are using the CLI, you can run the following command to remove Basic authentication:

```
update-connector --connector-id my-connector-id --as2-config  
'BasicAuthSecretId=""'
```

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Required: No

Compression

Specifies whether the AS2 file is compressed.

Type: String

Valid Values: ZLIB | DISABLED

Required: No

EncryptionAlgorithm

The algorithm that is used to encrypt the file.

Note the following:

- Do not use the DES_EDE3_CBC algorithm unless you must support a legacy client that requires it, as it is a weak encryption algorithm.
- You can only specify NONE if the URL for your connector uses HTTPS. Using HTTPS ensures that no traffic is sent in clear text.

Type: String

Valid Values: AES128_CBC | AES192_CBC | AES256_CBC | DES_EDE3_CBC | NONE

Required: No

LocalProfileId

A unique identifier for the AS2 local profile.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Required: No

MdnResponse

Used for outbound requests (from an AWS Transfer Family server to a partner AS2 server) to determine whether the partner response for transfers is synchronous or asynchronous. Specify either of the following values:

- SYNC: The system expects a synchronous MDN response, confirming that the file was transferred successfully (or not).
- NONE: Specifies that no MDN response is required.

Type: String

Valid Values: SYNC | NONE

Required: No

MdnSigningAlgorithm

The signing algorithm for the MDN response.

Note

If set to DEFAULT (or not set at all), the value for SigningAlgorithm is used.

Type: String

Valid Values: SHA256 | SHA384 | SHA512 | SHA1 | NONE | DEFAULT

Required: No

MessageSubject

Used as the Subject HTTP header attribute in AS2 messages that are being sent with the connector.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\p{Print}\p{Blank}]+`

Required: No

PartnerProfileId

A unique identifier for the partner profile for the connector.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Required: No

SigningAlgorithm

The algorithm that is used to sign the AS2 messages sent with the connector.

Type: String

Valid Values: SHA256 | SHA384 | SHA512 | SHA1 | NONE

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

CopyStepDetails

Each step type has its own StepDetails structure.

Contents

DestinationFileLocation

Specifies the location for the file being copied. Use `${Transfer:UserName}` or `${Transfer:UploadDate}` in this field to parametrize the destination prefix by username or uploaded date.

- Set the value of `DestinationFileLocation` to `${Transfer:UserName}` to copy uploaded files to an Amazon S3 bucket that is prefixed with the name of the Transfer Family user that uploaded the file.
- Set the value of `DestinationFileLocation` to `${Transfer:UploadDate}` to copy uploaded files to an Amazon S3 bucket that is prefixed with the date of the upload.

Note

The system resolves `UploadDate` to a date format of `YYYY-MM-DD`, based on the date the file is uploaded in UTC.

Type: [InputFileLocation](#) object

Required: No

Name

The name of the step, used as an identifier.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 30.

Pattern: `[\w-]*`

Required: No

OverwriteExisting

A flag that indicates whether to overwrite an existing file of the same name. The default is `FALSE`.

If the workflow is processing a file that has the same name as an existing file, the behavior is as follows:

- If `OverwriteExisting` is `TRUE`, the existing file is replaced with the file being processed.
- If `OverwriteExisting` is `FALSE`, nothing happens, and the workflow processing stops.

Type: String

Valid Values: `TRUE` | `FALSE`

Required: No

SourceFileLocation

Specifies which file to use as input to the workflow step: either the output from the previous step, or the originally uploaded file for the workflow.

- To use the previous file as the input, enter `${previous.file}`. In this case, this workflow step uses the output file from the previous workflow step as input. This is the default value.
- To use the originally uploaded file location as input for this step, enter `${original.file}`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `^\$\\{(\w+.\w+)\}`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

CustomStepDetails

Each step type has its own StepDetails structure.

Contents

Name

The name of the step, used as an identifier.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 30.

Pattern: `[\w-]*`

Required: No

SourceFileLocation

Specifies which file to use as input to the workflow step: either the output from the previous step, or the originally uploaded file for the workflow.

- To use the previous file as the input, enter `${previous.file}`. In this case, this workflow step uses the output file from the previous workflow step as input. This is the default value.
- To use the originally uploaded file location as input for this step, enter `${original.file}`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `\$\{(\w+.\w+)\}`

Required: No

Target

The ARN for the Lambda function that is being called.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 170.

Pattern: `arn:[a-z-]+:lambda:.*`

Required: No

TimeoutSeconds

Timeout, in seconds, for the step.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1800.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DecryptStepDetails

Each step type has its own `StepDetails` structure.

Contents

DestinationFileLocation

Specifies the location for the file being decrypted. Use `${Transfer:UserName}` or `${Transfer:UploadDate}` in this field to parametrize the destination prefix by username or uploaded date.

- Set the value of `DestinationFileLocation` to `${Transfer:UserName}` to decrypt uploaded files to an Amazon S3 bucket that is prefixed with the name of the Transfer Family user that uploaded the file.
- Set the value of `DestinationFileLocation` to `${Transfer:UploadDate}` to decrypt uploaded files to an Amazon S3 bucket that is prefixed with the date of the upload.

Note

The system resolves `UploadDate` to a date format of `YYYY-MM-DD`, based on the date the file is uploaded in UTC.

Type: [InputFileLocation](#) object

Required: Yes

Type

The type of encryption used. Currently, this value must be PGP.

Type: String

Valid Values: PGP

Required: Yes

Name

The name of the step, used as an identifier.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 30.

Pattern: `[\w-]*`

Required: No

OverwriteExisting

A flag that indicates whether to overwrite an existing file of the same name. The default is FALSE.

If the workflow is processing a file that has the same name as an existing file, the behavior is as follows:

- If `OverwriteExisting` is TRUE, the existing file is replaced with the file being processed.
- If `OverwriteExisting` is FALSE, nothing happens, and the workflow processing stops.

Type: String

Valid Values: TRUE | FALSE

Required: No

SourceFileLocation

Specifies which file to use as input to the workflow step: either the output from the previous step, or the originally uploaded file for the workflow.

- To use the previous file as the input, enter `${previous.file}`. In this case, this workflow step uses the output file from the previous workflow step as input. This is the default value.
- To use the originally uploaded file location as input for this step, enter `${original.file}`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `\$\{(\w+ .)+\w+\}`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DeleteStepDetails

The name of the step, used to identify the delete step.

Contents

Name

The name of the step, used as an identifier.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 30.

Pattern: `[\w-]*`

Required: No

SourceFileLocation

Specifies which file to use as input to the workflow step: either the output from the previous step, or the originally uploaded file for the workflow.

- To use the previous file as the input, enter `${previous.file}`. In this case, this workflow step uses the output file from the previous workflow step as input. This is the default value.
- To use the originally uploaded file location as input for this step, enter `${original.file}`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `^\$\{(\w+.\w+)\}`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)

- [AWS SDK for Ruby V3](#)

DescribedAccess

Describes the properties of the access that was specified.

Contents

ExternalId

A unique identifier that is required to identify specific groups within your directory. The users of the group that you associate have access to your Amazon S3 or Amazon EFS resources over the enabled protocols using AWS Transfer Family. If you know the group name, you can view the SID values by running the following command using Windows PowerShell.

```
Get-ADGroup -Filter {samAccountName -like "YourGroupName*"} -Properties * | Select SamAccountName, ObjectSid
```

In that command, replace *YourGroupName* with the name of your Active Directory group.

The regular expression used to validate this parameter is a string of characters consisting of uppercase and lowercase alphanumeric characters with no spaces. You can also include underscores or any of the following characters: =, ., @, /, -

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: S-1-[\d-]+

Required: No

HomeDirectory

The landing directory (folder) for a user when they log in to the server using the client.

A `HomeDirectory` example is `/bucket_name/home/mydirectory`.

Note

The `HomeDirectory` parameter is only used if `HomeDirectoryType` is set to `PATH`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: (| / . *)

Required: No

HomeDirectoryMappings

Logical directory mappings that specify what Amazon S3 or Amazon EFS paths and keys should be visible to your user and how you want to make them visible. You must specify the `Entry` and `Target` pair, where `Entry` shows how the path is made visible and `Target` is the actual Amazon S3 or Amazon EFS path. If you only specify a target, it is displayed as is. You also must ensure that your AWS Identity and Access Management (IAM) role provides access to paths in `Target`. This value can be set only when `HomeDirectoryType` is set to `LOGICAL`.

In most cases, you can use this value instead of the session policy to lock down the associated access to the designated home directory ("chroot"). To do this, you can set `Entry` to `'/'` and set `Target` to the `HomeDirectory` parameter value.

Type: Array of [HomeDirectoryMapEntry](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50000 items.

Required: No

HomeDirectoryType

The type of landing directory (folder) that you want your users' home directory to be when they log in to the server. If you set it to `PATH`, the user will see the absolute Amazon S3 bucket or Amazon EFS path as is in their file transfer protocol clients. If you set it to `LOGICAL`, you need to provide mappings in the `HomeDirectoryMappings` for how you want to make Amazon S3 or Amazon EFS paths visible to your users.

Note

If `HomeDirectoryType` is `LOGICAL`, you must provide mappings, using the `HomeDirectoryMappings` parameter. If, on the other hand, `HomeDirectoryType` is `PATH`, you provide an absolute path using the `HomeDirectory` parameter. You cannot have both `HomeDirectory` and `HomeDirectoryMappings` in your template.

Type: String

Valid Values: PATH | LOGICAL

Required: No

Policy

A session policy for your user so that you can use the same AWS Identity and Access Management (IAM) role across multiple users. This policy scopes down a user's access to portions of their Amazon S3 bucket. Variables that you can use inside this policy include `${Transfer:UserName}`, `${Transfer:HomeDirectory}`, and `${Transfer:HomeBucket}`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Required: No

PosixProfile

The full POSIX identity, including user ID (Uid), group ID (Gid), and any secondary groups IDs (SecondaryGids), that controls your users' access to your Amazon EFS file systems. The POSIX permissions that are set on files and directories in your file system determine the level of access your users get when transferring files into and out of your Amazon EFS file systems.

Type: [PosixProfile](#) object

Required: No

Role

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that controls your users' access to your Amazon S3 bucket or Amazon EFS file system. The policies attached to this role determine the level of access that you want to provide your users when transferring files into and out of your Amazon S3 bucket or Amazon EFS file system. The IAM role should also contain a trust relationship that allows the server to access your resources when servicing your users' transfer requests.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DescribedAgreement

Describes the properties of an agreement.

Contents

Arn

The unique Amazon Resource Name (ARN) for the agreement.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

AccessRole

Connectors are used to send files using either the AS2 or SFTP protocol. For the access role, provide the Amazon Resource Name (ARN) of the AWS Identity and Access Management role to use.

For AS2 connectors

With AS2, you can send files by calling `StartFileTransfer` and specifying the file paths in the request parameter, `SendFilePaths`. We use the file's parent directory (for example, for `--send-file-paths /bucket/dir/file.txt`, parent directory is `/bucket/dir/`) to temporarily store a processed AS2 message file, store the MDN when we receive them from the partner, and write a final JSON file containing relevant metadata of the transmission. So, the `AccessRole` needs to provide read and write access to the parent directory of the file location used in the `StartFileTransfer` request. Additionally, you need to provide read and write access to the parent directory of the files that you intend to send with `StartFileTransfer`.

If you are using Basic authentication for your AS2 connector, the access role requires the `secretsmanager:GetSecretValue` permission for the secret. If the secret is encrypted using a customer-managed key instead of the AWS managed key in Secrets Manager, then the role also needs the `kms:Decrypt` permission for that key.

For SFTP connectors

Make sure that the access role provides read and write access to the parent directory of the file location that's used in the `StartFileTransfer` request. Additionally, make sure that the role provides `secretsmanager:GetSecretValue` permission to AWS Secrets Manager.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

AgreementId

A unique identifier for the agreement. This identifier is returned when you create an agreement.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `a-([0-9a-f]{17})`

Required: No

BaseDirectory

The landing directory (folder) for files that are transferred by using the AS2 protocol.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `(|/.*)`

Required: No

Description

The name or short description that's used to identify the agreement.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[\p{Graph}]+`

Required: No

LocalProfileId

A unique identifier for the AS2 local profile.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Required: No

PartnerProfileId

A unique identifier for the partner profile used in the agreement.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Required: No

ServerId

A system-assigned unique identifier for a server instance. This identifier indicates the specific server that the agreement uses.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: No

Status

The current status of the agreement, either ACTIVE or INACTIVE.

Type: String

Valid Values: ACTIVE | INACTIVE

Required: No

Tags

Key-value pairs that can be used to group and search for agreements.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DescribedCertificate

Describes the properties of a certificate.

Contents

Arn

The unique Amazon Resource Name (ARN) for the certificate.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

ActiveDate

An optional date that specifies when the certificate becomes active.

Type: Timestamp

Required: No

Certificate

The file name for the certificate.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 16384.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF]*`

Required: No

CertificateChain

The list of certificates that make up the chain for the certificate.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2097152.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF]*`

Required: No

CertificateId

An array of identifiers for the imported certificates. You use this identifier for working with profiles and partner profiles.

Type: String

Length Constraints: Fixed length of 22.

Pattern: `cert-([0-9a-f]{17})`

Required: No

Description

The name or description that's used to identify the certificate.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[\p{Graph}]+`

Required: No

InactiveDate

An optional date that specifies when the certificate becomes inactive.

Type: Timestamp

Required: No

NotAfterDate

The final date that the certificate is valid.

Type: Timestamp

Required: No

NotBeforeDate

The earliest date that the certificate is valid.

Type: Timestamp

Required: No

Serial

The serial number for the certificate.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 48.

Pattern: `[\p{XDigit}{2}:?]*`

Required: No

Status

The certificate can be either ACTIVE, PENDING_ROTATION, or INACTIVE. PENDING_ROTATION means that this certificate will replace the current certificate when it expires.

Type: String

Valid Values: ACTIVE | PENDING_ROTATION | INACTIVE

Required: No

Tags

Key-value pairs that can be used to group and search for certificates.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

Type

If a private key has been specified for the certificate, its type is CERTIFICATE_WITH_PRIVATE_KEY. If there is no private key, the type is CERTIFICATE.

Type: String

Valid Values: CERTIFICATE | CERTIFICATE_WITH_PRIVATE_KEY

Required: No

Usage

Specifies how this certificate is used. It can be used in the following ways:

- SIGNING: For signing AS2 messages
- ENCRYPTION: For encrypting AS2 messages
- TLS: For securing AS2 communications sent over HTTPS

Type: String

Valid Values: SIGNING | ENCRYPTION

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DescribedConnector

Describes the parameters for the connector, as identified by the `ConnectorId`.

Contents

Arn

The unique Amazon Resource Name (ARN) for the connector.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

AccessRole

Connectors are used to send files using either the AS2 or SFTP protocol. For the access role, provide the Amazon Resource Name (ARN) of the AWS Identity and Access Management role to use.

For AS2 connectors

With AS2, you can send files by calling `StartFileTransfer` and specifying the file paths in the request parameter, `SendFilePaths`. We use the file's parent directory (for example, for `--send-file-paths /bucket/dir/file.txt`, parent directory is `/bucket/dir/`) to temporarily store a processed AS2 message file, store the MDN when we receive them from the partner, and write a final JSON file containing relevant metadata of the transmission. So, the `AccessRole` needs to provide read and write access to the parent directory of the file location used in the `StartFileTransfer` request. Additionally, you need to provide read and write access to the parent directory of the files that you intend to send with `StartFileTransfer`.

If you are using Basic authentication for your AS2 connector, the access role requires the `secretsmanager:GetSecretValue` permission for the secret. If the secret is encrypted using a customer-managed key instead of the AWS managed key in Secrets Manager, then the role also needs the `kms:Decrypt` permission for that key.

For SFTP connectors

Make sure that the access role provides read and write access to the parent directory of the file location that's used in the `StartFileTransfer` request. Additionally, make sure that the role provides `secretsmanager:GetSecretValue` permission to AWS Secrets Manager.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

As2Config

A structure that contains the parameters for an AS2 connector object.

Type: [As2ConnectorConfig](#) object

Required: No

ConnectorId

The unique identifier for the connector.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `c-([0-9a-f]{17})`

Required: No

LoggingRole

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that allows a connector to turn on CloudWatch logging for Amazon S3 events. When set, you can view connector activity in your CloudWatch logs.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

SecurityPolicyName

The text name of the security policy for the specified connector.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 100.

Pattern: TransferSFTPConnectorSecurityPolicy-[A-Za-z0-9-]+

Required: No

ServiceManagedEgressIpAddresses

The list of egress IP addresses of this connector. These IP addresses are assigned automatically when you create the connector.

Type: Array of strings

Pattern: \d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}

Required: No

SftpConfig

A structure that contains the parameters for an SFTP connector object.

Type: [SftpConnectorConfig](#) object

Required: No

Tags

Key-value pairs that can be used to group and search for connectors.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

Url

The URL of the partner's AS2 or SFTP endpoint.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DescribedExecution

The details for an execution object.

Contents

ExecutionId

A unique identifier for the execution of a workflow.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}`

Required: No

ExecutionRole

The IAM role associated with the execution.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

InitialFileLocation

A structure that describes the Amazon S3 or EFS file location. This is the file location when the execution begins: if the file is being copied, this is the initial (as opposed to destination) file location.

Type: [FileLocation](#) object

Required: No

LoggingConfiguration

The IAM logging role associated with the execution.

Type: [LoggingConfiguration](#) object

Required: No

PosixProfile

The full POSIX identity, including user ID (Uid), group ID (Gid), and any secondary groups IDs (SecondaryGids), that controls your users' access to your Amazon EFS file systems. The POSIX permissions that are set on files and directories in your file system determine the level of access your users get when transferring files into and out of your Amazon EFS file systems.

Type: [PosixProfile](#) object

Required: No

Results

A structure that describes the execution results. This includes a list of the steps along with the details of each step, error type and message (if any), and the OnExceptionSteps structure.

Type: [ExecutionResults](#) object

Required: No

ServiceMetadata

A container object for the session details that are associated with a workflow.

Type: [ServiceMetadata](#) object

Required: No

Status

The status is one of the execution. Can be in progress, completed, exception encountered, or handling the exception.

Type: String

Valid Values: IN_PROGRESS | COMPLETED | EXCEPTION | HANDLING_EXCEPTION

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DescribedHostKey

The details for a server host key.

Contents

Arn

The unique Amazon Resource Name (ARN) for the host key.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

DateImported

The date on which the host key was added to the server.

Type: Timestamp

Required: No

Description

The text description for this host key.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Pattern: `[\p{Print}]*`

Required: No

HostKeyFingerprint

The public key fingerprint, which is a short sequence of bytes used to identify the longer public key.

Type: String

Required: No

HostKeyId

A unique identifier for the host key.

Type: String

Length Constraints: Fixed length of 25.

Pattern: `hostkey-[0-9a-f]{17}`

Required: No

Tags

Key-value pairs that can be used to group and search for host keys.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

Type

The encryption algorithm that is used for the host key. The Type parameter is specified by using one of the following values:

- `ssh-rsa`
- `ssh-ed25519`
- `ecdsa-sha2-nistp256`
- `ecdsa-sha2-nistp384`
- `ecdsa-sha2-nistp521`

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DescribedProfile

The details for a local or partner AS2 profile.

Contents

Arn

The unique Amazon Resource Name (ARN) for the profile.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

As2Id

The As2Id is the *AS2-name*, as defined in the [RFC 4130](#). For inbound transfers, this is the AS2-From header for the AS2 messages sent from the partner. For outbound connectors, this is the AS2-To header for the AS2 messages sent to the partner using the `StartFileTransfer` API operation. This ID cannot include spaces.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[\p{Print}\s]*`

Required: No

CertificateIds

An array of identifiers for the imported certificates. You use this identifier for working with profiles and partner profiles.

Type: Array of strings

Length Constraints: Fixed length of 22.

Pattern: `cert-([0-9a-f]{17})`

Required: No

ProfileId

A unique identifier for the local or partner AS2 profile.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Required: No

ProfileType

Indicates whether to list only LOCAL type profiles or only PARTNER type profiles. If not supplied in the request, the command lists all types of profiles.

Type: String

Valid Values: LOCAL | PARTNER

Required: No

Tags

Key-value pairs that can be used to group and search for profiles.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DescribedSecurityPolicy

Describes the properties of a security policy that you specify. For more information about security policies, see [Working with security policies for servers](#) or [Working with security policies for SFTP connectors](#).

Contents

SecurityPolicyName

The text name of the specified security policy.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 100.

Pattern: `Transfer[A-Za-z0-9]*SecurityPolicy-[A-Za-z0-9-]+`

Required: Yes

Fips

Specifies whether this policy enables Federal Information Processing Standards (FIPS). This parameter applies to both server and connector security policies.

Type: Boolean

Required: No

Protocols

Lists the file transfer protocols that the security policy applies to.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Valid Values: SFTP | FTPS

Required: No

SshCiphers

Lists the enabled Secure Shell (SSH) cipher encryption algorithms in the security policy that is attached to the server or connector. This parameter applies to both server and connector security policies.


Type: Array of strings

Length Constraints: Minimum length of 0. Maximum length of 50.

Required: No

SshHostKeyAlgorithms

Lists the host key algorithms for the security policy.

 **Note**

This parameter only applies to security policies for connectors.

Type: Array of strings

Length Constraints: Minimum length of 0. Maximum length of 50.

Required: No

SshKexs

Lists the enabled SSH key exchange (KEX) encryption algorithms in the security policy that is attached to the server or connector. This parameter applies to both server and connector security policies.

Type: Array of strings

Length Constraints: Minimum length of 0. Maximum length of 50.

Required: No

SshMacs

Lists the enabled SSH message authentication code (MAC) encryption algorithms in the security policy that is attached to the server or connector. This parameter applies to both server and connector security policies.

Type: Array of strings

Length Constraints: Minimum length of 0. Maximum length of 50.

Required: No

TlsCiphers

Lists the enabled Transport Layer Security (TLS) cipher encryption algorithms in the security policy that is attached to the server.

Note

This parameter only applies to security policies for servers.

Type: Array of strings

Length Constraints: Minimum length of 0. Maximum length of 50.

Required: No

Type

The resource type to which the security policy applies, either server or connector.

Type: String

Valid Values: SERVER | CONNECTOR

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DescribedServer

Describes the properties of a file transfer protocol-enabled server that was specified.

Contents

Arn

Specifies the unique Amazon Resource Name (ARN) of the server.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

As2ServiceManagedEgressIpAddresses

The list of egress IP addresses of this server. These IP addresses are only relevant for servers that use the AS2 protocol. They are used for sending asynchronous MDNs.

These IP addresses are assigned automatically when you create an AS2 server. Additionally, if you update an existing server and add the AS2 protocol, static IP addresses are assigned as well.

Type: Array of strings

Pattern: `\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}`

Required: No

Certificate

Specifies the ARN of the AWS Certificate Manager (ACM) certificate. Required when `Protocols` is set to `FTPS`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1600.

Required: No

Domain

Specifies the domain of the storage system that is used for file transfers. There are two domains available: Amazon Simple Storage Service (Amazon S3) and Amazon Elastic File System (Amazon EFS). The default value is S3.

Type: String

Valid Values: S3 | EFS

Required: No

EndpointDetails

The virtual private cloud (VPC) endpoint settings that are configured for your server. When you host your endpoint within your VPC, you can make your endpoint accessible only to resources within your VPC, or you can attach Elastic IP addresses and make your endpoint accessible to clients over the internet. Your VPC's default security groups are automatically assigned to your endpoint.

Type: [EndpointDetails](#) object

Required: No

EndpointType

Defines the type of endpoint that your server is connected to. If your server is connected to a VPC endpoint, your server isn't accessible over the public internet.

Type: String

Valid Values: PUBLIC | VPC | VPC_ENDPOINT

Required: No

HostKeyFingerprint

Specifies the Base64-encoded SHA256 fingerprint of the server's host key. This value is equivalent to the output of the `ssh-keygen -l -f my-new-server-key` command.

Type: String

Required: No

IdentityProviderDetails

Specifies information to call a customer-supplied authentication API. This field is not populated when the `IdentityProviderType` of a server is `AWS_DIRECTORY_SERVICE` or `SERVICE_MANAGED`.

Type: [IdentityProviderDetails](#) object

Required: No

IdentityProviderType

The mode of authentication for a server. The default value is `SERVICE_MANAGED`, which allows you to store and access user credentials within the AWS Transfer Family service.

Use `AWS_DIRECTORY_SERVICE` to provide access to Active Directory groups in AWS Directory Service for Microsoft Active Directory or Microsoft Active Directory in your on-premises environment or in AWS using AD Connector. This option also requires you to provide a Directory ID by using the `IdentityProviderDetails` parameter.

Use the `API_GATEWAY` value to integrate with an identity provider of your choosing. The `API_GATEWAY` setting requires you to provide an Amazon API Gateway endpoint URL to call for authentication by using the `IdentityProviderDetails` parameter.

Use the `AWS_LAMBDA` value to directly use an AWS Lambda function as your identity provider. If you choose this value, you must specify the ARN for the Lambda function in the `Function` parameter for the `IdentityProviderDetails` data type.

Type: String

Valid Values: `SERVICE_MANAGED` | `API_GATEWAY` | `AWS_DIRECTORY_SERVICE` | `AWS_LAMBDA`

Required: No

LoggingRole

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that allows a server to turn on Amazon CloudWatch logging for Amazon S3 or Amazon EFS events. When set, you can view user activity in your CloudWatch logs.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Pattern: (`|arn:.*role/\S+`)

Required: No

PostAuthenticationLoginBanner

Specifies a string to display when users connect to a server. This string is displayed after the user authenticates.

Note

The SFTP protocol does not support post-authentication display banners.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Pattern: [`\x09-\x0D\x20-\x7E`]*

Required: No

PreAuthenticationLoginBanner

Specifies a string to display when users connect to a server. This string is displayed before the user authenticates. For example, the following banner displays details about using the system:

```
This system is for the use of authorized users only. Individuals using
this computer system without authority, or in excess of their authority,
are subject to having all of their activities on this system monitored
and recorded by system personnel.
```

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Pattern: [`\x09-\x0D\x20-\x7E`]*

Required: No

ProtocolDetails

The protocol settings that are configured for your server.

- To indicate passive mode (for FTP and FTPS protocols), use the `PassiveIp` parameter. Enter a single dotted-quad IPv4 address, such as the external IP address of a firewall, router, or load balancer.
- To ignore the error that is generated when the client attempts to use the `SETSTAT` command on a file that you are uploading to an Amazon S3 bucket, use the `SetStatOption` parameter. To have the AWS Transfer Family server ignore the `SETSTAT` command and upload files without needing to make any changes to your SFTP client, set the value to `ENABLE_NO_OP`. If you set the `SetStatOption` parameter to `ENABLE_NO_OP`, Transfer Family generates a log entry to Amazon CloudWatch Logs, so that you can determine when the client is making a `SETSTAT` call.
- To determine whether your AWS Transfer Family server resumes recent, negotiated sessions through a unique session ID, use the `TlsSessionResumptionMode` parameter.
- `As2Transports` indicates the transport method for the AS2 messages. Currently, only HTTP is supported.

Type: [ProtocolDetails](#) object

Required: No

Protocols

Specifies the file transfer protocol or protocols over which your file transfer protocol client can connect to your server's endpoint. The available protocols are:

- SFTP (Secure Shell (SSH) File Transfer Protocol): File transfer over SSH
- FTPS (File Transfer Protocol Secure): File transfer with TLS encryption
- FTP (File Transfer Protocol): Unencrypted file transfer
- AS2 (Applicability Statement 2): used for transporting structured business-to-business data

Note

- If you select FTPS, you must choose a certificate stored in AWS Certificate Manager (ACM) which is used to identify your server when clients connect to it over FTPS.
- If `Protocol` includes either FTP or FTPS, then the `EndpointType` must be `VPC` and the `IdentityProviderType` must be either `AWS_DIRECTORY_SERVICE`, `AWS_LAMBDA`, or `API_GATEWAY`.
- If `Protocol` includes FTP, then `AddressAllocationIds` cannot be associated.

- If Protocol is set only to SFTP, the EndpointType can be set to PUBLIC and the IdentityProviderType can be set any of the supported identity types: SERVICE_MANAGED, AWS_DIRECTORY_SERVICE, AWS_LAMBDA, or API_GATEWAY.
- If Protocol includes AS2, then the EndpointType must be VPC, and domain must be Amazon S3.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 4 items.

Valid Values: SFTP | FTP | FTPS | AS2

Required: No

S3StorageOptions

Specifies whether or not performance for your Amazon S3 directories is optimized. This is disabled by default.

By default, home directory mappings have a TYPE of DIRECTORY. If you enable this option, you would then need to explicitly set the HomeDirectoryMapEntry Type to FILE if you want a mapping to have a file target.

Type: [S3StorageOptions](#) object

Required: No

SecurityPolicyName

Specifies the name of the security policy for the server.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 100.

Pattern: Transfer[A-Za-z0-9]*SecurityPolicy-[A-Za-z0-9-]+

Required: No

ServerId

Specifies the unique system-assigned identifier for a server that you instantiate.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Required: No

State

The condition of the server that was described. A value of `ONLINE` indicates that the server can accept jobs and transfer files. A `State` value of `OFFLINE` means that the server cannot perform file transfer operations.

The states of `STARTING` and `STOPPING` indicate that the server is in an intermediate state, either not fully able to respond, or not fully offline. The values of `START_FAILED` or `STOP_FAILED` can indicate an error condition.

Type: String

Valid Values: `OFFLINE` | `ONLINE` | `STARTING` | `STOPPING` | `START_FAILED` | `STOP_FAILED`

Required: No

StructuredLogDestinations

Specifies the log groups to which your server logs are sent.

To specify a log group, you must provide the ARN for an existing log group. In this case, the format of the log group is as follows:

```
arn:aws:logs:region-name:amazon-account-id:log-group:log-group-name:*
```

For example, `arn:aws:logs:us-east-1:111122223333:log-group:mytestgroup:*`

If you have previously specified a log group for a server, you can clear it, and in effect turn off structured logging, by providing an empty value for this parameter in an `update-server` call. For example:

```
update-server --server-id s-1234567890abcdef0 --structured-log-destinations
```

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 1 item.

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: No

Tags

Specifies the key-value pairs that you can use to search for and group servers that were assigned to the server that was described.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

UserCount

Specifies the number of users that are assigned to a server you specified with the `ServerId`.

Type: Integer

Required: No

WorkflowDetails

Specifies the workflow ID for the workflow to assign and the execution role that's used for executing the workflow.

In addition to a workflow to execute when a file is uploaded completely, `WorkflowDetails` can also contain a workflow ID (and execution role) for a workflow to execute on partial upload. A partial upload occurs when the server session disconnects while the file is still being uploaded.

Type: [WorkflowDetails](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DescribedUser

Describes the properties of a user that was specified.

Contents

Arn

Specifies the unique Amazon Resource Name (ARN) for the user that was requested to be described.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

HomeDirectory

The landing directory (folder) for a user when they log in to the server using the client.

A HomeDirectory example is `/bucket_name/home/mydirectory`.

Note

The HomeDirectory parameter is only used if HomeDirectoryType is set to PATH.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `(|/.*)`

Required: No

HomeDirectoryMappings

Logical directory mappings that specify what Amazon S3 or Amazon EFS paths and keys should be visible to your user and how you want to make them visible. You must specify the Entry and Target pair, where Entry shows how the path is made visible and Target is the actual

Amazon S3 or Amazon EFS path. If you only specify a target, it is displayed as is. You also must ensure that your AWS Identity and Access Management (IAM) role provides access to paths in Target. This value can be set only when `HomeDirectoryType` is set to `LOGICAL`.

In most cases, you can use this value instead of the session policy to lock your user down to the designated home directory ("chroot"). To do this, you can set `Entry` to `'/'` and set `Target` to the `HomeDirectory` parameter value.

Type: Array of [HomeDirectoryMapEntry](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50000 items.

Required: No

HomeDirectoryType

The type of landing directory (folder) that you want your users' home directory to be when they log in to the server. If you set it to `PATH`, the user will see the absolute Amazon S3 bucket or Amazon EFS path as is in their file transfer protocol clients. If you set it to `LOGICAL`, you need to provide mappings in the `HomeDirectoryMappings` for how you want to make Amazon S3 or Amazon EFS paths visible to your users.

Note

If `HomeDirectoryType` is `LOGICAL`, you must provide mappings, using the `HomeDirectoryMappings` parameter. If, on the other hand, `HomeDirectoryType` is `PATH`, you provide an absolute path using the `HomeDirectory` parameter. You cannot have both `HomeDirectory` and `HomeDirectoryMappings` in your template.

Type: String

Valid Values: `PATH` | `LOGICAL`

Required: No

Policy

A session policy for your user so that you can use the same AWS Identity and Access Management (IAM) role across multiple users. This policy scopes down a user's access to portions of their Amazon S3 bucket. Variables that you can use inside this policy include `${Transfer:UserName}`, `${Transfer:HomeDirectory}`, and `${Transfer:HomeBucket}`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Required: No

PosixProfile

Specifies the full POSIX identity, including user ID (Uid), group ID (Gid), and any secondary groups IDs (SecondaryGids), that controls your users' access to your Amazon Elastic File System (Amazon EFS) file systems. The POSIX permissions that are set on files and directories in your file system determine the level of access your users get when transferring files into and out of your Amazon EFS file systems.

Type: [PosixProfile](#) object

Required: No

Role

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that controls your users' access to your Amazon S3 bucket or Amazon EFS file system. The policies attached to this role determine the level of access that you want to provide your users when transferring files into and out of your Amazon S3 bucket or Amazon EFS file system. The IAM role should also contain a trust relationship that allows the server to access your resources when servicing your users' transfer requests.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

SshPublicKeys

Specifies the public key portion of the Secure Shell (SSH) keys stored for the described user.

Type: Array of [SshPublicKey](#) objects

Array Members: Minimum number of 0 items. Maximum number of 5 items.

Required: No

Tags

Specifies the key-value pairs for the user requested. Tag can be used to search for and group users for a variety of purposes.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

UserName

Specifies the name of the user that was requested to be described. User names are used for authentication purposes. This is the string that will be used by your user when they log in to your server.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\w][\w@.-]{2,99}`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DescribedWorkflow

Describes the properties of the specified workflow

Contents

Arn

Specifies the unique Amazon Resource Name (ARN) for the workflow.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

Description

Specifies the text description for the workflow.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `[\w-]*`

Required: No

OnExceptionSteps

Specifies the steps (actions) to take if errors are encountered during execution of the workflow.

Type: Array of [WorkflowStep](#) objects

Array Members: Minimum number of 0 items. Maximum number of 8 items.

Required: No

Steps

Specifies the details for the steps that are in the specified workflow.

Type: Array of [WorkflowStep](#) objects

Array Members: Minimum number of 0 items. Maximum number of 8 items.

Required: No

Tags

Key-value pairs that can be used to group and search for workflows. Tags are metadata attached to workflows for any purpose.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `w-([a-z0-9]{17})`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

EfsFileLocation

Specifies the details for the file location for the file that's being used in the workflow. Only applicable if you are using Amazon Elastic File Systems (Amazon EFS) for storage.

Contents

FileSystemId

The identifier of the file system, assigned by Amazon EFS.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 128.

Pattern: `(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:(access-point/fsap|file-system/fs)-[0-9a-f]{8,40}|fs(ap)?-[0-9a-f]{8,40})`

Required: No

Path

The pathname for the folder being used by a workflow.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 65536.

Pattern: `[^\x00]+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

EndpointDetails

The virtual private cloud (VPC) endpoint settings that are configured for your file transfer protocol-enabled server. With a VPC endpoint, you can restrict access to your server and resources only within your VPC. To control incoming internet traffic, invoke the `UpdateServer` API and attach an Elastic IP address to your server's endpoint.

Note

After May 19, 2021, you won't be able to create a server using `EndpointType=VPC_ENDPOINT` in your AWS account if your account hasn't already done so before May 19, 2021. If you have already created servers with `EndpointType=VPC_ENDPOINT` in your AWS account on or before May 19, 2021, you will not be affected. After this date, use `EndpointType=VPC`. For more information, see [Discontinuing the use of VPC_ENDPOINT](#).

Contents

AddressAllocationIds

A list of address allocation IDs that are required to attach an Elastic IP address to your server's endpoint.

An address allocation ID corresponds to the allocation ID of an Elastic IP address. This value can be retrieved from the `allocationId` field from the Amazon EC2 [Address](#) data type. One way to retrieve this value is by calling the EC2 [DescribeAddresses](#) API.

This parameter is optional. Set this parameter if you want to make your VPC endpoint public-facing. For details, see [Create an internet-facing endpoint for your server](#).

Note

This property can only be set as follows:

- `EndpointType` must be set to `VPC`
- The Transfer Family server must be offline.
- You cannot set this parameter for Transfer Family servers that use the FTP protocol.

- The server must already have SubnetIds populated (SubnetIds and AddressAllocationIds cannot be updated simultaneously).
- AddressAllocationIds can't contain duplicates, and must be equal in length to SubnetIds. For example, if you have three subnet IDs, you must also specify three address allocation IDs.
- Call the UpdateServer API to set or change this parameter.

Type: Array of strings

Required: No

SecurityGroupIds

A list of security groups IDs that are available to attach to your server's endpoint.

Note

This property can only be set when EndpointType is set to VPC.

You can edit the SecurityGroupIds property in the [UpdateServer](#) API only if you are changing the EndpointType from PUBLIC or VPC_ENDPOINT to VPC. To change security groups associated with your server's VPC endpoint after creation, use the Amazon EC2 [ModifyVpcEndpoint](#) API.

Type: Array of strings

Length Constraints: Minimum length of 11. Maximum length of 20.

Pattern: sg-[0-9a-f]{8,17}

Required: No

SubnetIds

A list of subnet IDs that are required to host your server endpoint in your VPC.

Note

This property can only be set when EndpointType is set to VPC.

Type: Array of strings

Required: No

VpcEndpointId

The identifier of the VPC endpoint.

Note

This property can only be set when `EndpointType` is set to `VPC_ENDPOINT`. For more information, see [Discontinuing the use of VPC_ENDPOINT](#).

Type: String

Length Constraints: Fixed length of 22.

Pattern: `vpce-[0-9a-f]{17}`

Required: No

VpcId

The VPC identifier of the VPC in which a server's endpoint will be hosted.

Note

This property can only be set when `EndpointType` is set to `VPC`.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)

- [AWS SDK for Ruby V3](#)

ExecutionError

Specifies the error message and type, for an error that occurs during the execution of the workflow.

Contents

Message

Specifies the descriptive message that corresponds to the `ErrorType`.

Type: String

Required: Yes

Type

Specifies the error type.

- `ALREADY_EXISTS`: occurs for a copy step, if the overwrite option is not selected and a file with the same name already exists in the target location.
- `BAD_REQUEST`: a general bad request: for example, a step that attempts to tag an EFS file returns `BAD_REQUEST`, as only S3 files can be tagged.
- `CUSTOM_STEP_FAILED`: occurs when the custom step provided a callback that indicates failure.
- `INTERNAL_SERVER_ERROR`: a catch-all error that can occur for a variety of reasons.
- `NOT_FOUND`: occurs when a requested entity, for example a source file for a copy step, does not exist.
- `PERMISSION_DENIED`: occurs if your policy does not contain the correct permissions to complete one or more of the steps in the workflow.
- `TIMEOUT`: occurs when the execution times out.

Note

You can set the `TimeoutSeconds` for a custom step, anywhere from 1 second to 1800 seconds (30 minutes).

- `THROTTLED`: occurs if you exceed the new execution refill rate of one workflow per second.

Type: String

Valid Values: PERMISSION_DENIED | CUSTOM_STEP_FAILED | THROTTLED
| ALREADY_EXISTS | NOT_FOUND | BAD_REQUEST | TIMEOUT |
INTERNAL_SERVER_ERROR

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ExecutionResults

Specifies the steps in the workflow, as well as the steps to execute in case of any errors during workflow execution.

Contents

OnExceptionSteps

Specifies the steps (actions) to take if errors are encountered during execution of the workflow.

Type: Array of [ExecutionStepResult](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

Steps

Specifies the details for the steps that are in the specified workflow.

Type: Array of [ExecutionStepResult](#) objects

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ExecutionStepResult

Specifies the following details for the step: error (if any), outputs (if any), and the step type.

Contents

Error

Specifies the details for an error, if it occurred during execution of the specified workflow step.

Type: [ExecutionError](#) object

Required: No

Outputs

The values for the key/value pair applied as a tag to the file. Only applicable if the step type is TAG.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 65536.

Required: No

StepType

One of the available step types.

- **COPY** - Copy the file to another location.
- **CUSTOM** - Perform a custom step with an AWS Lambda function target.
- **DECRYPT** - Decrypt a file that was encrypted before it was uploaded.
- **DELETE** - Delete the file.
- **TAG** - Add a tag to the file.

Type: String

Valid Values: COPY | CUSTOM | TAG | DELETE | DECRYPT

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

FileLocation

Specifies the Amazon S3 or EFS file details to be used in the step.

Contents

EfsFileLocation

Specifies the Amazon EFS identifier and the path for the file being used.

Type: [EfsFileLocation](#) object

Required: No

S3FileLocation

Specifies the S3 details for the file being used, such as bucket, ETag, and so forth.

Type: [S3FileLocation](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

HomeDirectoryMapEntry

Represents an object that contains entries and targets for HomeDirectoryMappings.

The following is an Entry and Target pair example for chroot.

```
[ { "Entry": "/", "Target": "/bucket_name/home/mydirectory" } ]
```

Contents

Entry

Represents an entry for HomeDirectoryMappings.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: /. *

Required: Yes

Target

Represents the map target that is used in a HomeDirectoryMapEntry.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: /. *

Required: Yes

Type

Specifies the type of mapping. Set the type to FILE if you want the mapping to point to a file, or DIRECTORY for the directory to point to a directory.

Note

By default, home directory mappings have a Type of DIRECTORY when you create a Transfer Family server. You would need to explicitly set Type to FILE if you want a mapping to have a file target.

Type: String

Valid Values: FILE | DIRECTORY

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

IdentityProviderDetails

Returns information related to the type of user authentication that is in use for a file transfer protocol-enabled server's users. A server can have only one method of authentication.

Contents

DirectoryId

The identifier of the AWS Directory Service directory that you want to use as your identity provider.

Type: String

Length Constraints: Fixed length of 12.

Pattern: `d-[0-9a-f]{10}`

Required: No

Function

The ARN for a Lambda function to use for the Identity provider.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 170.

Pattern: `arn:[a-z-]+:lambda:.*`

Required: No

InvocationRole

This parameter is only applicable if your `IdentityProviderType` is `API_GATEWAY`. Provides the type of `InvocationRole` used to authenticate the user account.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

SftpAuthenticationMethods

For SFTP-enabled servers, and for custom identity providers *only*, you can specify whether to authenticate using a password, SSH key pair, or both.

- `PASSWORD` - users must provide their password to connect.
- `PUBLIC_KEY` - users must provide their private key to connect.
- `PUBLIC_KEY_OR_PASSWORD` - users can authenticate with either their password or their key. This is the default value.
- `PUBLIC_KEY_AND_PASSWORD` - users must provide both their private key and their password to connect. The server checks the key first, and then if the key is valid, the system prompts for a password. If the private key provided does not match the public key that is stored, authentication fails.

Type: String

Valid Values: `PASSWORD` | `PUBLIC_KEY` | `PUBLIC_KEY_OR_PASSWORD` | `PUBLIC_KEY_AND_PASSWORD`

Required: No

Url

Provides the location of the service endpoint used to authenticate users.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

InputFileLocation

Specifies the location for the file that's being processed.

Contents

EfsFileLocation

Specifies the details for the Amazon Elastic File System (Amazon EFS) file that's being decrypted.

Type: [EfsFileLocation](#) object

Required: No

S3FileLocation

Specifies the details for the Amazon S3 file that's being copied or decrypted.

Type: [S3InputFileLocation](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListedAccess

Lists the properties for one or more specified associated accesses.

Contents

ExternalId

A unique identifier that is required to identify specific groups within your directory. The users of the group that you associate have access to your Amazon S3 or Amazon EFS resources over the enabled protocols using AWS Transfer Family. If you know the group name, you can view the SID values by running the following command using Windows PowerShell.

```
Get-ADGroup -Filter {samAccountName -like "YourGroupName*"} -Properties * | Select SamAccountName, ObjectSid
```

In that command, replace *YourGroupName* with the name of your Active Directory group.

The regular expression used to validate this parameter is a string of characters consisting of uppercase and lowercase alphanumeric characters with no spaces. You can also include underscores or any of the following characters: =, ., @, /, -

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: S-1-[\d-]+

Required: No

HomeDirectory

The landing directory (folder) for a user when they log in to the server using the client.

A `HomeDirectory` example is `/bucket_name/home/mydirectory`.

Note

The `HomeDirectory` parameter is only used if `HomeDirectoryType` is set to `PATH`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: (| / . *)

Required: No

HomeDirectoryType

The type of landing directory (folder) that you want your users' home directory to be when they log in to the server. If you set it to `PATH`, the user will see the absolute Amazon S3 bucket or Amazon EFS path as is in their file transfer protocol clients. If you set it to `LOGICAL`, you need to provide mappings in the `HomeDirectoryMappings` for how you want to make Amazon S3 or Amazon EFS paths visible to your users.

Note

If `HomeDirectoryType` is `LOGICAL`, you must provide mappings, using the `HomeDirectoryMappings` parameter. If, on the other hand, `HomeDirectoryType` is `PATH`, you provide an absolute path using the `HomeDirectory` parameter. You cannot have both `HomeDirectory` and `HomeDirectoryMappings` in your template.

Type: String

Valid Values: `PATH` | `LOGICAL`

Required: No

Role

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that controls your users' access to your Amazon S3 bucket or Amazon EFS file system. The policies attached to this role determine the level of access that you want to provide your users when transferring files into and out of your Amazon S3 bucket or Amazon EFS file system. The IAM role should also contain a trust relationship that allows the server to access your resources when servicing your users' transfer requests.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListedAgreement

Describes the properties of an agreement.

Contents

AgreementId

A unique identifier for the agreement. This identifier is returned when you create an agreement.

Type: String

Length Constraints: Fixed length of 19.

Pattern: a-([0-9a-f]{17})

Required: No

Arn

The Amazon Resource Name (ARN) of the specified agreement.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: arn:\S+

Required: No

Description

The current description for the agreement. You can change it by calling the `UpdateAgreement` operation and providing a new description.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: [\p{Graph}]+

Required: No

LocalProfileId

A unique identifier for the AS2 local profile.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Required: No

PartnerProfileId

A unique identifier for the partner profile.

Type: String

Length Constraints: Fixed length of 19.

Pattern: p-([0-9a-f]{17})

Required: No

ServerId

The unique identifier for the agreement.

Type: String

Length Constraints: Fixed length of 19.

Pattern: s-([0-9a-f]{17})

Required: No

Status

The agreement can be either ACTIVE or INACTIVE.

Type: String

Valid Values: ACTIVE | INACTIVE

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListedCertificate

Describes the properties of a certificate.

Contents

ActiveDate

An optional date that specifies when the certificate becomes active.

Type: Timestamp

Required: No

Arn

The Amazon Resource Name (ARN) of the specified certificate.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: No

CertificateId

An array of identifiers for the imported certificates. You use this identifier for working with profiles and partner profiles.

Type: String

Length Constraints: Fixed length of 22.

Pattern: `cert-([0-9a-f]{17})`

Required: No

Description

The name or short description that's used to identify the certificate.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[\p{Graph}]+`

Required: No

InactiveDate

An optional date that specifies when the certificate becomes inactive.

Type: Timestamp

Required: No

Status

The certificate can be either `ACTIVE`, `PENDING_ROTATION`, or `INACTIVE`. `PENDING_ROTATION` means that this certificate will replace the current certificate when it expires.

Type: String

Valid Values: `ACTIVE` | `PENDING_ROTATION` | `INACTIVE`

Required: No

Type

The type for the certificate. If a private key has been specified for the certificate, its type is `CERTIFICATE_WITH_PRIVATE_KEY`. If there is no private key, the type is `CERTIFICATE`.

Type: String

Valid Values: `CERTIFICATE` | `CERTIFICATE_WITH_PRIVATE_KEY`

Required: No

Usage

Specifies how this certificate is used. It can be used in the following ways:

- `SIGNING`: For signing AS2 messages
- `ENCRYPTION`: For encrypting AS2 messages
- `TLS`: For securing AS2 communications sent over HTTPS

Type: String

Valid Values: SIGNING | ENCRYPTION

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListedConnector

Returns details of the connector that is specified.

Contents

Arn

The Amazon Resource Name (ARN) of the specified connector.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: No

ConnectorId

The unique identifier for the connector.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `c-([0-9a-f]{17})`

Required: No

Url

The URL of the partner's AS2 or SFTP endpoint.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListedExecution

Returns properties of the execution that is specified.

Contents

ExecutionId

A unique identifier for the execution of a workflow.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}`

Required: No

InitialFileLocation

A structure that describes the Amazon S3 or EFS file location. This is the file location when the execution begins: if the file is being copied, this is the initial (as opposed to destination) file location.

Type: [FileLocation](#) object

Required: No

ServiceMetadata

A container object for the session details that are associated with a workflow.

Type: [ServiceMetadata](#) object

Required: No

Status

The status is one of the execution. Can be in progress, completed, exception encountered, or handling the exception.

Type: String

Valid Values: IN_PROGRESS | COMPLETED | EXCEPTION | HANDLING_EXCEPTION

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListedHostKey

Returns properties of the host key that's specified.

Contents

Arn

The unique Amazon Resource Name (ARN) of the host key.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

DateImported

The date on which the host key was added to the server.

Type: Timestamp

Required: No

Description

The current description for the host key. You can change it by calling the `UpdateHostKey` operation and providing a new description.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Pattern: `[\p{Print}]*`

Required: No

Fingerprint

The public key fingerprint, which is a short sequence of bytes used to identify the longer public key.

Type: String

Required: No

HostKeyId

A unique identifier for the host key.

Type: String

Length Constraints: Fixed length of 25.

Pattern: `hostkey-[0-9a-f]{17}`

Required: No

Type

The encryption algorithm that is used for the host key. The Type parameter is specified by using one of the following values:

- `ssh-rsa`
- `ssh-ed25519`
- `ecdsa-sha2-nistp256`
- `ecdsa-sha2-nistp384`
- `ecdsa-sha2-nistp521`

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListedProfile

Returns the properties of the profile that was specified.

Contents

Arn

The Amazon Resource Name (ARN) of the specified profile.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: No

As2Id

The As2Id is the *AS2-name*, as defined in the [RFC 4130](#). For inbound transfers, this is the AS2-From header for the AS2 messages sent from the partner. For outbound connectors, this is the AS2-To header for the AS2 messages sent to the partner using the `StartFileTransfer` API operation. This ID cannot include spaces.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[\p{Print}\s]*`

Required: No

ProfileId

A unique identifier for the local or partner AS2 profile.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `p-([0-9a-f]{17})`

Required: No

ProfileType

Indicates whether to list only LOCAL type profiles or only PARTNER type profiles. If not supplied in the request, the command lists all types of profiles.

Type: String

Valid Values: LOCAL | PARTNER

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListedServer

Returns properties of a file transfer protocol-enabled server that was specified.

Contents

Arn

Specifies the unique Amazon Resource Name (ARN) for a server to be listed.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

Domain

Specifies the domain of the storage system that is used for file transfers. There are two domains available: Amazon Simple Storage Service (Amazon S3) and Amazon Elastic File System (Amazon EFS). The default value is S3.

Type: String

Valid Values: S3 | EFS

Required: No

EndpointType

Specifies the type of VPC endpoint that your server is connected to. If your server is connected to a VPC endpoint, your server isn't accessible over the public internet.

Type: String

Valid Values: PUBLIC | VPC | VPC_ENDPOINT

Required: No

IdentityProviderType

The mode of authentication for a server. The default value is SERVICE_MANAGED, which allows you to store and access user credentials within the AWS Transfer Family service.

Use `AWS_DIRECTORY_SERVICE` to provide access to Active Directory groups in AWS Directory Service for Microsoft Active Directory or Microsoft Active Directory in your on-premises environment or in AWS using AD Connector. This option also requires you to provide a Directory ID by using the `IdentityProviderDetails` parameter.

Use the `API_GATEWAY` value to integrate with an identity provider of your choosing. The `API_GATEWAY` setting requires you to provide an Amazon API Gateway endpoint URL to call for authentication by using the `IdentityProviderDetails` parameter.

Use the `AWS_LAMBDA` value to directly use an AWS Lambda function as your identity provider. If you choose this value, you must specify the ARN for the Lambda function in the `Function` parameter for the `IdentityProviderDetails` data type.

Type: String

Valid Values: `SERVICE_MANAGED` | `API_GATEWAY` | `AWS_DIRECTORY_SERVICE` | `AWS_LAMBDA`

Required: No

LoggingRole

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that allows a server to turn on Amazon CloudWatch logging for Amazon S3 or Amazon EFS events. When set, you can view user activity in your CloudWatch logs.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

ServerId

Specifies the unique system assigned identifier for the servers that were listed.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Required: No

State

The condition of the server that was described. A value of `ONLINE` indicates that the server can accept jobs and transfer files. A `State` value of `OFFLINE` means that the server cannot perform file transfer operations.

The states of `STARTING` and `STOPPING` indicate that the server is in an intermediate state, either not fully able to respond, or not fully offline. The values of `START_FAILED` or `STOP_FAILED` can indicate an error condition.

Type: String

Valid Values: `OFFLINE` | `ONLINE` | `STARTING` | `STOPPING` | `START_FAILED` | `STOP_FAILED`

Required: No

UserCount

Specifies the number of users that are assigned to a server you specified with the `ServerId`.

Type: Integer

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListedUser

Returns properties of the user that you specify.

Contents

Arn

Provides the unique Amazon Resource Name (ARN) for the user that you want to learn about.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: Yes

HomeDirectory

The landing directory (folder) for a user when they log in to the server using the client.

A `HomeDirectory` example is `/bucket_name/home/mydirectory`.

Note

The `HomeDirectory` parameter is only used if `HomeDirectoryType` is set to `PATH`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `(|/.*)`

Required: No

HomeDirectoryType

The type of landing directory (folder) that you want your users' home directory to be when they log in to the server. If you set it to `PATH`, the user will see the absolute Amazon S3 bucket or Amazon EFS path as is in their file transfer protocol clients. If you set it to `LOGICAL`, you need

to provide mappings in the `HomeDirectoryMappings` for how you want to make Amazon S3 or Amazon EFS paths visible to your users.

Note

If `HomeDirectoryType` is `LOGICAL`, you must provide mappings, using the `HomeDirectoryMappings` parameter. If, on the other hand, `HomeDirectoryType` is `PATH`, you provide an absolute path using the `HomeDirectory` parameter. You cannot have both `HomeDirectory` and `HomeDirectoryMappings` in your template.

Type: String

Valid Values: `PATH` | `LOGICAL`

Required: No

Role

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that controls your users' access to your Amazon S3 bucket or Amazon EFS file system. The policies attached to this role determine the level of access that you want to provide your users when transferring files into and out of your Amazon S3 bucket or Amazon EFS file system. The IAM role should also contain a trust relationship that allows the server to access your resources when servicing your users' transfer requests.

Note

The IAM role that controls your users' access to your Amazon S3 bucket for servers with `Domain=S3`, or your EFS file system for servers with `Domain=EFS`. The policies attached to this role determine the level of access you want to provide your users when transferring files into and out of your S3 buckets or EFS file systems.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

SshPublicKeyCount

Specifies the number of SSH public keys stored for the user you specified.

Type: Integer

Required: No

UserName

Specifies the name of the user whose ARN was specified. User names are used for authentication purposes.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\w][\w@.-]{2,99}`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ListedWorkflow

Contains the identifier, text description, and Amazon Resource Name (ARN) for the workflow.

Contents

Arn

Specifies the unique Amazon Resource Name (ARN) for the workflow.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 1600.

Pattern: `arn:\S+`

Required: No

Description

Specifies the text description for the workflow.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `[\w-]*`

Required: No

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `w-([a-z0-9]{17})`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

LoggingConfiguration

Consists of the logging role and the log group name.

Contents

LoggingRole

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that allows a server to turn on Amazon CloudWatch logging for Amazon S3 or Amazon EFS events. When set, you can view user activity in your CloudWatch logs.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: No

LogGroupName

The name of the CloudWatch logging group for the AWS Transfer Family server to which this workflow belongs.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 512.

Pattern: `[\.\-_\/#A-Za-z0-9]*`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

PosixProfile

The full POSIX identity, including user ID (Uid), group ID (Gid), and any secondary groups IDs (SecondaryGids), that controls your users' access to your Amazon EFS file systems. The POSIX permissions that are set on files and directories in your file system determine the level of access your users get when transferring files into and out of your Amazon EFS file systems.

Contents

Gid

The POSIX group ID used for all EFS operations by this user.

Type: Long

Valid Range: Minimum value of 0. Maximum value of 4294967295.

Required: Yes

Uid

The POSIX user ID used for all EFS operations by this user.

Type: Long

Valid Range: Minimum value of 0. Maximum value of 4294967295.

Required: Yes

SecondaryGids

The secondary POSIX group IDs used for all EFS operations by this user.

Type: Array of longs

Array Members: Minimum number of 0 items. Maximum number of 16 items.

Valid Range: Minimum value of 0. Maximum value of 4294967295.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ProtocolDetails

The protocol settings that are configured for your server.

Contents

As2Transports

Indicates the transport method for the AS2 messages. Currently, only HTTP is supported.

Type: Array of strings

Array Members: Fixed number of 1 item.

Valid Values: HTTP

Required: No

PassiveIp

Indicates passive mode, for FTP and FTPS protocols. Enter a single IPv4 address, such as the public IP address of a firewall, router, or load balancer. For example:

```
aws transfer update-server --protocol-details PassiveIp=0.0.0.0
```

Replace 0.0.0.0 in the example above with the actual IP address you want to use.

Note

If you change the `PassiveIp` value, you must stop and then restart your Transfer Family server for the change to take effect. For details on using passive mode (PASV) in a NAT environment, see [Configuring your FTPS server behind a firewall or NAT with AWS Transfer Family](#).

Special values

The `AUTO` and `0.0.0.0` are special values for the `PassiveIp` parameter. The value `PassiveIp=AUTO` is assigned by default to FTP and FTPS type servers. In this case, the server automatically responds with one of the endpoint IPs within the PASV response. `PassiveIp=0.0.0.0` has a more unique application for its usage. For example, if you have a High Availability (HA) Network Load Balancer (NLB) environment, where you have 3 subnets,

you can only specify a single IP address using the `PassiveIp` parameter. This reduces the effectiveness of having High Availability. In this case, you can specify `PassiveIp=0.0.0.0`. This tells the client to use the same IP address as the Control connection and utilize all AZs for their connections. Note, however, that not all FTP clients support the `PassiveIp=0.0.0.0` response. FileZilla and WinSCP do support it. If you are using other clients, check to see if your client supports the `PassiveIp=0.0.0.0` response.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 15.

Required: No

SetStatOption

Use the `SetStatOption` to ignore the error that is generated when the client attempts to use SETSTAT on a file you are uploading to an S3 bucket.

Some SFTP file transfer clients can attempt to change the attributes of remote files, including timestamp and permissions, using commands, such as SETSTAT when uploading the file. However, these commands are not compatible with object storage systems, such as Amazon S3. Due to this incompatibility, file uploads from these clients can result in errors even when the file is otherwise successfully uploaded.

Set the value to `ENABLE_NO_OP` to have the Transfer Family server ignore the SETSTAT command, and upload files without needing to make any changes to your SFTP client. While the `SetStatOption` `ENABLE_NO_OP` setting ignores the error, it does generate a log entry in Amazon CloudWatch Logs, so you can determine when the client is making a SETSTAT call.

Note

If you want to preserve the original timestamp for your file, and modify other file attributes using SETSTAT, you can use Amazon EFS as backend storage with Transfer Family.

Type: String

Valid Values: DEFAULT | ENABLE_NO_OP

Required: No

TlsSessionResumptionMode

A property used with Transfer Family servers that use the FTPS protocol. TLS Session Resumption provides a mechanism to resume or share a negotiated secret key between the control and data connection for an FTPS session. `TlsSessionResumptionMode` determines whether or not the server resumes recent, negotiated sessions through a unique session ID. This property is available during `CreateServer` and `UpdateServer` calls. If a `TlsSessionResumptionMode` value is not specified during `CreateServer`, it is set to `ENFORCED` by default.

- **DISABLED:** the server does not process TLS session resumption client requests and creates a new TLS session for each request.
- **ENABLED:** the server processes and accepts clients that are performing TLS session resumption. The server doesn't reject client data connections that do not perform the TLS session resumption client processing.
- **ENFORCED:** the server processes and accepts clients that are performing TLS session resumption. The server rejects client data connections that do not perform the TLS session resumption client processing. Before you set the value to `ENFORCED`, test your clients.

Note

Not all FTPS clients perform TLS session resumption. So, if you choose to enforce TLS session resumption, you prevent any connections from FTPS clients that don't perform the protocol negotiation. To determine whether or not you can use the `ENFORCED` value, you need to test your clients.

Type: String

Valid Values: `DISABLED` | `ENABLED` | `ENFORCED`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

S3FileLocation

Specifies the details for the file location for the file that's being used in the workflow. Only applicable if you are using S3 storage.

Contents

Bucket

Specifies the S3 bucket that contains the file being used.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 63.

Pattern: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Required: No

Etag

The entity tag is a hash of the object. The ETag reflects changes only to the contents of an object, not its metadata.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 65536.

Pattern: `.+`

Required: No

Key

The name assigned to the file when it was created in Amazon S3. You use the object key to retrieve the object.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `[\P{M}\p{M}]*`

Required: No

VersionId

Specifies the file version.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: . +

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

S3InputFileLocation

Specifies the customer input Amazon S3 file location. If it is used inside `copyStepDetails.DestinationFileLocation`, it should be the S3 copy destination.

You need to provide the bucket and key. The key can represent either a path or a file. This is determined by whether or not you end the key value with the forward slash (/) character. If the final character is "/", then your file is copied to the folder, and its name does not change. If, rather, the final character is alphanumeric, your uploaded file is renamed to the path value. In this case, if a file with that name already exists, it is overwritten.

For example, if your path is `shared-files/bob/`, your uploaded files are copied to the `shared-files/bob/` folder. If your path is `shared-files/today`, each uploaded file is copied to the `shared-files` folder and named `today`: each upload overwrites the previous version of the `bob` file.

Contents

Bucket

Specifies the S3 bucket for the customer input file.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 63.

Pattern: `[a-z0-9][\.-a-z0-9]{1,61}[a-z0-9]`

Required: No

Key

The name assigned to the file when it was created in Amazon S3. You use the object key to retrieve the object.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `[\P{M}\p{M}]*`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

S3StorageOptions

The Amazon S3 storage options that are configured for your server.

Contents

DirectoryListingOptimization

Specifies whether or not performance for your Amazon S3 directories is optimized. This is disabled by default.

By default, home directory mappings have a TYPE of DIRECTORY. If you enable this option, you would then need to explicitly set the HomeDirectoryMapEntry Type to FILE if you want a mapping to have a file target.

Type: String

Valid Values: ENABLED | DISABLED

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

S3Tag

Specifies the key-value pair that are assigned to a file during the execution of a Tagging step.

Contents

Key

The name assigned to the tag that you create.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: (`[\\p{L}\\p{Z}\\p{N}_. :/=+\\-@]*`)

Required: Yes

Value

The value that corresponds to the key.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: (`[\\p{L}\\p{Z}\\p{N}_. :/=+\\-@]*`)

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ServiceMetadata

A container object for the session details that are associated with a workflow.

Contents

UserDetails

The Server ID (`ServerId`), Session ID (`SessionId`) and user (`UserName`) make up the `UserDetails`.

Type: [UserDetails](#) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SftpConnectorConfig

Contains the details for an SFTP connector object. The connector object is used for transferring files to and from a partner's SFTP server.

Note

Because the `SftpConnectorConfig` data type is used for both creating and updating SFTP connectors, its parameters, `TrustedHostKeys` and `UserSecretId` are marked as not required. This is a bit misleading, as they are not required when you are updating an existing SFTP connector, but *are required* when you are creating a new SFTP connector.

Contents

TrustedHostKeys

The public portion of the host key, or keys, that are used to identify the external server to which you are connecting. You can use the `ssh-keyscan` command against the SFTP server to retrieve the necessary key.

The three standard SSH public key format elements are `<key type>`, `<body base64>`, and an optional `<comment>`, with spaces between each element. Specify only the `<key type>` and `<body base64>`: do not enter the `<comment>` portion of the key.

For the trusted host key, AWS Transfer Family accepts RSA and ECDSA keys.

- For RSA keys, the `<key type>` string is `ssh-rsa`.
- For ECDSA keys, the `<key type>` string is either `ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp384`, or `ecdsa-sha2-nistp521`, depending on the size of the key you generated.

Run this command to retrieve the SFTP server host key, where your SFTP server name is `ftp.host.com`.

```
ssh-keyscan ftp.host.com
```

This prints the public host key to standard output.

```
ftp.host.com ssh-rsa AAAAB3Nza...<long-string-for-public-key
```

Copy and paste this string into the `TrustedHostKeys` field for the `create-connector` command or into the **Trusted host keys** field in the console.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

UserSecretId

The identifier for the secret (in AWS Secrets Manager) that contains the SFTP user's private key, password, or both. The identifier must be the Amazon Resource Name (ARN) of the secret.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SshPublicKey

Provides information about the public Secure Shell (SSH) key that is associated with a Transfer Family user for the specific file transfer protocol-enabled server (as identified by `ServerId`). The information returned includes the date the key was imported, the public key contents, and the public key ID. A user can store more than one SSH public key associated with their user name on a specific server.

Contents

DateImported

Specifies the date that the public key was added to the Transfer Family user.

Type: Timestamp

Required: Yes

SshPublicKeyBody

Specifies the content of the SSH public key as specified by the `PublicKeyId`.

AWS Transfer Family accepts RSA, ECDSA, and ED25519 keys.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 2048.

Required: Yes

SshPublicKeyId

Specifies the `SshPublicKeyId` parameter contains the identifier of the public key.

Type: String

Length Constraints: Fixed length of 21.

Pattern: `key-[0-9a-f]{17}`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Tag

Creates a key-value pair for a specific resource. Tags are metadata that you can use to search for and group a resource for various purposes. You can apply tags to servers, users, and roles. A tag key can take more than one value. For example, to group servers for accounting purposes, you might create a tag called `Group` and assign the values `Research` and `Accounting` to that group.

Contents

Key

The name assigned to the tag that you create.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 128.

Required: Yes

Value

Contains one or more values that you assigned to the key name you create.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

TagStepDetails

Each step type has its own `StepDetails` structure.

The key/value pairs used to tag a file during the execution of a workflow step.

Contents

Name

The name of the step, used as an identifier.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 30.

Pattern: `[\w-]*`

Required: No

SourceFileLocation

Specifies which file to use as input to the workflow step: either the output from the previous step, or the originally uploaded file for the workflow.

- To use the previous file as the input, enter `${previous.file}`. In this case, this workflow step uses the output file from the previous workflow step as input. This is the default value.
- To use the originally uploaded file location as input for this step, enter `${original.file}`.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `\$\{(\w+.\w+)\}`

Required: No

Tags

Array that contains from 1 to 10 key/value pairs.

Type: Array of [S3Tag](#) objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

UserDetails

Specifies the user name, server ID, and session ID for a workflow.

Contents

ServerId

The system-assigned unique identifier for a Transfer server instance.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `s-([0-9a-f]{17})`

Required: Yes

UserName

A unique string that identifies a Transfer Family user associated with a server.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 100.

Pattern: `[\w][\w@.-]{2,99}`

Required: Yes

SessionId

The system-assigned unique identifier for a session that corresponds to the workflow.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 32.

Pattern: `[\w-]*`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

WorkflowDetail

Specifies the workflow ID for the workflow to assign and the execution role that's used for executing the workflow.

In addition to a workflow to execute when a file is uploaded completely, `WorkflowDetails` can also contain a workflow ID (and execution role) for a workflow to execute on partial upload. A partial upload occurs when the server session disconnects while the file is still being uploaded.

Contents

ExecutionRole

Includes the necessary permissions for S3, EFS, and Lambda operations that Transfer can assume, so that all workflow steps can operate on the required resources

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:.*role/\S+`

Required: Yes

WorkflowId

A unique identifier for the workflow.

Type: String

Length Constraints: Fixed length of 19.

Pattern: `w-([a-z0-9]{17})`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

WorkflowDetails

Container for the `WorkflowDetail` data type. It is used by actions that trigger a workflow to begin execution.

Contents

OnPartialUpload

A trigger that starts a workflow if a file is only partially uploaded. You can attach a workflow to a server that executes whenever there is a partial upload.

A *partial upload* occurs when a file is open when the session disconnects.

Note

`OnPartialUpload` can contain a maximum of one `WorkflowDetail` object.

Type: Array of [WorkflowDetail](#) objects

Array Members: Minimum number of 0 items. Maximum number of 1 item.

Required: No

OnUpload

A trigger that starts a workflow: the workflow begins to execute after a file is uploaded.

To remove an associated workflow from a server, you can provide an empty `OnUpload` object, as in the following example.

```
aws transfer update-server --server-id s-01234567890abcdef --workflow-  
details '{"OnUpload":[]}'
```

Note

`OnUpload` can contain a maximum of one `WorkflowDetail` object.

Type: Array of [WorkflowDetail](#) objects

Array Members: Minimum number of 0 items. Maximum number of 1 item.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

WorkflowStep

The basic building block of a workflow.

Contents

CopyStepDetails

Details for a step that performs a file copy.

Consists of the following values:

- A description
- An Amazon S3 location for the destination of the file copy.
- A flag that indicates whether to overwrite an existing file of the same name. The default is FALSE.

Type: [CopyStepDetails](#) object

Required: No

CustomStepDetails

Details for a step that invokes an AWS Lambda function.

Consists of the Lambda function's name, target, and timeout (in seconds).

Type: [CustomStepDetails](#) object

Required: No

DecryptStepDetails

Details for a step that decrypts an encrypted file.

Consists of the following values:

- A descriptive name
- An Amazon S3 or Amazon Elastic File System (Amazon EFS) location for the source file to decrypt.
- An S3 or Amazon EFS location for the destination of the file decryption.

- A flag that indicates whether to overwrite an existing file of the same name. The default is FALSE.
- The type of encryption that's used. Currently, only PGP encryption is supported.

Type: [DecryptStepDetails](#) object

Required: No

DeleteStepDetails

Details for a step that deletes the file.

Type: [DeleteStepDetails](#) object

Required: No

TagStepDetails

Details for a step that creates one or more tags.

You specify one or more tags. Each tag contains a key-value pair.

Type: [TagStepDetails](#) object

Required: No

Type

Currently, the following step types are supported.

- **COPY** - Copy the file to another location.
- **CUSTOM** - Perform a custom step with an AWS Lambda function target.
- **DECRYPT** - Decrypt a file that was encrypted before it was uploaded.
- **DELETE** - Delete the file.
- **TAG** - Add a tag to the file.

Type: String

Valid Values: COPY | CUSTOM | TAG | DELETE | DECRYPT

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Making API requests

In addition to using the console, you can use the AWS Transfer Family API to programmatically configure and manage your servers. This section describes the AWS Transfer Family operations, request signing for authentication and the error handling. For information about the regions and endpoints available for Transfer Family, see [AWS Transfer Family endpoints and quotas](#) in the *AWS General Reference*

Note

You can also use the AWS SDKs when developing applications with Transfer Family;. The AWS SDKs for Java, .NET, and PHP wrap the underlying Transfer Family API, simplifying your programming tasks. For information about downloading the SDK libraries, see [Sample code libraries](#).

Topics

- [Transfer Family required request headers](#)
- [Transfer Family request inputs and signing](#)
- [Error responses](#)
- [Available libraries](#)

Transfer Family required request headers

This section describes the required headers that you must send with every POST request to AWS Transfer Family. You include HTTP headers to identify key information about the request including the operation you want to invoke, the date of the request, and information that indicates the

authorization of you as the sender of the request. Headers are case insensitive and the order of the headers is not important.

The following example shows headers that are used in the [ListServers](#) operation.

```
POST / HTTP/1.1
Host: transfer.us-east-1.amazonaws.com
x-amz-target: TransferService.ListServers
x-amz-date: 20220507T012034Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIDEXAMPLE/20220507/us-east-1/transfer/
aws4_request,
    SignedHeaders=content-type;host;x-amz-date;x-amz-target,
    Signature=13550350a8681c84c861aac2e5b440161c2b33a3e4f302ac680ca5b686de48de
Content-Type: application/x-amz-json-1.1
Content-Length: 17

{"MaxResults":10}
```

The following are the headers that must include with your POST requests to Transfer Family. Headers shown below that begin with "x-amz" are specific for AWS. All other headers listed are common header used in HTTP transactions.

Header	Description
Authorization	Authorization header is required. The format is standard Sigv4 request signature, which is documented at Signing AWS API requests .
Content-Type	Use <code>application/x-amz-json-1.1</code> as the content type for all requests to Transfer Family. <div style="border: 1px solid #ccc; border-radius: 10px; padding: 5px; width: fit-content; margin: 10px auto;">Content-Type: application/x-amz-json-1.1</div>
Host	Use the host header to specify the Transfer Family endpoint where you send your request. For example, <code>transfer.us-east-1.amazonaws.com</code> is the endpoint for the US East (Ohio) region. For more information about the endpoints available for Transfer Family, see AWS Transfer Family endpoints and quotas in the <i>AWS General Reference</i> .

Header	Description
	<p>Host: transfer. <i>region</i>.amazonaws.com</p>
x-amz-date	<p>You must provide the time stamp in either the HTTP Date header or the AWS x-amz-date header. (Some HTTP client libraries don't let you set the Date header.) When an x-amz-date header is present, the Transfer Family ignores any Date header during the request authentication. The x-amz-date format must be ISO8601, in the YYYYMMDD'T'HHMMSS'Z' format.</p> <p>x-amz-date: <i>YYYYMMDD'T'HHMMSS'Z'</i></p>
x-amz-target	<p>This header specifies the version of the API and the operation that you are requesting. The target header values are formed by concatenating the API version with the API name and are in the following format.</p> <p>x-amz-target: TransferService. <i>operationName</i></p> <p>The <i>operationName</i> value (for example ListServers) can be found from the API list, ListServers.</p>
x-amz-security-token	<p>This header is required when credentials used to sign the request are temporary or session credentials (for details, see Using temporary credentials with AWS resources in the <i>IAM User Guide</i>. See Add the signature to the HTTP request in the Amazon Web Services General Reference for more information.</p>

Transfer Family request inputs and signing

All request inputs must be sent as part of JSON payload in request body. For Actions in which all request fields are optional, for example ListServers, you still need to provide an empty JSON

object in the request body, such as `{}`. The structure of Transfer Family payload request/response is documented in existing the API reference, for example [DescribeServer](#).

Transfer Family supports authentication using AWS Signature Version 4. For details, see [Signing AWS API requests](#).

Error responses

When there is an error, the response header information contains:

- Content-Type: `application/x-amz-json-1.1`
- An appropriate 4xx or 5xx HTTP status code

The body of an error response contains information about the error that occurred. The following sample error response shows the output syntax of response elements common to all error responses.

```
{
  "__type": "String",
  "Message": "String", <!-- Message is lowercase in some instances -->
  "Resource": String,
  "ResourceType": String
  "RetryAfterSeconds": String
}
```

The following table explains the JSON error response fields shown in the preceding syntax.

__type

One of the exceptions from a Transfer Family API call.

Type: String

Message or message

One of the operation error code messages.

Note

Some exceptions use `message`, and others use `Message`. You can check the code for your interface to determine the proper case. Alternatively, you can test each option to see which works.

Type: String

Resource

The resource for which the error is invoked. For example, if you try to create a user that already exists, the `Resource` is the username for the existing user.

Type: String

ResourceType

The resource type for which the error is invoked. For example, if you try to create a user that already exists, the `ResourceType` is `User`.

Type: String

RetryAfterSeconds

The number of seconds to wait before retrying the command.

Type: String

Error response examples

The following JSON body is returned if you call the `DescribeServer` API and specify a server that does not exist.

```
{
  "__type": "ResourceNotFoundException",
  "Message": "Unknown server",
  "Resource": "s-11112222333344444",
  "ResourceType": "Server"
}
```

The following JSON body is returned if executing an API causes throttling to occur.

```
{
  "__type": "ThrottlingException",
  "RetryAfterSeconds": "1"
}
```

The following JSON body is returned if you use the `CreateServer` API and you do not have sufficient permissions to create a Transfer Family server.

```
{
  "__type": "AccessDeniedException",
  "Message": "You do not have sufficient access to perform this action."
}
```

The following JSON body is returned if you use the `CreateUser` API and specify a user that already exists.

```
{
  "__type": "ResourceExistsException",
  "Message": "User already exists",
  "Resource": "Alejandro-Rosalez",
  "ResourceType": "User"
}
```

Available libraries

AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of the command-line tools and Query API. These libraries provide basic functions (not included in the APIs), such as request authentication, request retries, and error handling so that it is easier to get started. See [Tools to build on AWS](#)

For libraries and sample code in all languages, see [Sample code & libraries](#).

Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see [Signing AWS API requests](#) in the *IAM User Guide*.

Action

The action to be performed.

Type: string

Required: Yes

Version

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Type: string

Required: Yes

X-Amz-Algorithm

The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: AWS4-HMAC-SHA256

Required: Conditional

X-Amz-Credential

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: *access_key/YYYYMMDD/region/service/aws4_request*.

For more information, see [Create a signed AWS API request](#) in the *IAM User Guide*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-Date

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: 20120325T120000Z.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see [Elements of an AWS API request signature](#) in the *IAM User Guide*.

Type: string

Required: Conditional

X-Amz-Security-Token

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS STS, see [AWS services that work with IAM](#) in the *IAM User Guide*.

Condition: If you're using temporary security credentials from AWS STS, you must include the security token.

Type: string

Required: Conditional

X-Amz-Signature

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-SignedHeaders

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see [Create a signed AWS API request](#) in the *IAM User Guide*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

Common Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

IncompleteSignature

The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure

The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

InvalidAction

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientTokenId

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

NotAuthorized

You do not have permission to perform this action.

HTTP Status Code: 400

OptInRequired

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationError

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

Document history for AWS Transfer Family

The following table describes the documentation for this release of AWS Transfer Family.

- **API version:** transfer-2018-11-05
- **Latest documentation update:** April 23, 2024

Change	Description	Date
Ability for SFTP connectors to list remote files and directories	Transfer Family has added the ability for our customers to use SFTP connectors to list files stored in remote SFTP servers. For details, see List contents of a remote directory	April 23, 2024
Ability to use a trading partner's self-signed TLS certificate with AS2 message exchange	AWS Transfer Family has added the option to import and use a trading partner's public, self-signed TLS certificate for sending Applicability Statement 2 (AS2) messages to their server over HTTPS .	April 12, 2024
Addition of security policies for SFTP connectors	AWS Transfer Family has added security policies for use with SFTP connectors. For details, see Security policies for AWS Transfer Family SFTP connectors .	April 5, 2024
Integrate with Amazon EventBridge	AWS Transfer Family now automatically publishes events to Amazon EventBridge	February 8, 2024

Change	Description	Date
	ge for all file transfer operations. For details, see Managing Transfer Family events using Amazon EventBridge .	
Addition of new security policies	AWS Transfer Family has added new FIPS and non-FIPS security policies. Also, the default security policy assigned to servers is always the latest security policy. For details, see Security policies for AWS Transfer Family servers .	February 5, 2024
Support for static IP addresses for SFTP connectors and AS2	Transfer Family now provides static IP addresses for SFTP connectors and AS2. This enables connection with remote SFTP servers that are secured by IP allowlisting controls. For AS2, we're introducing static IP addresses for asynchronous MDN responses from AS2 servers.	January 16, 2024
The user guide has been reorganized to align more closely with the latest version of AWS Transfer Family.	Transfer Family has added multiple features since the guide originated, necessitating a restructuring of the guide.	January 3, 2024

Change	Description	Date
<p>Logical directory mappings enhancements</p> <p>Amazon S3 list performance optimization</p>	<p>Transfer Family now supports logical directory mappings up to 2.1 MB. You can also now declare whether a user mapping is to a file. For more information, see Rules for using logical directories.</p> <p>When creating or updating a server that uses Amazon S3 for storage, you can now optimize the performance of listing your S3 directories (or folders). For more information, see Configuring an SFTP, FTPS, or FTP server endpoint.</p>	<p>November 17, 2023</p>
<p>Alternate port for SFTP servers with virtual private cloud (VPC) endpoints</p>	<p>You can now enable an alternate nonstandard port for your SFTP Transfer Family servers that have VPC endpoints. For more information, see Create a server in a virtual private cloud.</p>	<p>November 17, 2023</p>
<p>Support for SFTP connectors</p>	<p>SFTP Connectors extend the capabilities of AWS Transfer Family to communicate with remote servers both in the cloud and on-premises. For more information, see Send and retrieve files by using an SFTP connector.</p>	<p>July 25, 2023</p>

Change	Description	Date
Support for AS2 Basic authentication	Transfer Family now supports using Basic authentication for servers that use the Applicability Statement 2 (AS2) protocol. For more information, see Basic authentication for AS2 connectors .	June 30, 2023
Support for structured JSON logging	Transfer Family now supports delivering structured JSON logs to Amazon CloudWatch, grouping log streams into custom log groups, and performing common log queries across protocols. For more information, see Amazon CloudWatch logging for AWS Transfer Family .	June 24, 2023
Support for multiple methods of authentication	Transfer Family has support for authenticating by using a password, a public/private key pair, or both. This is available for servers that use the SFTP protocol and a custom identity provider. For more information, see Create an SFTP-enabled server .	May 17, 2023

Change	Description	Date
Support for Pretty Good Privacy (PGP) decryption with files that Transfer Family processes with workflows	Transfer Family has built-in support for Pretty Good Privacy (PGP) decryption. You can use PGP decryption on files that are uploaded over SFTP, FTPS, or FTP to Amazon Simple Storage Service (Amazon S3) or Amazon Elastic File System (Amazon EFS). For more information, see Generate and manage PGP keys and Use PGP decryption in your workflow .	December 21, 2022
Fully managed support for Applicability Statement 2 (AS2) file transfer protocol with Transfer Family servers	You can create servers that use the AS2 protocol for sending and receiving information to and from trading partners who are inside or outside the AWS environment. For more information, see Configuring AS2 .	July 25, 2022

Change	Description	Date
Support for display banners when creating a server	You can add customized messages when you create servers. You can display a pre-authentication message (all protocols), and a post-authentication message (for FTP and FTPS servers). For more information, see Create an SFTP-enabled server , Create an FTPS-enabled server , or Create an FTP-enabled server .	February 17, 2022
Support for AWS Lambda as an identity provider	You can now connect to a custom identity provider using AWS Lambda with their Transfer Family servers. Previously, you had to supply an Amazon API Gateway URL to integrate a custom identity provider. For more information, see Using AWS Lambda to integrate your identity provider .	November 16, 2021
Support for Managed File Transfer Workflows	Managed File Transfer Workflows provide you with post-upload processing abstractions for the common tasks that you currently perform manually. For more information, see AWS Transfer Family managed workflows .	September 2, 2021

Change	Description	Date
Support for AWS Directory Service for Microsoft Active Directory	In addition to service managed and custom identity providers, you can now use AWS Directory Service for Microsoft Active Directory to manage user access for authentication and authorization. For more information, see Using AWS Directory Service for Microsoft Active Directory .	May 24, 2021
New AWS Regions	AWS Transfer Family is now available in the Africa (Cape Town) Region. For more information about Transfer Family endpoints, see AWS Transfer Family endpoints and quotas in the <i>AWS General Reference</i> .	February 24, 2021
New AWS Regions	AWS Transfer Family is now available in the Asia Pacific (Hong Kong) and Middle East (Bahrain) Regions. For more information about Transfer Family endpoints , see AWS Transfer Family endpoints and quotas in the <i>AWS General Reference</i> .	February 17, 2021

Change	Description	Date
Support for Amazon EFS as a data store	Transfer Family now supports file transfers into and out of Amazon Elastic File System (Amazon EFS). Amazon EFS is a simple, scalable, fully managed elastic NFS file system. For more information, see Configure an Amazon EFS file system .	January 06, 2021
Support for AWS WAF	Transfer Family now supports AWS WAF, a web application firewall that helps protect web applications and API operations from attacks. For more information, see Add a web application firewall .	November 24, 2020
Support for multiple security groups in a virtual private cloud (VPC)	You can now attach multiple security groups to a server in a VPC. For more information, see Create a server in a virtual private cloud .	October 15, 2020

Change	Description	Date
New AWS Regions	Transfer Family is now available in the AWS GovCloud (US) Regions. For more information about Transfer Family endpoints for AWS GovCloud (US) Regions, see AWS Transfer Family endpoints and quotas in the <i>AWS General Reference</i> . For information about using Transfer Family in the AWS GovCloud (US) Regions, see AWS Transfer Family in the <i>AWS GovCloud (US) User Guide</i> .	September 30, 2020
A security policy with supported cryptographic algorithms can now be attached to your server	You can now attach a security policy that contains a set of supported cryptographic algorithms to your server. For more information, see Security policies for AWS Transfer Family servers .	August 12, 2020

Change	Description	Date
Support for Federal Information Processing Standard (FIPS) endpoints	FIPS-enabled endpoints are now available in North American AWS Regions. For available Regions, see AWS Transfer Family endpoints and quotas in the <i>AWS General Reference</i> . To enable FIPS for an SFTP-enabled server endpoint, see Create an SFTP-enabled server . To enable FIPS for an FTPS-enabled server endpoint, see Create an FTPS-enabled server . To enable FIPS for an FTP-enabled server endpoint, see Create an FTP-enabled server .	August 12, 2020
Username character-length increase and additional allowed characters	Usernames can now contain at signs (@) and periods (.), and can be a maximum length of 100 characters. To add a user, see Managing users for server endpoints .	August 12, 2020
Support for automatic Amazon CloudWatch logging AWS Identity and Access Management (IAM) role creation	Transfer Family now supports automatic creation of a CloudWatch logging IAM role to view end-user activity. For more information, see Create an SFTP-enabled server , Create an FTPS-enabled server , or Create an FTP-enabled server .	July 30, 2020

Change	Description	Date
AWS Transfer Family now supports Source IP as a factor for authorization.	Transfer Family adds support for using end users' source IP addresses as a factor for authorization, enabling you to apply an additional layer of security when authorizing access over Secure File Transfer Protocol (SFTP), File Transfer Protocol over SSL (FTPS), or File Transfer Protocol (FTP). For more information, see Working with custom identity providers .	June 9, 2020
AWS Transfer for SFTP is now AWS Transfer Family and adds support for FTP and FTPS.	You can now use two additional protocols for your users' file transfers: File Transfer Protocol Secure (FTPS) and File Transfer Protocol (FTP). Users can move, run, secure, and integrate FTP over SSL (FTPS) and plaintext FTP based workflows in AWS, in addition to existing Secure File Transfer Protocol (SFTP) support.	April 23, 2020

Change	Description	Date
Support for virtual private cloud (VPC) security groups and Elastic IP addresses	You can now create an allowlist for incoming IP addresses using security groups, providing an additional layer of security for servers. You can also associate Elastic IP addresses with your server's endpoint. By doing this, you can enable users behind firewalls to allow access to that endpoint. For more information, see Create a server in a virtual private cloud .	January 10, 2020
Support for working in a VPC	You can now create a server in a VPC. You can use your server to transfer data over your client to and from an Amazon S3 bucket without going over the public internet. For more information, see Create a server in a virtual private cloud .	March 27, 2019
First version of AWS Transfer Family released.	This initial release includes setting up directions, describes how to get started, and provides information on client configuration, user configuration, and monitoring activity.	November 25, 2018

AWS Glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS Glossary Reference*.