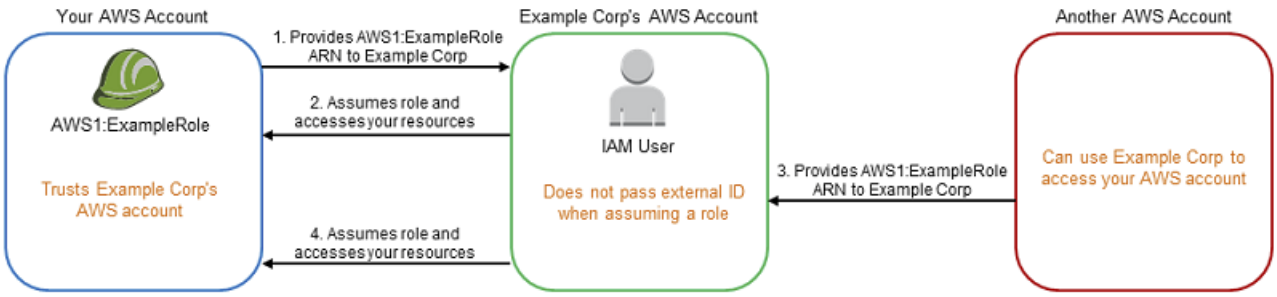
The confused deputy problem

PDF (iam-ug.pdf#confused-deputy)
Kindle (https://www.amazon.com/dp/B07642VLTV)
RSS (aws-iam-release-notes.rss)

To continue the previous example (./id_roles_create_for-user_externalid.html) , Example Corp requires access to certain resources in your AWS account. But in addition to you, Example Corp has other customers and needs a way to access each customer's AWS resources. Instead of asking its customers for their AWS account access keys, which are secrets that should never be shared, Example Corp requests a role ARN from each customer. But another Example Corp customer might be able to guess or obtain your role ARN. That customer could then use your role ARN to gain access to your AWS resources by way of Example Corp. This form of permission escalation is known as the confused deputy problem.

The following diagram illustrates the confused deputy problem.



This scenario assumes the following:

- **AWS1** is your AWS account.
- **AWS1:ExampleRole** is a role in your account. This role's trust policy trusts Example Corp by specifying Example Corp's AWS account as the one that can assume the role.

Here's what happens:

1. When you start using Example Corp's service, you provide the ARN of **AWS1:ExampleRole** to Example Corp.
2. Example Corp uses that role ARN to obtain temporary security credentials to access resources in your AWS account. In this way, you are trusting Example Corp as a "deputy" that can act on your behalf.
3. Another AWS customer also starts using Example Corp's service, and this customer also provides the ARN of **AWS1:ExampleRole** for Example Corp to use. Presumably the other customer learned or guessed the **AWS1:ExampleRole**, which isn't a secret.
4. When the other customer asks Example Corp to access AWS resources in (what it claims to be) its account, Example Corp uses **AWS1:ExampleRole** to access resources in your account.

This is how the other customer could gain unauthorized access to your resources. Because this other customer was able to trick Example Corp into unwittingly acting on your resources, Example Corp is now a "confused deputy."

How does an external ID prevent the confused deputy problem?

You address the confused deputy problem by including the `ExternalId` condition check in the role's trust policy. The "deputy" company inserts a unique external ID value for each customer into the request for AWS credentials. The external ID is a customer ID value that must be unique among Example Corp's customers and is out of the control of Example Corp's customers. This is why you get it from Example Corp and you don't come up with it on your own. This helps prevent one customer from successfully impersonating another customer. Example Corp always inserts the customer's assigned external ID, so you should never see a request coming from Example Corp with any external ID except your own.

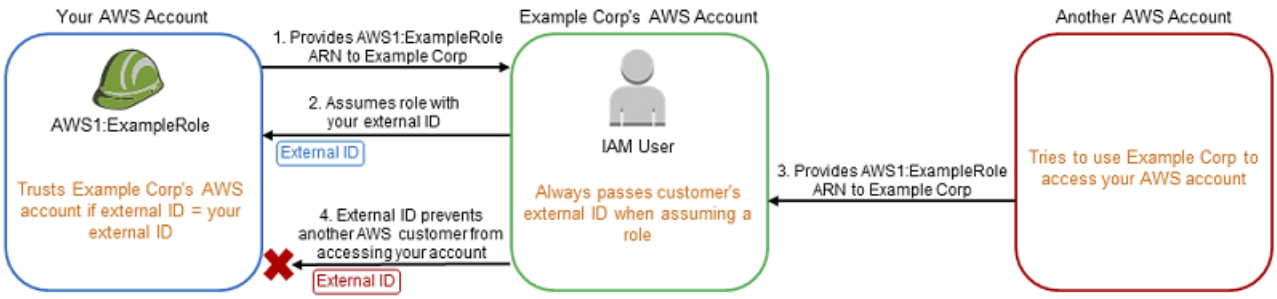
In our scenario, imagine Example Corp's unique identifier for you is "12345," and its identifier for the other customer is "67890." These identifiers are simplified for this scenario. Generally, these identifiers are GUIDs. Assuming that these identifiers are unique among Example Corp's customers, they are sensible values to use for the external ID.

Example Corp gives the external ID value of "12345" to you. You must then add a `Condition` element to the role's trust policy that requires the `sts:ExternalId` value to be 12345, like this:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Principal": {"AWS": "Example Corp's AWS Account ID"},
    "Condition": {"StringEquals": {"sts:ExternalId": "12345"}}
  }
}
```

The Condition element in this policy allows Example Corp to assume the role only when the AssumeRole API call includes the external ID value of "12345". Example Corp makes sure that whenever it assumes a role on behalf of a customer, it always includes that customer's external ID value in the AssumeRole call. Even if another customer supplies Example Corp with your ARN, it cannot control the external ID that Example Corp includes in its request to AWS. This helps prevent an unauthorized customer from gaining access to your resources.

The following diagram illustrates this.



1. As before, when you start using Example Corp's service, you provide the ARN of **AWS1:ExampleRole** to Example Corp.
2. When Example Corp uses that role ARN to assume the role **AWS1:ExampleRole**, Example Corp includes your external ID ("12345") in the AssumeRole API call. The external ID matches the role's trust policy, so the AssumeRole API call succeeds and Example Corp obtains temporary security credentials to access resources in your AWS account.
3. Another AWS customer also starts using Example Corp's service, and as before, this customer also provides the ARN of **AWS1:ExampleRole** for Example Corp to use.
4. But this time, when Example Corp attempts to assume the role **AWS1:ExampleRole**, it provides the external ID associated with the other customer ("67890"). The other customer has no way to change this. Example Corp does this because the request to use the role came from the other customer, so "67890" indicates the circumstance in which Example Corp is acting. Because you added a condition with your own external ID ("12345") to the trust policy of **AWS1:ExampleRole**, the AssumeRole API call fails. The other customer is prevented from gaining unauthorized access to resources in your account (indicated by the red "X" in the diagram).

The external ID helps prevent any other customer from tricking Example Corp into unwittingly accessing your resources–it mitigates the confused deputy problem.