The Application Permission Model

Updated on Tue, 2011-07-12 10:16

Table of Contents

A few updates about the permission model change

Sent Jun 13, 2011

Hey everyone,

A number of updates were made to the Direct Message methods and OAuth screens at the end of last week. Here's what went out:

- force_login is now supported on https://api.twitter.com/oauth/authorize
- the OAuth screens now support a feature phone tier of handsets and render them in a simpler format
- the language on all the screens is standardized to say "direct message"
- there is a "Return to App" URL on the Deny and Cancel screens that redirects the user to the oauth_callback url with a 'denied' parameter instead of oauth_token.

This next parameter isn't needed by everybody but we will be adding screen_name support to the authorize and authenticate pages in the next few days. If you want to add this to your code ready for when we release the feature you can, but please know the screen_name parameter will be ignored unless you also provide the force_login parameter. The screen_name parameter pre-fills the username field of the OAuth screen when force_login is true. The user is still able to edit the field, even if it is prefilled.

Lastly, these are the main points discussed in previous emails and Tweets:

- The new permission level will be enforced on 30th June.
- If you don't need to read or delete direct messages you do not need to update the permission level of your application.
- Read/Write applications will still be able to send direct messages, even after the enforcement date.
- Existing oauth_tokens will not be invalidated, even if the application permission level is altered.
- You can find out the current permission level of an oauth_token by inspecting the headers of an authenticated request to the API. Look for the X-Access-Level header.

The new permission level announcement email

Sent May 18, 2011

Hey everyone,

We recently updated our OAuth screens to give users greater transparency about the level of access applications have to their accounts. The valuable feedback Twitter users and developers have given us played a large part in that redesign and helped us identify where we can do more.

In particular, users and developers have requested greater granularity for permission levels.

In response to this feedback, we have created a new permission level for applications called "Read, Write & Direct Messages". This permission will allow an application to read or delete a user's direct messages. When we enforce this permission, applications without a "Read, Write & Direct Messages" token will be unable to read or delete direct

Authentication & Authorization

Tags

- Direct Messages (11)
- Auth (19)
- permissions (9)

messages. To ensure users know that an application is receiving access to their direct messages, we are also restricting this permission to the OAuth /authorize web flow only. This means applications which use xAuth and want to access direct messages must send a user through the full OAuth flow.

What does this mean for your application?

If you do not need access to direct messages: you won't need to make any changes to your application. When we enforce the new permission level your read or read/write token will automatically lose access to direct messages.

If you do need access to direct messages: you will need to edit your application record on https://dev.twitter.com/apps and change the permission level of your application to "Read, Write and Direct Messages". The new permission will not affect existing tokens which means existing users or your app or service will need to reauthorize.

We know this will take some time so we are allowing a transition period until the end of this month. During this time there will be no change to the access Read/Write tokens have to a users account. However, at the end of the month any tokens which have not been upgrade to "Read, Write and Direct Messages" will be unable to access and delete direct messages.

Affected APIs and requests

On the REST API, Read and Read/Write applications will no longer be able to use these API methods:

- /1/direct_messages.{format}
- /1/direct_messages/sent.{format}
- /1/direct_messages/show.{format}
- /1/direct_messages/destroy.{format}

For the Streaming API, both User Streams and Site Streams will only receive direct messages if the user has authorised an application to access direct messages.

Applications that use "Sign-in with Twitter" or xAuth will only be able to receive Read or Read/Write tokens.

What this means is only applications which direct a user through the OAuth web flow will be able to receive access tokens that allow access to direct messages. Any other method of authorization, including xAuth, will only be able to receive Read/Write tokens.

What will happen when the permission is activated

When we activate the new permission, all Read and Read/Write user_tokens issued to third-party applications will lose their ability to read direct messages. Any attempt to read direct messages will result in an HTTP 403 error being returned.

For example, a GET request to `https://api.twitter.com/1/direct_messages/sent.json` will return an HTTP 403 Forbidden with the response body:

```
{"errors":[{"code":93,"message":"This application is not allowed to access or delete your direct messages"}]}
```

Key Points

- If you wish to access a user's direct messages you will need to update your application and reauthorize existing tokens.
- The only way to get direct message access is to request access through the OAuth /authorize web flow. You will not be permitted to access direct messages if you use xAuth.
- When we enforce the permission Read/Write and Read tokens will be unable to access and delete direct messages.
- Read/Write tokens will be able to send direct messages after the permission is enforced.

We have also blogged about this on the Twitter blog:http://blog.twitter.com/2011/05/mission-permission.html

Questions and Answers

Please take a look on our dedicated Application Permission Model FAQ page.

← OAuth FAQ Authentication & Authorization Application Permission Model FAQ →