
AWS Security Token Service

API Reference

API Version 2011-06-15



AWS Security Token Service: API Reference

Copyright © 2015 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks of Amazon Web Services, Inc.: Amazon, Amazon Web Services Design, AWS, Amazon CloudFront, AWS CloudTrail, AWS CodeDeploy, Amazon Cognito, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Amazon Kinesis, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC, and Amazon WorkDocs. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. 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

Welcome	1
Actions	2
AssumeRole	3
Request Parameters	4
Response Elements	5
Errors	6
Examples	6
AssumeRoleWithSAML	8
Request Parameters	8
Response Elements	9
Errors	10
AssumeRoleWithWebIdentity	12
Request Parameters	13
Response Elements	14
Errors	15
Examples	16
DecodeAuthorizationMessage	17
Request Parameters	17
Response Elements	17
Errors	18
Examples	18
GetFederationToken	20
Request Parameters	21
Response Elements	22
Errors	22
Examples	22
GetSessionToken	24
Request Parameters	24
Response Elements	25
Errors	25
Examples	25
Data Types	27
AssumedRoleUser	27
Description	27
Contents	27
Credentials	28
Description	28
Contents	28
FederatedUser	28
Description	28
Contents	29
Common Parameters	30
.....	30
Common Errors	32
.....	32

Welcome

The AWS Security Token Service (STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users). This guide provides descriptions of the STS API. For more detailed information about using this service, go to [Using Temporary Security Credentials](#).

Note

As an alternative to using the API, you can use one of the AWS SDKs, which consist of libraries and sample code for various programming languages and platforms (Java, Ruby, .NET, iOS, Android, etc.). The SDKs provide a convenient way to create programmatic access to STS. For example, the SDKs take care of cryptographically signing requests, managing errors, and retrying requests automatically. For information about the AWS SDKs, including how to download and install them, see the [Tools for Amazon Web Services page](#).

For information about setting up signatures and authorization through the API, go to [Signing AWS API Requests](#) in the *AWS General Reference*. For general information about the Query API, go to [Making Query Requests](#) in *Using IAM*. For information about using security tokens with other AWS products, go to [Using Temporary Security Credentials to Access AWS](#) in *Using Temporary Security Credentials*.

If you're new to AWS and need additional technical information about a specific AWS product, you can find the product's technical documentation at <http://aws.amazon.com/documentation/>.

Endpoints

The AWS Security Token Service (STS) has a default endpoint of <https://sts.amazonaws.com> that maps to the US East (N. Virginia) region. Additional regions are available, but must first be activated in the AWS Management Console before you can use a different region's endpoint. For more information about activating a region for STS see [Activating STS in a New Region](#) in the *Using Temporary Security Credentials* guide.

For information about STS endpoints, see [Regions and Endpoints](#) in the *AWS General Reference*.

Recording API requests

STS supports AWS CloudTrail, which is a service that records AWS calls for your AWS account and delivers log files to an Amazon S3 bucket. By using information collected by CloudTrail, you can determine what requests were successfully made to STS, who made the request, when it was made, and so on. To learn more about CloudTrail, including how to turn it on and find your log files, see the [AWS CloudTrail User Guide](#).

This document was last updated on September 3, 2015.

Actions

The following actions are supported:

- [AssumeRole](#) (p. 3)
- [AssumeRoleWithSAML](#) (p. 8)
- [AssumeRoleWithWebIdentity](#) (p. 12)
- [DecodeAuthorizationMessage](#) (p. 17)
- [GetFederationToken](#) (p. 20)
- [GetSessionToken](#) (p. 24)

AssumeRole

Returns a set of temporary security credentials (consisting of an access key ID, a secret access key, and a security token) that you can use to access AWS resources that you might not normally have access to. Typically, you use `AssumeRole` for cross-account access or federation.

Important: You cannot call `AssumeRole` by using AWS account credentials; access will be denied. You must use IAM user credentials or temporary security credentials to call `AssumeRole`.

For cross-account access, imagine that you own multiple accounts and need to access resources in each account. You could create long-term credentials in each account to access those resources. However, managing all those credentials and remembering which one can access which account can be time consuming. Instead, you can create one set of long-term credentials in one account and then use temporary security credentials to access all the other accounts by assuming roles in those accounts. For more information about roles, see [IAM Roles \(Delegation and Federation\)](#) in *Using IAM*.

For federation, you can, for example, grant single sign-on access to the AWS Management Console. If you already have an identity and authentication system in your corporate network, you don't have to recreate user identities in AWS in order to grant those user identities access to AWS. Instead, after a user has been authenticated, you call `AssumeRole` (and specify the role with the appropriate permissions) to get temporary security credentials for that user. With those temporary security credentials, you construct a sign-in URL that users can use to access the console. For more information, see [Scenarios for Granting Temporary Access](#) in *Using Temporary Security Credentials*.

The temporary security credentials are valid for the duration that you specified when calling `AssumeRole`, which can be from 900 seconds (15 minutes) to 3600 seconds (1 hour). The default is 1 hour.

Optionally, you can pass an IAM access policy to this operation. If you choose not to pass a policy, the temporary security credentials that are returned by the operation have the permissions that are defined in the access policy of the role that is being assumed. If you pass a policy to this operation, the temporary security credentials that are returned by the operation have the permissions that are allowed by both the access policy of the role that is being assumed, *and* the policy that you pass. This gives you a way to further restrict the permissions for the resulting temporary security credentials. You cannot use the passed policy to grant permissions that are in excess of those allowed by the access policy of the role that is being assumed. For more information, see [Permissions for AssumeRole, AssumeRoleWithSAML, and AssumeRoleWithWebIdentity](#) in *Using Temporary Security Credentials*.

To assume a role, your AWS account must be trusted by the role. The trust relationship is defined in the role's trust policy when the role is created. You must also have a policy that allows you to call `sts:AssumeRole`.

Using MFA with AssumeRole

You can optionally include multi-factor authentication (MFA) information when you call `AssumeRole`. This is useful for cross-account scenarios in which you want to make sure that the user who is assuming the role has been authenticated using an AWS MFA device. In that scenario, the trust policy of the role being assumed includes a condition that tests for MFA authentication; if the caller does not include valid MFA information, the request to assume the role is denied. The condition in a trust policy that tests for MFA authentication might look like the following example.

```
"Condition": {"Bool": {"aws:MultiFactorAuthPresent": true}}
```

For more information, see [Configuring MFA-Protected API Access](#) in *Using IAM* guide.

To use MFA with `AssumeRole`, you pass values for the `SerialNumber` and `TokenCode` parameters. The `SerialNumber` value identifies the user's hardware or virtual MFA device. The `TokenCode` is the time-based one-time password (TOTP) that the MFA devices produces.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 30\)](#).

DurationSeconds

The duration, in seconds, of the role session. The value can range from 900 seconds (15 minutes) to 3600 seconds (1 hour). By default, the value is set to 3600 seconds.

Type: Integer

Valid range: Minimum value of 900. Maximum value of 3600.

Required: No

ExternalId

A unique identifier that is used by third parties when assuming roles in their customers' accounts. For each role that the third party can assume, they should instruct their customers to ensure the role's trust policy checks for the external ID that the third party generated. Each time the third party assumes the role, they should pass the customer's external ID. The external ID is useful in order to help third parties bind a role to the customer who created it. For more information about the external ID, see [How to Use External ID When Granting Access to Your AWS Resources](#) in *Using Temporary Security Credentials*.

Type: String

Length constraints: Minimum length of 2. Maximum length of 1224.

Pattern: [\w+=, .@:\/-] *

Required: No

Policy

An IAM policy in JSON format.

This parameter is optional. If you pass a policy, the temporary security credentials that are returned by the operation have the permissions that are allowed by both (the intersection of) the access policy of the role that is being assumed, *and* the policy that you pass. This gives you a way to further restrict the permissions for the resulting temporary security credentials. You cannot use the passed policy to grant permissions that are in excess of those allowed by the access policy of the role that is being assumed. For more information, see [Permissions for AssumeRole, AssumeRoleWithSAML, and AssumeRoleWithWebIdentity](#) in *Using Temporary Security Credentials*.

Note

The policy plain text must be 2048 bytes or shorter. However, an internal conversion compresses it into a packed binary format with a separate limit. The PackedPolicySize response element indicates by percentage how close to the upper size limit the policy is, with 100% equaling the maximum allowed size.

Type: String

Length constraints: Minimum length of 1. Maximum length of 2048.

Pattern: [\u0009\u000A\u000D\u0020-\u00FF] +

Required: No

RoleArn

The Amazon Resource Name (ARN) of the role to assume.

Type: String

Length constraints: Minimum length of 20. Maximum length of 2048.

Required: Yes

RoleSessionName

An identifier for the assumed role session.

Use the role session name to uniquely identify a session when the same role is assumed by different principals or for different reasons. In cross-account scenarios, the role session name is visible to, and can be logged by the account that owns the role. The role session name is also used in the ARN of the assumed role principal. This means that subsequent cross-account API requests using the temporary security credentials will expose the role session name to the external account in their CloudTrail logs.

Type: String

Length constraints: Minimum length of 2. Maximum length of 32.

Pattern: [`\w+=, .@-`]*

Required: Yes

SerialNumber

The identification number of the MFA device that is associated with the user who is making the `AssumeRole` call. Specify this value if the trust policy of the role being assumed includes a condition that requires MFA authentication. The value is either the serial number for a hardware device (such as `GAHT12345678`) or an Amazon Resource Name (ARN) for a virtual device (such as `arn:aws:iam::123456789012:mfa/user`).

Type: String

Length constraints: Minimum length of 9. Maximum length of 256.

Pattern: [`\w+=/: , .@-`]*

Required: No

TokenCode

The value provided by the MFA device, if the trust policy of the role being assumed requires MFA (that is, if the policy includes a condition that tests for MFA). If the role being assumed requires MFA and if the `TokenCode` value is missing or expired, the `AssumeRole` call returns an "access denied" error.

Type: String

Length constraints: Minimum length of 6. Maximum length of 6.

Pattern: [`\d`]*

Required: No

Response Elements

The following elements are returned.

AssumedRoleUser

The Amazon Resource Name (ARN) and the assumed role ID, which are identifiers that you can use to refer to the resulting temporary security credentials. For example, you can reference these credentials as a principal in a resource-based policy by using the ARN or assumed role ID. The ARN and ID include the `RoleSessionName` that you specified when you called `AssumeRole`.

Type: [AssumedRoleUser](#) (p. 27)

Credentials

The temporary security credentials, which include an access key ID, a secret access key, and a security (or session) token.

Type: [Credentials \(p. 28\)](#)

PackedPolicySize

A percentage value that indicates the size of the policy in packed form. The service rejects any policy with a packed size greater than 100 percent, which means the policy exceeded the allowed space.

Type: Integer

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 32\)](#).

MalformedPolicyDocument

The request was rejected because the policy document was malformed. The error message describes the specific error.

HTTP Status Code: 400

PackedPolicyTooLarge

The request was rejected because the policy document was too large. The error message describes how big the policy document is, in packed form, as a percentage of what the API allows.

HTTP Status Code: 400

Examples

Sample Request

```
https://sts.amazonaws.com/  
?Version=2011-06-15  
&Action=AssumeRole  
&RoleSessionName=Bob  
&RoleArn=arn:aws:iam::123456789012:role/demo  
&Policy={"Version":"2012-10-17","Statement":[{"Sid":"Stmt1",  
"Effect":"Allow","Action":"s3:*","Resource":"*"}]}  
&DurationSeconds=3600  
&ExternalId=123ABC  
&AUTHPARAMS
```

Sample Response

```
<AssumeRoleResponse xmlns="https://sts.amazonaws.com/doc/  
2011-06-15/">  
  <AssumeRoleResult>  
    <Credentials>  
      <SessionToken>  
        AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22wD0k4x4HIz8j4FZTwdQW  
        LwSKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd  
        QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSDvp75YU  
        9HFv1Rd8Tx6q6fE8YQcHNVXakiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
```

AWS Security Token Service API Reference Examples

```
+scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iSIlTJabIQwj2ICCR/oLxBA==
</SessionToken>
<SecretAccessKey>
  wJalrXUtnFEMI/K7MDENG/bPxrFiCYzEXAMPLEKEY
</SecretAccessKey>
<Expiration>2011-07-15T23:28:33.359Z</Expiration>
<AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
</Credentials>
<AssumedRoleUser>
  <Arn>arn:aws:sts::123456789012:assumed-role/demo/Bob</Arn>
  <AssumedRoleId>ARO123EXAMPLE123:Bob</AssumedRoleId>
</AssumedRoleUser>
<PackedPolicySize>6</PackedPolicySize>
</AssumeRoleResult>
<ResponseMetadata>
  <RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
</ResponseMetadata>
</AssumeRoleResponse>
```

AssumeRoleWithSAML

Returns a set of temporary security credentials for users who have been authenticated via a SAML authentication response. This operation provides a mechanism for tying an enterprise identity store or directory to role-based AWS access without user-specific credentials or configuration.

The temporary security credentials returned by this operation consist of an access key ID, a secret access key, and a security token. Applications can use these temporary security credentials to sign calls to AWS services. The credentials are valid for the duration that you specified when calling `AssumeRoleWithSAML`, which can be up to 3600 seconds (1 hour) or until the time specified in the SAML authentication response's `SessionNotOnOrAfter` value, whichever is shorter.

Note

The maximum duration for a session is 1 hour, and the minimum duration is 15 minutes, even if values outside this range are specified.

Optionally, you can pass an IAM access policy to this operation. If you choose not to pass a policy, the temporary security credentials that are returned by the operation have the permissions that are defined in the access policy of the role that is being assumed. If you pass a policy to this operation, the temporary security credentials that are returned by the operation have the permissions that are allowed by both the access policy of the role that is being assumed, *and* the policy that you pass. This gives you a way to further restrict the permissions for the resulting temporary security credentials. You cannot use the passed policy to grant permissions that are in excess of those allowed by the access policy of the role that is being assumed. For more information, see [Permissions for AssumeRoleWithSAML](#) in *Using Temporary Security Credentials*.

Before your application can call `AssumeRoleWithSAML`, you must configure your SAML identity provider (IdP) to issue the claims required by AWS. Additionally, you must use AWS Identity and Access Management (IAM) to create a SAML provider entity in your AWS account that represents your identity provider, and create an IAM role that specifies this SAML provider in its trust policy.

Calling `AssumeRoleWithSAML` does not require the use of AWS security credentials. The identity of the caller is validated by using keys in the metadata document that is uploaded for the SAML provider entity for your identity provider.

For more information, see the following resources:

- [Creating Temporary Security Credentials for SAML Federation](#).
- [SAML Providers](#) in *Using IAM*.
- [Configuring a Relying Party and Claims](#) in *Using IAM*.
- [Creating a Role for SAML-Based Federation](#) in *Using IAM*.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters](#) (p. 30).

DurationSeconds

The duration, in seconds, of the role session. The value can range from 900 seconds (15 minutes) to 3600 seconds (1 hour). By default, the value is set to 3600 seconds. An expiration can also be specified in the SAML authentication response's `SessionNotOnOrAfter` value. The actual expiration time is whichever value is shorter.

Note

The maximum duration for a session is 1 hour, and the minimum duration is 15 minutes, even if values outside this range are specified.

Type: Integer

Valid range: Minimum value of 900. Maximum value of 3600.

Required: No

Policy

An IAM policy in JSON format.

The policy parameter is optional. If you pass a policy, the temporary security credentials that are returned by the operation have the permissions that are allowed by both the access policy of the role that is being assumed, *and* the policy that you pass. This gives you a way to further restrict the permissions for the resulting temporary security credentials. You cannot use the passed policy to grant permissions that are in excess of those allowed by the access policy of the role that is being assumed. For more information, see [Permissions for AssumeRoleWithSAML](#) in *Using Temporary Security Credentials*.

Note

The policy plain text must be 2048 bytes or shorter. However, an internal conversion compresses it into a packed binary format with a separate limit. The PackedPolicySize response element indicates by percentage how close to the upper size limit the policy is, with 100% equaling the maximum allowed size.

Type: String

Length constraints: Minimum length of 1. Maximum length of 2048.

Pattern: [\u0009\u000A\u000D\u0020-\u00FF] +

Required: No

PrincipalArn

The Amazon Resource Name (ARN) of the SAML provider in IAM that describes the IdP.

Type: String

Length constraints: Minimum length of 20. Maximum length of 2048.

Required: Yes

RoleArn

The Amazon Resource Name (ARN) of the role that the caller is assuming.

Type: String

Length constraints: Minimum length of 20. Maximum length of 2048.

Required: Yes

SAMLAssertion

The base-64 encoded SAML authentication response provided by the IdP.

For more information, see [Configuring a Relying Party and Adding Claims](#) in the *Using IAM* guide.

Type: String

Length constraints: Minimum length of 4. Maximum length of 50000.

Required: Yes

Response Elements

The following elements are returned.

AssumedRoleUser

The identifiers for the temporary security credentials that the operation returns.

Type: [AssumedRoleUser](#) (p. 27)

Audience

The value of the `Recipient` attribute of the `SubjectConfirmationData` element of the SAML assertion.

Type: String

Credentials

AWS credentials for API authentication.

Type: [Credentials](#) (p. 28)

Issuer

The value of the `Issuer` element of the SAML assertion.

Type: String

NameQualifier

A hash value based on the concatenation of the `Issuer` response value, the AWS account ID, and the friendly name (the last part of the ARN) of the SAML provider in IAM. The combination of `NameQualifier` and `Subject` can be used to uniquely identify a federated user.

The following pseudocode shows how the hash value is calculated:

```
BASE64 ( SHA1 ( "https://example.com/saml" + "123456789012" + "/MySAMLIdP" ) ) )
```

Type: String

PackedPolicySize

A percentage value that indicates the size of the policy in packed form. The service rejects any policy with a packed size greater than 100 percent, which means the policy exceeded the allowed space.

Type: Integer

Subject

The value of the `NameID` element in the `Subject` element of the SAML assertion.

Type: String

SubjectType

The format of the name ID, as defined by the `Format` attribute in the `NameID` element of the SAML assertion. Typical examples of the format are `transient` or `persistent`.

If the format includes the prefix `urn:oasis:names:tc:SAML:2.0:nameid-format`, that prefix is removed. For example, `urn:oasis:names:tc:SAML:2.0:nameid-format:transient` is returned as `transient`. If the format includes any other prefix, the format is returned with no modifications.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors](#) (p. 32).

ExpiredToken

The web identity token that was passed is expired or is not valid. Get a new identity token from the identity provider and then retry the request.

HTTP Status Code: 400

IDPRejectedClaim

The identity provider (IdP) reported that authentication failed. This might be because the claim is invalid.

If this error is returned for the `AssumeRoleWithWebIdentity` operation, it can also mean that the claim has expired or has been explicitly revoked.

HTTP Status Code: 403

InvalidIdentityToken

The web identity token that was passed could not be validated by AWS. Get a new identity token from the identity provider and then retry the request.

HTTP Status Code: 400

MalformedPolicyDocument

The request was rejected because the policy document was malformed. The error message describes the specific error.

HTTP Status Code: 400

PackedPolicyTooLarge

The request was rejected because the policy document was too large. The error message describes how big the policy document is, in packed form, as a percentage of what the API allows.

HTTP Status Code: 400

AssumeRoleWithWebIdentity

Returns a set of temporary security credentials for users who have been authenticated in a mobile or web application with a web identity provider, such as Amazon Cognito, Login with Amazon, Facebook, Google, or any OpenID Connect-compatible identity provider.

Note

For mobile applications, we recommend that you use Amazon Cognito. You can use Amazon Cognito with the AWS SDK for iOS and the AWS SDK for Android to uniquely identify a user and supply the user with a consistent identity throughout the lifetime of an application. To learn more about Amazon Cognito, see Amazon Cognito Overview in the AWS SDK for Android Developer Guide guide and Amazon Cognito Overview in the AWS SDK for iOS Developer Guide.

Calling `AssumeRoleWithWebIdentity` does not require the use of AWS security credentials. Therefore, you can distribute an application (for example, on mobile devices) that requests temporary security credentials without including long-term AWS credentials in the application, and without deploying server-based proxy services that use long-term AWS credentials. Instead, the identity of the caller is validated by using a token from the web identity provider.

The temporary security credentials returned by this API consist of an access key ID, a secret access key, and a security token. Applications can use these temporary security credentials to sign calls to AWS service APIs. The credentials are valid for the duration that you specified when calling `AssumeRoleWithWebIdentity`, which can be from 900 seconds (15 minutes) to 3600 seconds (1 hour). By default, the temporary security credentials are valid for 1 hour.

Optionally, you can pass an IAM access policy to this operation. If you choose not to pass a policy, the temporary security credentials that are returned by the operation have the permissions that are defined in the access policy of the role that is being assumed. If you pass a policy to this operation, the temporary security credentials that are returned by the operation have the permissions that are allowed by both the access policy of the role that is being assumed, *and* the policy that you pass. This gives you a way to further restrict the permissions for the resulting temporary security credentials. You cannot use the passed policy to grant permissions that are in excess of those allowed by the access policy of the role that is being assumed. For more information, see [Permissions for AssumeRoleWithWebIdentity](#).

Before your application can call `AssumeRoleWithWebIdentity`, you must have an identity token from a supported identity provider and create a role that the application can assume. The role that your application assumes must trust the identity provider that is associated with the identity token. In other words, the identity provider must be specified in the role's trust policy.

For more information about how to use web identity federation and the `AssumeRoleWithWebIdentity` API, see the following resources:

- [Creating a Mobile Application with Third-Party Sign-In](#) and [Creating Temporary Security Credentials for Mobile Apps Using Third-Party Identity Providers](#).
- [Web Identity Federation Playground](#). This interactive website lets you walk through the process of authenticating via Login with Amazon, Facebook, or Google, getting temporary security credentials, and then using those credentials to make a request to AWS.
- [AWS SDK for iOS](#) and [AWS SDK for Android](#). These toolkits contain sample apps that show how to invoke the identity providers, and then how to use the information from these providers to get and use temporary security credentials.
- [Web Identity Federation with Mobile Applications](#). This article discusses web identity federation and shows an example of how to use web identity federation to get access to content in Amazon S3.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 30\)](#).

DurationSeconds

The duration, in seconds, of the role session. The value can range from 900 seconds (15 minutes) to 3600 seconds (1 hour). By default, the value is set to 3600 seconds.

Type: Integer

Valid range: Minimum value of 900. Maximum value of 3600.

Required: No

Policy

An IAM policy in JSON format.

The policy parameter is optional. If you pass a policy, the temporary security credentials that are returned by the operation have the permissions that are allowed by both the access policy of the role that is being assumed, *and* the policy that you pass. This gives you a way to further restrict the permissions for the resulting temporary security credentials. You cannot use the passed policy to grant permissions that are in excess of those allowed by the access policy of the role that is being assumed. For more information, see [Permissions for AssumeRoleWithWebIdentity](#).

Note

The policy plain text must be 2048 bytes or shorter. However, an internal conversion compresses it into a packed binary format with a separate limit. The `PackedPolicySize` response element indicates by percentage how close to the upper size limit the policy is, with 100% equaling the maximum allowed size.

Type: String

Length constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF] +`

Required: No

ProviderId

The fully qualified host component of the domain name of the identity provider.

Specify this value only for OAuth 2.0 access tokens. Currently `www.amazon.com` and `graph.facebook.com` are the only supported identity providers for OAuth 2.0 access tokens. Do not include URL schemes and port numbers.

Do not specify this value for OpenID Connect ID tokens.

Type: String

Length constraints: Minimum length of 4. Maximum length of 2048.

Required: No

RoleArn

The Amazon Resource Name (ARN) of the role that the caller is assuming.

Type: String

Length constraints: Minimum length of 20. Maximum length of 2048.

Required: Yes

RoleSessionName

An identifier for the assumed role session. Typically, you pass the name or identifier that is associated with the user who is using your application. That way, the temporary security credentials that your application will use are associated with that user. This session name is included as part of the ARN and assumed role ID in the `AssumedRoleUser` response element.

Type: String

Length constraints: Minimum length of 2. Maximum length of 32.

Pattern: `[\w+=, .@-]*`

Required: Yes

WebIdentityToken

The OAuth 2.0 access token or OpenID Connect ID token that is provided by the identity provider. Your application must get this token by authenticating the user who is using your application with a web identity provider before the application makes an `AssumeRoleWithWebIdentity` call.

Type: String

Length constraints: Minimum length of 4. Maximum length of 2048.

Required: Yes

Response Elements

The following elements are returned.

AssumedRoleUser

The Amazon Resource Name (ARN) and the assumed role ID, which are identifiers that you can use to refer to the resulting temporary security credentials. For example, you can reference these credentials as a principal in a resource-based policy by using the ARN or assumed role ID. The ARN and ID include the `RoleSessionName` that you specified when you called `AssumeRole`.

Type: [AssumedRoleUser \(p. 27\)](#)

Audience

The intended audience (also known as client ID) of the web identity token. This is traditionally the client identifier issued to the application that requested the web identity token.

Type: String

Credentials

The temporary security credentials, which include an access key ID, a secret access key, and a security token.

Type: [Credentials \(p. 28\)](#)

PackedPolicySize

A percentage value that indicates the size of the policy in packed form. The service rejects any policy with a packed size greater than 100 percent, which means the policy exceeded the allowed space.

Type: Integer

Provider

The issuing authority of the web identity token presented. For OpenID Connect ID Tokens this contains the value of the `iss` field. For OAuth 2.0 access tokens, this contains the value of the `ProviderId` parameter that was passed in the `AssumeRoleWithWebIdentity` request.

Type: String

SubjectFromWebIdentityToken

The unique user identifier that is returned by the identity provider. This identifier is associated with the `WebIdentityToken` that was submitted with the `AssumeRoleWithWebIdentity` call. The identifier is typically unique to the user and the application that acquired the `WebIdentityToken` (pairwise identifier). For OpenID Connect ID tokens, this field contains the value returned by the identity provider as the token's `sub` (Subject) claim.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 32\)](#).

ExpiredToken

The web identity token that was passed is expired or is not valid. Get a new identity token from the identity provider and then retry the request.

HTTP Status Code: 400

IDPCommunicationError

The request could not be fulfilled because the non-AWS identity provider (IDP) that was asked to verify the incoming identity token could not be reached. This is often a transient error caused by network conditions. Retry the request a limited number of times so that you don't exceed the request rate. If the error persists, the non-AWS identity provider might be down or not responding.

HTTP Status Code: 400

IDPRejectedClaim

The identity provider (IdP) reported that authentication failed. This might be because the claim is invalid.

If this error is returned for the `AssumeRoleWithWebIdentity` operation, it can also mean that the claim has expired or has been explicitly revoked.

HTTP Status Code: 403

InvalidIdentityToken

The web identity token that was passed could not be validated by AWS. Get a new identity token from the identity provider and then retry the request.

HTTP Status Code: 400

MalformedPolicyDocument

The request was rejected because the policy document was malformed. The error message describes the specific error.

HTTP Status Code: 400

PackedPolicyTooLarge

The request was rejected because the policy document was too large. The error message describes how big the policy document is, in packed form, as a percentage of what the API allows.

HTTP Status Code: 400

Examples

Sample Request

```
https://sts.amazonaws.com/  
?Action=AssumeRoleWithWebIdentity  
&DurationSeconds=3600  
&ProviderId=www.amazon.com  
&RoleSessionName=appl  
&RoleArn=arn:aws:iam::123456789012:role/FederatedWebIdentityRole  
&WebIdentityToken=Atza%7CIEBLjAsAhRFiXuWpUXuRvQ9PZL3GMFcYevydwIUFAHZwXZXX  
XXXXXXJnrulxKDHwy87oGKPznh0D6bEQZTSCzyoCtL_8S07pLpr0zMbn6w1lfVZKNTBdDansFB  
mtGnIsIapji6xKR02Yc_2bQ8LZbUXSGm6Ry6_BG7PrtLZtj_dfCTj92xNGed-CrKqjG7nPBjNI  
L016GGvuS5gSvPRUxWES3VYfmlw17WTI7jn-Pcb6M-buCgHhFOzTQxod27L9CqnOLio7N3gZAG  
psp6nl-AJBOCJckcyXe2c6uD0srOJeZlKUm2eTDVMf8IehDVI0r1QOnTV6KzzAI3OY87Vd_cVMQ  
&Version=2011-06-15
```

Sample Response

```
<AssumeRoleWithWebIdentityResponse xmlns="https://sts.amazonaws.com/doc/2011-  
06-15/">  
  <AssumeRoleWithWebIdentityResult>  
    <SubjectFromWebIdentityToken>amzn1.account.AF6RH07KZU5XRVQJGXK6HB56KR2A</Sub  
jectFromWebIdentityToken>  
    <Audience>client.5498841531868486423.1548@apps.example.com</Audience>  
    <AssumedRoleUser>  
      <Arn>arn:aws:sts::123456789012:assumed-role/FederatedWebIdentityR  
ole/appl</Arn>  
      <AssumedRoleId>AROACLKWSQRAOEXAMPLE:appl</AssumedRoleId>  
    </AssumedRoleUser>  
    <Credentials>  
      <SessionToken>AQoDYXdzEE0a8ANXXXXXXXXXN0lewxE5TijQyp+IEXAMPLE</SessionToken>  
  
      <SecretAccessKey>wJalrXUtnFEMI/K7MDENG/bPxrFiCYzEXAMPLEKEY</SecretAccess  
Key>  
      <Expiration>2014-10-24T23:00:23Z</Expiration>  
      <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>  
    </Credentials>  
    <Provider>www.amazon.com</Provider>  
  </AssumeRoleWithWebIdentityResult>  
  <ResponseMetadata>  
    <RequestId>ad4156e9-bce1-11e2-82e6-6b6efEXAMPLE</RequestId>  
  </ResponseMetadata>  
</AssumeRoleWithWebIdentityResponse>
```

DecodeAuthorizationMessage

Decodes additional information about the authorization status of a request from an encoded message returned in response to an AWS request.

For example, if a user is not authorized to perform an action that he or she has requested, the request returns a `Client.UnauthorizedOperation` response (an HTTP 403 response). Some AWS actions additionally return an encoded message that can provide details about this authorization failure.

Note

Only certain AWS actions return an encoded authorization message. The documentation for an individual action indicates whether that action returns an encoded message in addition to returning an HTTP code.

The message is encoded because the details of the authorization status can constitute privileged information that the user who requested the action should not see. To decode an authorization status message, a user must be granted permissions via an IAM policy to request the `DecodeAuthorizationMessage` (`sts:DecodeAuthorizationMessage`) action.

The decoded message includes the following type of information:

- Whether the request was denied due to an explicit deny or due to the absence of an explicit allow. For more information, see [Determining Whether a Request is Allowed or Denied](#) in *Using IAM*.
- The principal who made the request.
- The requested action.
- The requested resource.
- The values of condition keys in the context of the user's request.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters](#) (p. 30).

EncodedMessage

The encoded message that was returned with the response.

Type: String

Length constraints: Minimum length of 1. Maximum length of 10240.

Required: Yes

Response Elements

The following element is returned.

DecodedMessage

An XML document that contains the decoded message. For more information, see `DecodeAuthorizationMessage`.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 32\)](#).

InvalidAuthorizationMessage

The error returned if the message passed to `DecodeAuthorizationMessage` was invalid. This can happen if the token contains invalid characters, such as linebreaks.

HTTP Status Code: 400

Examples

Sample Request

```
POST https://sts.amazonaws.com / HTTP/1.1
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Host: sts.amazonaws.com
Content-Length: 1148
Expect: 100-continue
Connection: Keep-Alive
Action=DecodeAuthorizationMessage
&EncodedMessage=<encoded-message>
&Version=2011-06-15
&AUTHPARAMS
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<DecodeAuthorizationMessageResponse xmlns="http://sts.amazonaws.com/doc/2011-06-15/">
  <requestId>6624a9ca-cd25-4f50-b2a5-7ba65bf07453</requestId>
  <DecodedMessage>
    {
      "allowed": "false",
      "explicitDeny": "false",
      "matchedStatements": "",
      "failures": "",
      "context": {
        "principal": {
          "id": "AIDACKCEVSQ6C2EXAMPLE",
          "name": "Bob",
          "arn": "arn:aws:iam::123456789012:user/Bob"
        },
        "action": "ec2:StopInstances",
        "resource": "arn:aws:ec2:us-east-1:123456789012:instance/i-dd01c9bd",
        "conditions": [
          {
            "item": {
              "key": "ec2:Tenancy",
              "values": ["default"]
            },
            {
              "item": {
```

AWS Security Token Service API Reference Examples

```
        "key": "ec2:ResourceTag/elasticbeanstalk:environment-name",
        "values": ["Default-Environment"]
    }
},
(Additional items ...)
]
}
</DecodedMessage>
</DecodeAuthorizationMessageResponse>
```

GetFederationToken

Returns a set of temporary security credentials (consisting of an access key ID, a secret access key, and a security token) for a federated user. A typical use is in a proxy application that gets temporary security credentials on behalf of distributed applications inside a corporate network. Because you must call the `GetFederationToken` action using the long-term security credentials of an IAM user, this call is appropriate in contexts where those credentials can be safely stored, usually in a server-based application.

Note

If you are creating a mobile-based or browser-based app that can authenticate users using a web identity provider like Login with Amazon, Facebook, Google, or an OpenID Connect-compatible identity provider, we recommend that you use Amazon Cognito or `AssumeRoleWithWebIdentity`. For more information, see [Creating Temporary Security Credentials for Mobile Apps Using Identity Providers](#).

The `GetFederationToken` action must be called by using the long-term AWS security credentials of an IAM user. You can also call `GetFederationToken` using the security credentials of an AWS account (root), but this is not recommended. Instead, we recommend that you create an IAM user for the purpose of the proxy application and then attach a policy to the IAM user that limits federated users to only the actions and resources they need access to. For more information, see [IAM Best Practices in Using IAM](#).

The temporary security credentials that are obtained by using the long-term credentials of an IAM user are valid for the specified duration, between 900 seconds (15 minutes) and 129600 seconds (36 hours). Temporary credentials that are obtained by using AWS account (root) credentials have a maximum duration of 3600 seconds (1 hour)

Permissions

The permissions for the temporary security credentials returned by `GetFederationToken` are determined by a combination of the following:

- The policy or policies that are attached to the IAM user whose credentials are used to call `GetFederationToken`.
- The policy that is passed as a parameter in the call.

The passed policy is attached to the temporary security credentials that result from the `GetFederationToken` API call--that is, to the *federated user*. When the federated user makes an AWS request, AWS evaluates the policy attached to the federated user in combination with the policy or policies attached to the IAM user whose credentials were used to call `GetFederationToken`. AWS allows the federated user's request only when both the federated user *and* the IAM user are explicitly allowed to perform the requested action. The passed policy cannot grant more permissions than those that are defined in the IAM user policy.

A typical use case is that the permissions of the IAM user whose credentials are used to call `GetFederationToken` are designed to allow access to all the actions and resources that any federated user will need. Then, for individual users, you pass a policy to the operation that scopes down the permissions to a level that's appropriate to that individual user, using a policy that allows only a subset of permissions that are granted to the IAM user.

If you do not pass a policy, the resulting temporary security credentials have no effective permissions. The only exception is when the temporary security credentials are used to access a resource that has a resource-based policy that specifically allows the federated user to access the resource.

For more information about how permissions work, see [Permissions for GetFederationToken](#). For information about using `GetFederationToken` to create temporary security credentials, see [Creating Temporary Credentials to Enable Access for Federated Users](#).

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 30\)](#).

DurationSeconds

The duration, in seconds, that the session should last. Acceptable durations for federation sessions range from 900 seconds (15 minutes) to 129600 seconds (36 hours), with 43200 seconds (12 hours) as the default. Sessions obtained using AWS account (root) credentials are restricted to a maximum of 3600 seconds (one hour). If the specified duration is longer than one hour, the session obtained by using AWS account (root) credentials defaults to one hour.

Type: Integer

Valid range: Minimum value of 900. Maximum value of 129600.

Required: No

Name

The name of the federated user. The name is used as an identifier for the temporary security credentials (such as `Bob`). For example, you can reference the federated user name in a resource-based policy, such as in an Amazon S3 bucket policy.

Type: String

Length constraints: Minimum length of 2. Maximum length of 32.

Pattern: `[\w+=, .@-]*`

Required: Yes

Policy

An IAM policy in JSON format that is passed with the `GetFederationToken` call and evaluated along with the policy or policies that are attached to the IAM user whose credentials are used to call `GetFederationToken`. The passed policy is used to scope down the permissions that are available to the IAM user, by allowing only a subset of the permissions that are granted to the IAM user. The passed policy cannot grant more permissions than those granted to the IAM user. The final permissions for the federated user are the most restrictive set based on the intersection of the passed policy and the IAM user policy.

If you do not pass a policy, the resulting temporary security credentials have no effective permissions. The only exception is when the temporary security credentials are used to access a resource that has a resource-based policy that specifically allows the federated user to access the resource.

Note

The policy plain text must be 2048 bytes or shorter. However, an internal conversion compresses it into a packed binary format with a separate limit. The `PackedPolicySize` response element indicates by percentage how close to the upper size limit the policy is, with 100% equaling the maximum allowed size.

For more information about how permissions work, see [Permissions for GetFederationToken](#).

Type: String

Length constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF]+`

Required: No

Response Elements

The following elements are returned.

Credentials

Credentials for the service API authentication.

Type: [Credentials \(p. 28\)](#)

FederatedUser

Identifiers for the federated user associated with the credentials (such as `arn:aws:sts::123456789012:federated-user/Bob` or `123456789012:Bob`). You can use the federated user's ARN in your resource-based policies, such as an Amazon S3 bucket policy.

Type: [FederatedUser \(p. 28\)](#)

PackedPolicySize

A percentage value indicating the size of the policy in packed form. The service rejects policies for which the packed size is greater than 100 percent of the allowed value.

Type: Integer

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 32\)](#).

MalformedPolicyDocument

The request was rejected because the policy document was malformed. The error message describes the specific error.

HTTP Status Code: 400

PackedPolicyTooLarge

The request was rejected because the policy document was too large. The error message describes how big the policy document is, in packed form, as a percentage of what the API allows.

HTTP Status Code: 400

Examples

Sample Request

```
https://sts.amazonaws.com/  
?Version=2011-06-15  
&Action=GetFederationToken  
&Name=Bob  
&Policy=%7B%22Version%22%3A%222012-10-17%22%2C%22State  
ment%22%3A%5B%7B%22Sid%22%3A%22Stmt1%22%2C%22Effect%22%  
3A%22Allow%22%2C%22Action%22%3A%22s3%3A%22%2C%22Resource%22%3A%22%7D  
%5D%7D  
&DurationSeconds=3600  
&AUTHPARAMS
```

Sample Response

```
<GetFederationTokenResponse xmlns="https://sts.amazonaws.com/doc/
2011-06-15/">
  <GetFederationTokenResult>
    <Credentials>
      <SessionToken>
        AQoDYXdzEPT//////////wEXAMPLEetc764bNrC9SAPBSM22wD0k4x4HIZ8j4FZTWdQW
        LWSKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
        QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
        9HFv1Rd8Tx6q6fE8YQcHNvXAKiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
        +scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iS1lTJabIQwj2ICCR/oLxBA==
      </SessionToken>
      <SecretAccessKey>
        wJalrXUtnFEMI/K7MDENG/bPxrFiCYzEXAMPLEKEY
      </SecretAccessKey>
      <Expiration>2011-07-15T23:28:33.359Z</Expiration>
      <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
    </Credentials>
    <FederatedUser>
      <Arn>arn:aws:sts::123456789012:federated-user/Bob</Arn>
      <FederatedUserId>123456789012:Bob</FederatedUserId>
    </FederatedUser>
    <PackedPolicySize>6</PackedPolicySize>
  </GetFederationTokenResult>
  <ResponseMetadata>
    <RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
  </ResponseMetadata>
</GetFederationTokenResponse>
```

GetSessionToken

Returns a set of temporary credentials for an AWS account or IAM user. The credentials consist of an access key ID, a secret access key, and a security token. Typically, you use `GetSessionToken` if you want to use MFA to protect programmatic calls to specific AWS APIs like Amazon EC2 `StopInstances`. MFA-enabled IAM users would need to call `GetSessionToken` and submit an MFA code that is associated with their MFA device. Using the temporary security credentials that are returned from the call, IAM users can then make programmatic calls to APIs that require MFA authentication.

The `GetSessionToken` action must be called by using the long-term AWS security credentials of the AWS account or an IAM user. Credentials that are created by IAM users are valid for the duration that you specify, between 900 seconds (15 minutes) and 129600 seconds (36 hours); credentials that are created by using account credentials have a maximum duration of 3600 seconds (1 hour).

Note

We recommend that you do not call `GetSessionToken` with root account credentials. Instead, follow our best practices by creating one or more IAM users, giving them the necessary permissions, and using IAM users for everyday interaction with AWS.

The permissions associated with the temporary security credentials returned by `GetSessionToken` are based on the permissions associated with account or IAM user whose credentials are used to call the action. If `GetSessionToken` is called using root account credentials, the temporary credentials have root account permissions. Similarly, if `GetSessionToken` is called using the credentials of an IAM user, the temporary credentials have the same permissions as the IAM user.

For more information about using `GetSessionToken` to create temporary credentials, go to [Creating Temporary Credentials to Enable Access for IAM Users](#).

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 30\)](#).

DurationSeconds

The duration, in seconds, that the credentials should remain valid. Acceptable durations for IAM user sessions range from 900 seconds (15 minutes) to 129600 seconds (36 hours), with 43200 seconds (12 hours) as the default. Sessions for AWS account owners are restricted to a maximum of 3600 seconds (one hour). If the duration is longer than one hour, the session for AWS account owners defaults to one hour.

Type: Integer

Valid range: Minimum value of 900. Maximum value of 129600.

Required: No

SerialNumber

The identification number of the MFA device that is associated with the IAM user who is making the `GetSessionToken` call. Specify this value if the IAM user has a policy that requires MFA authentication. The value is either the serial number for a hardware device (such as `GAHT12345678`) or an Amazon Resource Name (ARN) for a virtual device (such as `arn:aws:iam::123456789012:mfa/user`). You can find the device for an IAM user by going to the AWS Management Console and viewing the user's security credentials.

Type: String

Length constraints: Minimum length of 9. Maximum length of 256.

Pattern: `[\w+=/:\.,@-]*`

Required: No

TokenCode

The value provided by the MFA device, if MFA is required. If any policy requires the IAM user to submit an MFA code, specify this value. If MFA authentication is required, and the user does not provide a code when requesting a set of temporary security credentials, the user will receive an "access denied" response when requesting resources that require MFA authentication.

Type: String

Length constraints: Minimum length of 6. Maximum length of 6.

Pattern: [\d]*

Required: No

Response Elements

The following element is returned.

Credentials

The session credentials for API authentication.

Type: [Credentials \(p. 28\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 32\)](#).

Examples

Sample Request

```
https://sts.amazonaws.com/  
?Version=2011-06-15  
&Action=GetSessionToken  
&DurationSeconds=3600  
&SerialNumber=YourMFADeviceSerialNumber  
&TokenCode=123456  
&AUTHPARAMS
```

Sample Response

```
<GetSessionTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">  
  <GetSessionTokenResult>  
    <Credentials>  
      <SessionToken>  
        AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrRh3c/L  
        To6UdDyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/IvU1dYUg2RVAJBanLiHb4IgRmpRV3z  
        rkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/AXlzBBko7b15fjrBs2+cTQtp  
        Z3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE
```

AWS Security Token Service API Reference Examples

```
</SessionToken>
<SecretAccessKey>
wJalrXUtnFEMI/K7MDENG/bPxrFiCYzEXAMPLEKEY
</SecretAccessKey>
<Expiration>2011-07-11T19:55:29.611Z</Expiration>
<AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
</Credentials>
</GetSessionTokenResult>
<ResponseMetadata>
  <RequestId>58c5dbae-abef-11e0-8cfe-09039844ac7d</RequestId>
</ResponseMetadata>
</GetSessionTokenResponse>
```

Data Types

The AWS Security Token Service API contains several data types that various actions use. This section describes each data type in detail.

Note

The order of each element in the response is not guaranteed. Applications should not assume a particular order.

The following data types are supported:

- [AssumedRoleUser](#) (p. 27)
- [Credentials](#) (p. 28)
- [FederatedUser](#) (p. 28)

AssumedRoleUser

Description

The identifiers for the temporary security credentials that the operation returns.

Contents

Arn

The ARN of the temporary security credentials that are returned from the [AssumeRole](#) (p. 3) action. For more information about ARNs and how to use them in policies, see [IAM Identifiers](#) in *Using IAM*.

Type: String

Length constraints: Minimum length of 20. Maximum length of 2048.

Required: Yes

AssumedRoleId

A unique identifier that contains the role ID and the role session name of the role that is being assumed. The role ID is generated by AWS when the role is created.

Type: String

Length constraints: Minimum length of 2. Maximum length of 96.

Pattern: [\w+=, .@: -] *

Required: Yes

Credentials

Description

AWS credentials for API authentication.

Contents

AccessKeyId

The access key ID that identifies the temporary security credentials.

Type: String

Length constraints: Minimum length of 16. Maximum length of 32.

Pattern: [\w] *

Required: Yes

Expiration

The date on which the current credentials expire.

Type: DateTime

Required: Yes

SecretAccessKey

The secret access key that can be used to sign requests.

Type: String

Required: Yes

SessionToken

The token that users must pass to the service API to use the temporary credentials.

Type: String

Required: Yes

FederatedUser

Description

Identifiers for the federated user that is associated with the credentials.

Contents

Arn

The ARN that specifies the federated user that is associated with the credentials. For more information about ARNs and how to use them in policies, see [IAM Identifiers](#) in *Using IAM*.

Type: String

Length constraints: Minimum length of 20. Maximum length of 2048.

Required: Yes

FederatedUserId

The string that identifies the federated user associated with the credentials, similar to the unique ID of an IAM user.

Type: String

Length constraints: Minimum length of 2. Maximum length of 96.

Pattern: [\w+=, .@\:-] *

Required: Yes

Common Parameters

This section lists the request parameters that all actions use. Any action-specific parameters are listed in the topic for the action.

Action

The action to be performed.

Default: None

Type: string

Required: Yes

AuthParams

The parameters that are required to authenticate a Conditional request. Contains:

- `AWSAccessKeyID`
- `SignatureVersion`
- `Timestamp`
- `Signature`

Default: None

Required: Conditional

AWSAccessKeyID

The access key ID that corresponds to the secret access key that you used to sign the request.

Default: None

Type: string

Required: Yes

Expires

The date and time when the request signature expires, expressed in the format `YYYY-MM-DDThh:mm:ssZ`, as specified in the ISO 8601 standard.

Condition: Requests must include either *Timestamp* or *Expires*, but not both.

Default: None

Type: string

Required: Conditional

SecurityToken

The temporary security token that was obtained through a call to AWS Security Token Service. For a list of services that support AWS Security Token Service, go to [Using Temporary Security Credentials to Access AWS](#) in **Using Temporary Security Credentials**.

Default: None

Type: string

Required: No

Signature

The digital signature that you created for the request. For information about generating a signature, go to the service's developer documentation.

Default: None

Type: string

Required: Yes

SignatureMethod

The hash algorithm that you used to create the request signature.

Default: None

Type: string

Valid Values: HmacSHA256 | HmacSHA1

Required: Yes

SignatureVersion

The signature version you use to sign the request. Set this to the value that is recommended for your service.

Default: None

Type: string

Required: Yes

Timestamp

The date and time when the request was signed, expressed in the format YYYY-MM-DDThh:mm:ssZ, as specified in the ISO 8601 standard.

Condition: Requests must include either *Timestamp* or *Expires*, but not both.

Default: None

Type: string

Required: Conditional

Version

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Default: None

Type: string

Required: Yes

Common Errors

This section lists the common errors that all actions return. Any action-specific errors are listed in the topic for the action.

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

InvalidParameterCombination

Parameters that must not be used together were used together.

HTTP Status Code: 400

InvalidParameterValue

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

InvalidQueryStringParameter

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

MalformedQueryString

The query string contains a syntax error.

HTTP Status Code: 404

MissingAction

The request is missing an action or a required parameter.

HTTP Status Code: 400

MissingAuthenticationToken

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

MissingParameter

A required parameter for the specified action is not supplied.

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

Throttling

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