Upcoming Changes to PagerDuty OAuth

Intended audience: developers who have created OAuth applications with PagerDuty

PagerDuty is making some changes to its OAuth authentication infrastructure, and, as part of this work, we are deprecating some specific API features. This document is intended to help guide you through these changes, and is split into **Immediate Actions** and **Long Term Actions,** each of which has specific deprecation timelines. **To ensure that your app keeps working, make sure that you have a plan to implement these actions, if required, before the dates mentioned.**

You may have mistakenly received an email because you have installed a PagerDuty-built integration like Salesforce Data Cloud (SFDC) -- please note that these integrations have already been updated and require no customer actions.

These changes work towards keeping the PagerDuty ecosystem secure, protecting both our customers and yours, building strong foundations that will continue to support you in the future.

On that note, we have also added some **Recommended Actions**. These are optional, but they may simplify your app, and make it easier for you to maintain and support going forward.

This is a long document, but we really appreciate your time in taking a look. Note that your OAuth application may not be using all of these API features, but for thoroughness, here is a full list of upcoming changes in order of urgency.

Immediate Action Required

These issues are urgent for technical or security reasons.

Implicit OAuth Deprecated

The Implicit OAuth flow (`response_type=token`) is deprecated and cannot be used moving forward. Read here about the security problems with Implicit OAuth. Support for Implicit OAuth will be dropped after **December 15th 2021**.

To ensure your apps continue to work, you will need to make sure they only use the Authentication Code flow (`response_type=code`). For front-end apps running in the browser or on a mobile device, use PKCE without a client secret. For back-end apps, use normal client id/client secret credentials to authenticate.

PKCE Code Verifier length

The PKCE Authentication Code RFC specifies that the `code_verifier` field should be between 43 and 128 characters. Previously PagerDuty did not enforce this limit, but we will be turning on validation for this moving forward.

To ensure your apps continue to work, you will need to ensure that your app's `code_verifier` field is between the allowable number of characters. If your app uses a `code_verifier` that is longer than 128 characters, or shorter than 43, it will no longer work after **December 15th 2021**.

A small mea culpa: previously in our Developer Documentation, our sample PKCE code did not respect this limit, because it did not account for the overhead of base64-encoding a random string. This has been updated in our public docs, but for your convenience the corrected snippet is also provided below:

```
var gen128ishVerifier = function() {
  // account for the overhead of going to base64
  var bytes = Math.floor(128  / 1.37);
  var array = new Uint8Array(bytes);

  // note: there was also a bug where getRandomValues
  // was assumed to modify the reference to the array
  // and not return a value
  array = window.crypto.getRandomValues(array);
  return base64Url(array.buffer);
};
```

Long Term Action Required

These API features have been deprecated. See each item for the specific dates to be aware of.

Again, in order of urgency:

One Year Access Token Expiry

Previously, PagerDuty OAuth access tokens did not enforce a set expiry time. To put it mildly, this is not the best practice, security-wise.

As of **November 4th 2021, we began issuing access tokens with a one year expiry**, and added support for refresh tokens.

To ensure your app continues to work, you will need to make sure it handles access token expiry, at a minimum, before the new 1-year tokens begin to expire on **November 4th 2022**. For an ideal user experience, we also recommend that you implement refresh logic.

The token refresh flow currently requires providing the client secret. For public OAuth clients using the PKCE flow, we are currently exploring options for allowing the use of refresh tokens without the use of a client secret, and will be communicating those in advance of the token expiry deadline.

In the long term, once the majority of apps support expiring tokens and refresh logic, we plan to reduce the expiry window for our tokens to a much shorter time, as well as eventually revoke the existing OAuth tokens that do not currently have an expiry date. Please be on the lookout for further updates on this topic.

POST /oauth/token with query params in URL

Previously, PagerDuty supported sending parameters to the `oauth/token` endpoint through URL query parameters. This is not actually specified by the OAuth RFC, and we expect to drop this support after **April 30th 2022**, after which all parameters must be part of the POST body.

To make sure your app continues to work, you must update your implementation to send body parameters, with the content type `application/x-www-form-urlencoded`.

Scopes must be specified for /oauth/authorize

Previously, PagerDuty accepted specifying read or read/write scopes when creating an OAuth client. After **May 31st 2022**, we will require specifying scopes on the `/oauth/authorize` call instead.

To make sure your app continues to work, you must specify `scopes=read` or `scopes=write` as a GET parameter when sending users to the `oauth/authorize` endpoint.

/oauth/authorize and /oauth/token at account subdomain URL

Previously, PagerDuty supported accessing the `/oauth/authorize` and `/oauth/token` routes at both `app.pagerduty.com` and `any-account-subdomain.pagerduty.com`.

The account-specific route is now deprecated, and we are expecting to drop support for this after **June 30th 2022**.

To ensure your app continues to function, make sure it uses only `app.pagerduty.com` for OAuth authorization traffic moving forward.

Subdomain parameter will no longer be returned after /oauth/authorize

Previously, PagerDuty provided a GET parameter that specified the account subdomain that installed the app when redirecting back to your OAuth client after successful authorization.

This `subdomain` parameter will no longer be provided after **July 31st 2022**. If you still require the account subdomain, moving forward this is now provided through the new `id_token` JWT in the `/oauth/token` response.

Recommended Actions

These are net-new API features which may simplify your app, and make it easier to maintain and support in the future!

New ID Token Field in Access Token Response

Starting November 4th 2021, the `oauth/token` response now contains an `id_token` field, which is a JWT.

This is the recommended way to determine the user's `account_id`, `user_id`, and `subdomain` moving forward, and it does not require additional API calls to PagerDuty! See here for a complete guide on how to crack open this token.

The JWT is signed by PagerDuty, and you can find the public key used to sign the token at the following URL: `https://app.pagerduty.com/global/oauth/anonymous/jwks` Please note however that the signing key could be rotated at any time, so we would recommend fetching the signing key from PagerDuty on each use instead of storing it locally.

API support for PagerDuty Service Regions

PagerDuty recently launched a new, separate service region in the European Union.

Starting November 4th 2021, the `id_token` described above also includes the `aud` (audience) field, which gives an API URL that is specific to the account's service region. For example, if the user has an EU account, you could send the API requests directly to the EU Service Region at `api.eu.pagerduty.com`

Note however that this only applies to the follow-up API requests after receiving the access token -- authorization and token-issuing should still be done through `app.pagerduty.com` only.

Sending traffic directly to the correct service region is optional, but it may be helpful if your business is located in the European Union, or if you have customers there.

Thanks

A quick thanks for making it all the way to the end! We value our partnerships with external developers, and we appreciate your time and attention to these issues.

These changes work towards keeping the PagerDuty ecosystem secure, protecting both our customers and yours, building strong foundations that will continue to support us in the future.