

# Recommendations for Token Lifetimes

## (AARC-G081)

Publication Date: 2026-03-09  
Authors: AARC Community members; ApplInt members; Marcus Hardt (ed.), Valeria Ardizzone, Dave Dykstra, Christos Kanellopoulos, Tomasz Kuczyński, Nicolas Liampotis, Maarten Litmaath, Mischa Sallé, Hannah Short, Jan Pavlíček, Gabriel Zachmann

Document Code: AARC-G081  
Supported by: the AARC TREE project  
Publishing Organisation: AARC Community  
DOI: 10.5281/zenodo.16737674

© members of the AARC community.  
This work is licensed under a Creative Commons Attribution 4.0 License.

### Abstract

*This document provides a short overview over selected types of tokens used to identify and authorise users. We analyse the different properties of tokens and categorise available authorisation patterns to give recommendations about the lifetimes of tokens associated with specific properties and authorisation levels.*

|                                  |          |
|----------------------------------|----------|
| <b>1. Introduction</b>           | <b>2</b> |
| 1.1. Conventions                 | 2        |
| <b>2. Token Properties</b>       | <b>3</b> |
| <b>3. Scope of this document</b> | <b>3</b> |
| <b>4. Recommendations</b>        | <b>3</b> |
| <b>5. References</b>             | <b>4</b> |

# 1. Introduction

For authenticating and authorising users to services, credentials of various types are being used. This document focuses on lifetimes of OAuth2 tokens.

Compromised tokens could be used by attackers to impersonate the original owner of the token. Recommendations for token lifetimes must take the degree of protection, the impact in case of loss, and existing mitigation mechanisms into account. This is particularly important when tokens are being used across infrastructures.

Recommendations are therefore needed in the multitude of different scientific services and infrastructures, to cater for common lifetimes and the underlying security assessments. Finally, infrastructure users, infrastructure operators, and IT security experts have diverging opinions about optimal lifetimes.

The recommendations were collected by consulting representatives of several user communities, as well as security, architecture, and policy representatives. Lifetimes of commercial offerings have been compared with.

Recommendations on lifetime are not based on the strength of the underlying cryptographic algorithms used. The recommendations given in this document reflect the impact that may be caused when tokens are obtained by an attacker.

Operational token lifetimes ultimately depend on balancing risks, between the security properties of the system (ensuring the triad of confidentiality, integrity, and availability) and the requirements of the use cases. Considerations include operational and functional constraints and the ability to recover from compromise in the systems (i.e. the impact of compromise and the time to restoration of the intended state).

In this document, we give a high priority to interoperability between infrastructures. I.e. the recommended lifetimes were chosen to work with the large majority of use cases that require interoperability between infrastructures. Use cases may exist that require different token lifetimes. Those use cases require special attention for appropriate mitigation mechanisms and may require special agreements, when cross infrastructure use is needed. Such use cases are out of scope of this document.

## 1.1. Conventions

In this document the key terms 'must', 'must not', 'required', 'shall', 'shall not', 'recommended', 'may', and 'optional' are to be interpreted as described in RFC 2119. If a 'should' or 'should not' is not followed, the reasoning for this exception must be explained to make an informed decision about accepting the exception, or the implementer must be able to defend that an equivalent or better solution is in place [\[IGTF\]](#).

## 2. Token Properties

Tokens can be characterised by different properties they have. Those relevant for this document are:

- **Bound:** Tokens can be bound to a specific instance of a client. This limits the potential impact of compromised tokens. However, this may limit support for delegation scenarios.
- **Rotation:** Rotation enforces that each refresh token can only be used once with the issuer of the token. Upon use, a new refresh token will be returned along with the response to the actual request. This may be used to detect compromised tokens, because rotation will fail when an attacker previously used the compromised token. This should trigger a revocation of this chain of tokens, and requires user and tool awareness. While it increases security, rotation has a potential of erroneously revoking tokens.
- **Revocable:** Revoked tokens can not be used any longer, despite not having expired yet. This allows longer lifetimes than for non-revocable tokens. In situations where tokens are verified offline (e.g. via a signature mechanism), additional measures (i.e. to convey the revocation information to resource servers) may be required (e.g. in OAuth2).
- **Signed:** Signed tokens can be cryptographically verified by a recipient. This allows tokens to be evaluated by a Resource Server (RS) without contacting the Authorisation Server (AS), therefore also called "offline verification". This feature may be used when the authentication process must not depend on the availability of a network connection, or to reduce load on the (often central) token issuer. Since the issuer does not need to be contacted for their verification, signed tokens might not be revocable unless additional measures<sup>1</sup> are taken.

## 3. Scope of this document

In scope for this document are cross-infrastructure use cases, e.g. a web portal of one infrastructure wants to browse data stored in a different infrastructure. This generally includes web flows as well as API access.

Out of scope are multi-step delegation scenarios which often implement automation, and other situations in which tokens need to be readily available throughout the (potentially very long) lifetime of a session with the user present only initially (if these use cases cannot be addressed with the standard OAuth2/OIDC refresh token flows). Whilst the guidelines proposed by these documents may be applicable for some of those cases, the specific risks presented by such workflows may be more effectively mitigated via alternative methods not described here.

OAuth2 distinguishes between confidential and public clients. The implications introduced by public clients are complex. This document therefore treats only confidential clients as in scope and does not give recommendations on tokens for public clients.

---

<sup>1</sup> Even though [RFC7009](#) specifies revocation, this does not necessarily apply to signed tokens, since those can be verified "offline" i.e. regardless of their revocation status on the AS.

## 4. Recommendations

The recommendations reflect a range of lifetimes that SHOULD be used unless specific requirements apply. We give recommendations for a default, a minimal and a maximal lifetime. The minimal and maximal lifetimes SHOULD NOT be reduced or exceeded, respectively. The default lifetimes MAY be altered.

1. **Access Tokens** that can be **verified offline** (and therefore can de facto not be revoked)
  - **Default: 1 hour.** Typically long enough for reaching a protected resource and interacting with it, e.g. going through a login process.
  - **Min: 15 minutes.** This is aligned with the typical session lifetimes at a service.
  - **Max: 6 hours.** In line with typical incident response times.
2. **Access Tokens** that must be **verified online** (and are therefore revocable, online verification MUST be enforced e.g. by not signing the tokens)
  - **Default 1 hour.** Same as SAML, enough for reaching a protected resource and doing something with it, like going through a login process.
  - **Min: 15 minutes.** Enough for reaching the protected resource.
  - **Max: 25 hours.** Taken into account: revocability; enough time to run a short job; checking results of the previous day.
3. **Refresh Tokens** are designed to live longer than Access Tokens. Unlike Access Tokens, they are not bearer tokens. They are normally bound to a specific `client_id` and supposed to be kept in a safe place (i.e. not be transported with user data across infrastructures). Refresh tokens themselves can be refreshed<sup>2</sup>. We recommend refreshing RTs only until the maximum time specified below is reached<sup>3</sup>.
  - **Default: 30 days.** Roughly the geometric mean between a day and a year.
  - **Min: 24 hours.** Helps avoid high loads on issuers.
  - **Max: 400 days.** Proof of the owner still being involved.

## 5. References

- [IGTF] IGTF AP Classic  
<https://www.eugridpma.org/guidelines/classic-doc/IGTF-AP-classic-5-0.pdf>
- [RFC7009] OAuth2 Token Revocation  
<https://datatracker.ietf.org/doc/html/rfc7009>

---

<sup>2</sup> Token Rotation (RFC 6749 Sec 10.4) may be considered.

<sup>3</sup> This is often used to ensure users will need to prove their presence periodically. Other methods of enforcing this may exist and take precedence.