International Standard

ISO/IEC 26135

Information technology — OpenID connect — OpenID connect session management 1.0

First edition
2024-10

Reference number
ISO/IEC 26135:2024(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by the OpenID Foundation (OIDF) (as OpenID Connect Session Management 1.0) and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Abstract

TOC

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

This document describes how to manage sessions for OpenID Connect, including when to log out the End-User.

Table of Contents

Information technology — OpenID Connect — OpenID Connect Session Management 1.0

1. Introduction

TOC

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 [RFC6749] protocol. It enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

This specification complements the OpenID Connect Core 1.0 [OpenID.Core] specification by defining how to monitor the End-User's login status at the OpenID Provider on an ongoing basis so that the Relying Party can log out an End-User who has logged out of the OpenID Provider.

Both this specification and the OpenID Connect Front-Channel Logout 1.0 [OpenID.FrontChannel] specification use front-channel communication, which communicate logout requests from the OP to RPs via the User Agent. In contrast, the OpenID Connect Back-Channel Logout 1.0 [OpenID.BackChannel] specification uses direct back-channel communication between the OP and RPs being logged out. The OpenID Connect RP-Initiated Logout 1.0 [OpenID.RPInitiated] specification complements these specifications by defining a mechanism for a Relying Party to request that an OpenID Provider log out the End-User. This specification can be used separately from or in combination with these other three specifications.

1.1. Requirements Notation and Conventions

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

In the .txt version of this document, values are quoted to indicate that they are to be taken literally. When using these values in protocol messages, the quotes MUST NOT be used as part of the value. In the HTML version of this document, values to be taken literally are indicated by the use of `this fixed-width font`.

1.2. Terminology

TOC

This specification uses the terms "Authorization Endpoint", "Authorization Server", "Client", and "Client Identifier" defined by OAuth 2.0 [RFC6749], the term "User Agent" defined by RFC 7230 [RFC7230], and the terms defined by OpenID Connect Core 1.0 [OpenID.Core].

This specification also defines the following term:

Session

> Continuous period of time during which an End-User accesses a Relying Party relying on the Authentication of the End-User performed by the OpenID Provider.

IMPORTANT NOTE TO READERS: The terminology definitions in this section are a normative portion of this specification, imposing requirements upon implementations. All the capitalized words in the text of this specification, such as "Session", reference these defined terms. Whenever the reader encounters them, their definitions found in this section must be followed.

2. Creating and Updating Sessions

TOC

In OpenID Connect, the session at the RP typically starts when the RP validates the End-User's ID Token. Refer to the OpenID Connect Core 1.0 [OpenID.Core] specification to find out how to obtain an ID Token and validate it. When the OP supports session management, it MUST also return the Session State as an additional `session_state` parameter in the Authentication Response and SHOULD also return the Session State as an additional `session_state` parameter in the Authentication Error Response. The OpenID Connect Authentication Response is specified in Section 3.1.2.5 of OpenID Connect Core 1.0. The OpenID Connect Authentication Error Response is specified in Section 3.1.2.6 of OpenID Connect Core 1.0.

This parameter is:

session_state

Session State. JSON [RFC7159] string that represents the End-User's login state at the OP. It MUST NOT contain the space (" ") character. This value is opaque to the RP. This is REQUIRED if session management is supported.

The Session State value is initially calculated on the server. The same Session State value is also recalculated by the OP iframe in the User Agent. The generation of suitable Session State values is specified in Section 3.2, and is based on a salted cryptographic hash of Client ID, origin URL, and OP User Agent state. For the origin URL, the server can use the origin URL of the Authentication Response, following the algorithm specified in Section 4 of RFC 6454 [RFC6454].

3. Session Status Change Notification

It is highly desirable to be able to determine the login status of the End-User at the OP. To do so, it is possible to repeat the Authentication Request with `prompt=none`. However, this causes network traffic and this is problematic on the mobile devices that are becoming increasingly popular. Therefore, once the session is established with the Authentication Request and Response, it is desirable to be able to check the login status at the OP without causing network traffic by polling a hidden OP iframe from an RP iframe with an origin restricted postMessage as follows.

3.1. RP iframe

The RP loads an invisible iframe from itself. This iframe MUST know:

- the ID of the OP iframe, as described in Section 3.2, so that it can postMessage to the OP iframe, and
- the origin URL of the OP iframe, so that it can ensure messages are dispatched to and only processed when originating from the OP.

The RP iframe polls the OP iframe with postMessage at an interval suitable for the RP application. With each postMessage, it sends the session state defined in Section 3.2. The RP iframe MUST enforce that it only processes messages from the origin of the OP frame. It MUST reject postMessage requests from any other source origin, to prevent cross-site scripting attacks.

The postMessage from the RP iframe delivers the following concatenation as the data:

- Client ID + " " + Session State

It also has to be able to receive the postMessage back from the OP iframe. The received data will either be `changed` or `unchanged` unless the syntax of the message sent was determined by the OP to be malformed, in which case the received data will be `error`. Upon receipt of `changed`, the RP MUST perform re-authentication with `prompt=none` to obtain the current session state at the OP. Upon receipt of `error`, the RP MUST NOT perform re-authentication with `prompt=none`, so as to not cause potential infinite loops that generate network traffic to the OP.

Following is non-normative example pseudo-code for the RP iframe:

```
  var stat = "unchanged";
  var mes = client_id + " " + session_state;
  var targetOrigin = "https://server.example.com"; //
Validates origin
  var opFrameId = "op";
  var timerID;

  function check_session()   {
    var win =
window.parent.frames[opFrameId].contentWindow
    win.postMessage(mes, targetOrigin);
  }

  function setTimer() {
    check_session();
    timerID = setInterval(check_session, 5 * 1000);
  }

  window.addEventListener("message", receiveMessage,
false);

  function receiveMessage(e) {
    if (e.origin !== targetOrigin) {
      return;
    }
    stat = e.data;
  if (stat === "changed") {
    clearInterval(timerID);
    // then take the actions below...
  }
}

setTimer();
```

When the RP detects a session state change, it SHOULD first try a `prompt=none` request within an iframe to obtain a new ID Token and session state, sending the old ID Token as the `id_token_hint`. If the RP receives an ID token for the same End-User, it SHOULD simply update the value of the session state. If it does not receive an ID token or receives an ID token for another End-User, then it needs to handle this case as a logout for the original End-User. If the original End-User is already logged out at the RP when the state changes indicate that End-User should be logged out, the logout is considered to have succeeded.

Note that the session state is origin bound. In deployments with multiple subdomains sharing the same RP session, it is important that the parent window and the RP iframe both set the same `document.domain` to comply with same-origin restrictions. This will allow the RP iframe to target the parent window's embedded OP iframe.

3.2. OP iframe

TOC

The RP also loads an invisible OP iframe into itself from the OP's `check_session_iframe`. The RP MUST assign an `id` attribute to the iframe so that it can address it, as described above. The OP iframe MUST enforce that the caller is from an expected origin. It MUST reject postMessage requests from any other source origin to prevent cross-site scripting attacks.

As specified in Section 3.1, the postMessage from the RP iframe delivers the following concatenation as the data:

- Client ID + " " + Session State

The OP iframe has access to User Agent state at the OP (in a cookie or in HTML5 storage) that it uses to calculate and compare with the OP session state that was passed by the RP. The OP iframe MUST recalculate it from the previously obtained Client ID, the source origin URL (from the postMessage), and the current OP User Agent state. The session state includes all of this information for privacy reasons, so that different clients active in the same User Agent have distinct session state values.

If the postMessage received is syntactically malformed in such a way that the posted Client ID and origin URL cannot be determined or are syntactically invalid, then the OP iframe SHOULD postMessage the string `error` back to the source. If the received value and the calculated value do not match, then the OP iframe MUST postMessage the string `changed` back to the source. If it matched, then it MUST postMessage the string `unchanged`.

Following is non-normative example pseudo-code for the OP iframe:

```
  window.addEventListener("message", receiveMessage,
false);

  function receiveMessage(e){ // e.data has client_id
and session_state

    var client_id = e.data.substr(0,
e.data.lastIndexOf(' '));
    var session_state =
e.data.substr(e.data.lastIndexOf(' ') + 1);
    var salt = session_state.split('.')[1];

    // if message is syntactically invalid
    //     postMessage('error', e.origin) and return

    // if message comes an unexpected origin
    //     postMessage('error', e.origin) and return

    // get_op_user_agent_state() is an OP defined
function
    // that returns the User Agent's login status at the
OP.
    // How it is done is entirely up to the OP.
    var opuas = get_op_user_agent_state();

    // Here, the session_state is calculated in this
particular way,
    // but it is entirely up to the OP how to do it
under the
    // requirements defined in this specification.
    var ss = CryptoJS.SHA256(client_id + ' ' + e.origin
+ ' ' +
      opuas + ' ' + salt) + "." + salt;

    var stat = '';
    if (session_state === ss) {
      stat = 'unchanged';
    } else {
      stat = 'changed';
    }
    e.source.postMessage(stat, e.origin);
  };
```

The OP User Agent state is typically going to be stored in a cookie or HTML5 local storage. It is origin bound to the Authorization Server. It captures meaningful events such as logins, logouts, change of user, change of authentication status for Clients being used by the End-User, etc. Thus, the OP SHOULD update the value of the User Agent state in response to such meaningful events. As a result, the next call to check_session() after such an event will return the value `changed`. It is RECOMMENDED that the OP not update the User Agent state too frequently in the absence of meaningful events so as to spare excessive network traffic at the Client in response to spurious `changed` events.

The computation of the session state returned in response to unsuccessful Authentication Requests SHOULD, in addition to the User Agent state, incorporate sufficient randomness in the form of a salt so as to prevent identification of an End-User across successive calls to the OP's Authorization Endpoint.

In the case of an authorized Client (successful Authentication Response), the OP SHOULD change the value of the session state returned to the Client under one of the following events:

- The set of users authenticated to the User Agent changes (login, logout, session add).
- The authentication status of Clients being used by the End-User changes.

In addition, the User Agent state used to verify the session state SHOULD change with such events. Calls to check_session() will return `changed` against earlier versions of session state after such events. It is RECOMMENDED that the User Agent state SHOULD NOT vary too frequently in the absence of such events to minimize network traffic caused by the Client's response to `changed` notifications.

In the case of an unsuccessful Authentication Request that results in an Authentication Error Response as specified in Section 3.1.2.6 of OpenID Connect Core 1.0, the value of the session state returned SHOULD vary with each request. However, the User Agent session state need not change unless a meaningful event happens. In particular, many values of session state can be simultaneously valid, for instance by the introduction of random salt in the session states issued in response to unsuccessful Authentication Requests.

If a cookie is used to maintain the OP User Agent state, the HttpOnly flag likely cannot be set for this cookie because it needs to be accessed from JavaScript. Therefore, information that can be used for identifying the user should not be put into the cookie, as it could be read by unrelated JavaScript.

In some implementations, `changed` notifications will occur only when changes to the End-User's session occur, whereas in other implementations, they might also occur as a result of changes to other sessions between the User Agent and the OP. RPs need to be prepared for either eventuality, silently handling any false positives that might occur.

3.3. OpenID Provider Discovery Metadata

TOC

To support OpenID Connect Session Management, the RP needs to obtain the Session Management related OP metadata. This OP metadata is normally obtained via the OP's Discovery response, as described in OpenID Connect Discovery 1.0 [OpenID.Discovery], or MAY be learned via other mechanisms.

This OpenID Provider Metadata parameter MUST be included in the Server's discovery responses when Session Management and Discovery are supported:

check_session_iframe

> REQUIRED. URL of an OP iframe that supports cross-origin communications for session state information with the RP Client, using the HTML5 postMessage API. This URL MUST use the `https` scheme and MAY contain port, path, and query parameter components. The page is loaded from an invisible iframe embedded in an RP page so that it can run in the OP's security context. It accepts postMessage requests from the relevant RP iframe and uses postMessage to post back the login status of the End-User at the OP.

4. Validation

TOC

If any of the validation procedures defined in this specification fail, any operations requiring the information that failed to correctly validate MUST be aborted and the information that failed to validate MUST NOT be used.

5. Implementation Considerations

TOC

This specification defines features used by both Relying Parties and OpenID Providers that choose to implement Session Management. All of these Relying Parties and OpenID Providers MUST implement the features that are listed in this specification as being "REQUIRED" or are described with a "MUST".

5.1. User Agents Blocking Access to Third-Party Content

TOC

Note that at the time of this writing, some User Agents (browsers) are starting to block access to third-party content by default to block some mechanisms used to track the End-User's activity across sites. Specifically, the third-party content being blocked is website content with an origin different that the origin of the focused User Agent window. Site data includes cookies and any web storage APIs (sessionStorage, localStorage, etc.).

This can prevent the ability for notifications from the OP at the RP from being able to access the RP's User Agent state to implement local logout actions. In particular, cookies and web storage APIs may not be available in the OP frame loaded in the RP context. The side effect here is that, depending on the used mechanism (cookies or web storage), the data needed to recalculate `session_state` might not be available. Cookie based implementations might then return `changed` for every single call, resulting in infinite loops of re-authentications. Therefore, deployments of this specification are recommended to include defensive code to detect this situation, and if possible, notify the End-User that the requested RP logouts could not be performed. The details of the defensive code needed are beyond the scope of this specification; it may vary per User Agent and may vary over time, as the User Agent tracking prevention situation is fluid and continues to evolve.

OpenID Connect Back-Channel Logout 1.0 [OpenID.BackChannel] is not known to be affected by these developments.

6. Security Considerations

TOC

The OP iframe MUST enforce that the caller is from an expected origin. It MUST reject postMessage requests from any other source origin to prevent cross-site scripting attacks.

The RP iframe MUST enforce that it only processes messages from the origin of the OP frame. It MUST reject postMessage requests from any other source origin to prevent cross-site scripting attacks.

7. IANA Considerations

TOC

7.1. OAuth Parameters Registry

TOC

This specification registers the following parameter in the IANA "OAuth Parameters" registry [IANA.OAuth.Parameters] established by RFC 6749 [RFC6749].

7.1.1. Registry Contents

TOC

- Parameter name: `session_state`
- Parameter usage location: Authorization Response, Access Token Response
- Change controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
- Specification document(s): Section 2 of this document
- Related information: None

7.2. OAuth Authorization Server Metadata Registry

TOC

This specification registers the following metadata name in the IANA "OAuth Authorization Server Metadata" registry [IANA.OAuth.Parameters] established by [RFC8414].

7.2.1. Registry Contents

TOC

- Metadata Name: `check_session_iframe`
- Metadata Description: URL of an OP iframe that supports cross-origin communications for session state information with the RP Client, using the HTML5 postMessage API
- Change Controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
- Specification Document(s): Section 3.3 of this document

8. References

TOC

8.1. Normative References

TOC

[IANA.OAuth.Parameters] IANA, “OAuth Parameters.”

[OpenID.BackChannel] Jones, M. and J. Bradley, “OpenID Connect Back-Channel Logout 1.0,” September 2022.

[OpenID.Core] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, “OpenID Connect Core 1.0,” November 2014.

[OpenID.Discovery] Sakimura, N., Bradley, J., Jones, M., and E. Jay, “OpenID Connect Discovery 1.0,” November 2014.

[OpenID.FrontChannel] Jones, M., "OpenID Connect Front-Channel Logout 1.0," September 2022.

[OpenID.RPInitiated] Jones, M., de Medeiros, B., Agarwal, N., Sakimura, N., and J. Bradley, "OpenID Connect RP-Initiated Logout 1.0," September 2022.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.

[RFC6454] Barth, A., "The Web Origin Concept," RFC 6454, DOI 10.17487/RFC6454, December 2011.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework," RFC 6749, DOI 10.17487/RFC6749, October 2012.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format," RFC 7159, DOI 10.17487/RFC7159, March 2014.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," RFC 7230, DOI 10.17487/RFC7230, June 2014.

8.2. Informative References

TOC

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata," RFC 8414, DOI 10.17487/RFC8414, June 2018.

ICS 35.030

Price based on 12 pages

iso.org