**Requesting Temporary Security Credentials**

## [AssumeRole](https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html)—Cross-Account Delegation and Federation Through a Custom Identity Broker

The AssumeRole API operation is useful for allowing existing IAM users to access AWS resources that they don't already have access to, such as resources in another AWS account. It is also useful as a means to temporarily gain privileged access—for example, to provide multi-factor authentication (MFA). You must call this API using existing IAM user credentials..

## [AssumeRoleWithWebIdentity](https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRoleWithWebIdentity.html)—Federation Through a Web-Based Identity Provider

The AssumeRoleWithWebIdentity API operation returns a set of temporary security credentials for federated users who are authenticated through a public identity provider. Examples of public identity providers include Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC)-compatible identity provider. This operation is useful for creating mobile applications or client-based web applications that require access to AWS. Using this operation means that your users do not need their own AWS or IAM identities. For more information, see [About Web Identity Federation](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_oidc.html).

**Note**

Instead of directly calling AssumeRoleWithWebIdentity, we recommend that you use Amazon Cognito and the Amazon Cognito credentials provider with the AWS SDKs for mobile development.

## [AssumeRoleWithSAML](https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRoleWithSAML.html)—Federation Through an Enterprise Identity Provider Compatible with SAML 2.0

The AssumeRoleWithSAML API operation returns a set of temporary security credentials for federated users who are authenticated by your organization's existing identity system. The users must also use [SAML](https://www.oasis-open.org/standards#samlv2.0) 2.0 (Security Assertion Markup Language) to pass authentication and authorization information to AWS. This API operation is useful in organizations that have integrated their identity systems (such as Windows Active Directory or OpenLDAP) with software that can produce SAML assertions. Such an integration provides information about user identity and permissions (such as Active Directory Federation Services or Shibboleth).

e old ones expire.

## [GetFederationToken](https://docs.aws.amazon.com/STS/latest/APIReference/API_GetFederationToken.html)—Federation Through a Custom Identity Broker

The GetFederationToken API operation returns a set of temporary security credentials for federated users. This API differs from AssumeRole in that the default expiration period is substantially longer (12 hours instead of one hour). Additionally, you can use the DurationSeconds parameter to specify a duration for the temporary security credentials to remain valid. The resulting credentials are valid for the specified duration, between 900 seconds (15 minutes) to 129,600 seconds (36 hours).The longer expiration period can help reduce the number of calls to AWS because you do not need to get new credentials as often.

When you make a request to get temporary security credentials for a federated user, you use the credentials of a specific user identity (an IAM user) to make the request. The permissions for the temporary security credentials are determined by the session policies that you pass when you call GetFederationToken. The resulting session permissions are the intersection of the IAM user policies and the session policies that you pass. Session policies cannot be used to grant more permissions than those allowed by the identity-based policy of the IAM user that is requesting federation. For more information about role session permissions, see [Session Policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html#policies_session).

The GetFederationToken call returns temporary security credentials that consist of the security token, access key, secret key, and expiration. You can use GetFederationToken if you want to manage permissions inside your organization (for example, using the proxy application to assign permissions).

## [GetSessionToken](https://docs.aws.amazon.com/STS/latest/APIReference/API_GetSessionToken.html)—Temporary Credentials for Users in Untrusted Environments

The GetSessionToken API operation returns a set of temporary security credentials to an existing IAM user. This is useful for providing enhanced security, such as allowing AWS requests only when MFA is enabled for the IAM user. Because the credentials are temporary, they provide enhanced security when you have an IAM user who accesses your resources through a less secure environment. Examples of less secure environments include a mobile device or web browser.

## AssumeRole vs GetFederationToken

GetFederationToken takes long-lived ("you must call the GetFederationToken action using the long-term security credentials of an IAM user") and high-powered (even root) AWS credentials and scopes them down in power (weaker) and time (shorter: up to 36h) so they can be handed out to someone else.

On the opposite end of the spectrum, AssumeRoleWithWebIdentity requires no AWS credentials to start with; instead, it takes an external identity and uses the trust policy built into the role to elevate AWS access for a short time (max 1h).

In the middle of those two, AssumeRole does require you to start with some basic AWS credentials, but those are allowed to be short-term ones as opposed to GetFederationToken. So, for example, you could call AssumeRole based on the role associated with an EC2 instance so your identity broker doesn't need to manage IAM user credentials.

As far as I understand, GetFederationToken was the way you used to have to do things--before roles came along. Now that we have roles, though, they generally offer a much better (including more secure) approach that should be preferred over the old long-lived credentials approaches. If you want to update an existing identity broker, though, which role API you should use will depend on the situation/integrations in question.

You've probably already seen this, but [here's a link to AWS's comparison between all the STS APIs](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp_request.html#stsapi_comparison).

OR

Quoting it, it says that both return a set of temporary security credentials, but they differ in how permissions associated with the temporary security credentials are derived.

Calling GetFederationToken requires an IAM user or root. The resulting permissions inherit the permissions of the caller, scoped down by the permissions attached in the request (if you don’t attach permissions, the resulting permissions will be deny \*). The actual permissions associated with the session are the intersection of the caller’s base permissions and the permissions attached as an API parameter. If you use an IAM user to call GetFederationToken, you’ll want this user to have the minimum permissions necessary to cover every federated user’s use case, since the actual delegated permissions are always a subset of the caller’s privileges.

Unlike GetFederationToken, AssumeRole sessions derive their permissions from the role policies that you’ve pre-defined, scoped down by the optional permissions attached in the request. The actual permissions associated with the session are the intersection of the role’s permissions and the permissions attached as an API parameter. Only an IAM user or another role with permissions to call AssumeRole can assume a role. By default, you can define up to 250 roles (use the IAM Limit Increase form to request more), and your users/applications can assume any role you allow them to assume. As a result, you can create role permission profiles tailored for individual use cases and don’t have to pass in a policy with your AssumeRole requests (though you can if you wish).

It’s important to understand that using either GetFederationToken or AssumeRole, the permissions are evaluated each time an AWS API call is made. That means that even though you cannot revoke the session, you can always modify the permissions associated with a session even after the session has been issued. Simply modify the permissions on the IAM user (for GetFederationToken) or IAM role (for AssumeRole), and the permissions on the session will automatically be affected as well.