RFC 6749: OAuth Core

- /authorize

  The endpoint which the end user (resource owner) interacts with to grant permission to the resource for the application

  Could return an authorization code or an access token

- /token

  The endpoint which the application uses to trade an authorization code or refresh token for an access token

Everything else is an extension.

Good: Because there are some gaps to cover
Bad: They're all optional

RFC 7009: Token Revocation

- /revoke

  The endpoint which applications use to deactivate (invalidate) a token

  Valid for access or refresh tokens

RFC 7662: Token Introspection

- /introspect

  The endpoint which applications use to learn more about a token

  Whether it is active or not (not revoked, within expiration)

  (Optional) Additional information such as expiration time, scopes included, the issued to client_id, etc.

RFC 7591: Dynamic Client Registration

- /register

  The endpoint which applications use to create new OAuth clients (client_id and client_secret) for provisioning new applications or users

  Introduces the concept of "metadata discovery documents"

OpenID Connect Core

- /userinfo

  The endpoint that applications use to retrieve profile information about the authenticated user

  This returns a spec-defined set of fields, depending on the permissions (scope) requested.

Working Draft: Server Discovery

- /.well-known/openid-configuration

  The endpoint which applications use to retrieve the configuration information for the OIDC server

  This returns a spec-defined set of fields.

There are lots more RFCs.

We'll cover some more.

RFC 6749: OAuth Core

- Access Token

  The token given to the application to access the protected resource on the user or application's behalf

  No formal requirements to format, contents, etc.

- Refresh Token

  The token given to the application to request a new access token on its expiration

I don't want to log in repeatedly…Can we have long lived access tokens? Days? Months? Years?

Short-Lived Access Tokens

Force an application to return to the server and exchange the Refresh Token, checking permissions

OpenID Connect Core

- ID Token

  It is specifically designed for user profile information.

  The specification details the main structures, how to name them, and when to use them.

They have fundamentally different purposes.

Token Management

RFC 6749: Access Tokens

"Access tokens are credentials used to access protected resources. An access token is a string representing an authorization issued to the client. The string is usually opaque to the client."

-Internet Engineering Task Force

Tokens are strings and might be opaque.

RFC 6749: Access Tokens

"Tokens represent specific scopes and durations of access, granted by the resource owner, and enforced by the resource server and authorization server."

-Internet Engineering Task Force

Tokens represent scopes
and expiration/duration aspects.

RFC 6749: Access Tokens

"The access token provides an abstraction layer, replacing different authorization constructs (e.g., username and password) with a single token understood by the resource server."

-Internet Engineering Task Force

Token is not a username and password.

RFC 6749: Access Tokens

"Access tokens can have different formats, structures, and methods of utilization (e.g., cryptographic properties) based on the resource server security requirements."

-Internet Engineering Task Force

What We Actually Know

- Access Tokens might include information, or reference it
- If an Access Token has data, it can be any data
- The data can be in any format
- But it will never be the username and password
- But it might be opaque anyway
- Not very helpful…

RFC 7519 JSON Web Token (JWT)

Finally, something that can help!

RFC 7519 Defines

- "iss" – the issuer of the token, an entity we trust
- "sub" – the subject (user) of the token
- "aud" – the audience or intended recipient of the token
- "exp" – the expiration time of the token
- Additional fields that aren't relevant right now…

RFC 7009: Token Revocation

- /revoke

  The endpoint which applications use to deactivate (invalidate) a token

  Valid for access or refresh tokens

  This revocation is not broadcast to applications or APIs.

We need to be careful revoking a token.

Using /revoke is not foolproof in all scenarios with all applications.

Scopes and Claims

Scope

Also known as: "What am I allowed to do?"

A group of permissions that a user can grant to an application to take action on their behalf

In OAuth, they're completely undefined.

Except when you…

Let's come back to that.

Claim

A key/value pair within the token that gives the client application information.

That's it! There's nothing else to them!

In OAuth, they're completely undefined.

RFC 7519 Defines

- "iss" – the issuer of the token, an entity we trust
- "sub"– the subject (user) of the token
- "aud" – the audience or intended recipient of the token
- "exp" – the expiration time of the token

Getting some more structure would be great!

Efforts to Standardize Scopes and Claims

- Healthcare

  Heart Working Group – http://openid.net/wg/heart

  FHIR – http://www.hl7.org/implement/standards/fhir

- Finance

  Financial API – http://openid.net/wg/fapi

OIDC Scopes	OIDC Claims
OpenId (required)	(Varies depending on the scopes)
Profile	Sub
Email	Name (many pieces)
Address	Address (street address+pieces)
Phone	Email
	And many others…

"Log in with…" uses OpenID Connect.

OpenID Connect is a huge interoperability win for OAuth.