XEP-0516: XMPP Decentralized ID (XID)

Jérôme Poisson
mailto:goffi@goffi.org
xmpp:goffi@jabber.fr
https://www.goffi.org

2026-06-30
Version 0.1.0

Status	Type	Short Name
Experimental	Standards Track	xid

XMPP Decentralized ID (XID) is a DNS independent XMPP entity identifier. This specification describes how to generate, use, and handle them.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Identification on XMPP is done with the Jabber ID (JID), which is an email-like, human-friendly identifier. While handy, JID is usually tied to a Domain Name Server, which makes it difficult to change server (while Moved (XEP-0283) [1] helps, it needs the server to be available and cooperating), and which is not usable in a serverless environment.
XMPP Decentralized ID (XID) uses a simple mechanism based on cryptographic signature to have an entity identity independent of any central organization or component. It is designed to be simple to implement and use, and to stay compatible with normal JID.
XID can be used to implement the "cryptographic verification" mentioned in Moved (XEP-0283) [2] with the benefit that the old server doesn't need to be online or cooperating (thus it will work if the server is down or actively blocking migrations). It can also be used in a serverless environment. XID allows keeping track of entities, so if a user's contacts change server, or communicate in a serverless configuration, a client can use it to verify who it is talking to, and to keep all existing cached data (message history, vcard, avatars, etc.). XID could also be used to sign any external data to associate it with the XMPP entity, like a document or an image, or metadata (e.g., to associate a resource like a website with an XMPP entity).

2 Requirements

The design goals of this XEP are:

- Be independent of any central authority or component.
- Be as simple as possible to implement, re-using existing components when possible.
- Be usable in serverless configurations.
- Be future proof by allowing to change cryptographic algorithms.

3 Overview

A XID is derived from a cryptographic signature. It currently uses Ed25519 (which is already used for OMEMO Encryption (XEP-0384) [3] so most clients have an implementation already). The private key is shared between devices of the entity, and the public key is used to generate the ID (printed as a hex string in lowercase, with a prefix byte). The ID can then be used as a JID in the "<ID>@id.internal" form (this is this form that we call a "XID").
The XID is published on a well-known PEP node ('urn:xmpp:xid'), and private key synchronization is done automatically via end-to-end encrypted messages (via OMEMO Encryption

[1] XEP-0283: Moved <https://xmpp.org/extensions/xep-0283.html>.
[2] XEP-0283: Moved <https://xmpp.org/extensions/xep-0283.html>.
[3] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.

(XEP-0384) [4] or OpenPGP for XMPP Instant Messaging (XEP-0374) [5]) or another means. XID can also be revoked via a PEP node.

4 XID Generation and JID Mapping

To generate a XID, an entity must generate a public key and a private key. The former will be used to generate the ID, and the latter must be shared between devices of the entity, and kept secret otherwise (see below for private key synchronization).
The private key can be used to sign any kind of file or document, even outside of XMPP, and the corresponding public key can be used to verify those signatures. This allows an entity to associate external data with its XID identity independently of any XMPP communication.
To generate the ID, the public key is converted to a hex string (which MUST be lowercase), then the prefix byte "00" is prepended to indicate that Ed25519 is used. A JID is built by using this ID as node part, and "id.internal" as domain part.

4.1 Example

Juliet's client generates her XID. To do so, it first generates a pair of private and public keys. It gets those values, converted to hex strings:

- Private key (hex): 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
- Public key (hex): 03a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc8664125531b8

The public key is used for the ID. As the key has been generated with Ed25519, the "00" byte is prepended to it, so it becomes 0003a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc8664125531b8. Finally, the "@id.internal" domain is used to generate the corresponding JID, which is 0003a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc8664125531b8@id.internal.
The final JID is called Juliet's XID.

5 Publishing and Revoking XID

5.1 Publishing

A XID can be published on an entity's Personal Eventing Protocol (XEP-0163) [6] to the well-known node 'urn:xmpp:xid'. By default, and according to XEP-0163, the access model MUST

[4] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.
[5] XEP-0374: OpenPGP for XMPP Instant Messaging <https://xmpp.org/extensions/xep-0374.html>.
[6] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.

be ”presence”, but a user MAY choose to use another access model (e.g., ”open”) if they want to have their XID more or less accessible.
The item payload consists of a <xid/> element qualified by the ’urn:xmpp:xid:0’ namespace which MUST have a ’created’ attribute with a DateTime value as specified in XMPP Date and Time Profiles (XEP-0082) [7] to indicate when the XID has been generated. The <xid/> element content MUST be the generated JID as explained in XID Generation and JID Mapping.
The main XID MUST be published to an item with an itemID of ”current” as explained in XEP-0060 §12.21 Singleton Nodes. Other XIDs can be added as backups with other IDs.
Entities SHOULD subscribe to the ’urn:xmpp:xid’ PEP node with the +notify option to be informed of changes published by other devices. Before publishing a new XID, an entity SHOULD retrieve the items from the ’urn:xmpp:xid’ PEP node to obtain the current state and check whether a XID is already published. If a XID with itemID ”current” already exists, the entity SHOULD NOT overwrite it with a different XID.

Listing 1: Juliet publishes her XID

```
<iq type='set'
    from='juliet@capulet.lit/balcony'
    to='juliet@capulet.lit'
    id='xidpub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:xid'>
      <item id='current'>
        <xid xmlns='urn:xmpp:xid:0'
             created='2026-05-27T14:30:00Z'>
          0003
              a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc8664125531b8@id
              .internal
        </xid>
      </item>
    </publish>
  </pubsub>
</iq>
```

5.2 Revoking

To revoke a XID (because it has been compromised, for rotation, or any other reason), it first MUST be removed from the ’urn:xmpp:xid’ PEP node. Then a <revoked/> element MUST be published on the ’urn:xmpp:xid:revoked’ well-known PEP node of the entity.
The item payload consists of a <revoked/> element qualified by the ’urn:xmpp:xid:0’ namespace which MUST have ’created’ and ’revoked’ attributes with DateTime values as specified in XMPP Date and Time Profiles (XEP-0082) [8]; the ’created’ attribute indicates the XID creation datetime (the same as in publishing), while the ’revoked’ attribute indicates the datetime of

[7] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.
[8] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.

the revocation. The <revoked/> element content MUST be the generated JID of the revoked XID as explained in XID Generation and JID Mapping.
The <revoked/> element MAY have a <reason/> child element whose content is the human-readable reason for the revocation.
If the main XID (the one with the "current" ID in 'urn:xmpp:xid' PEP node) is revoked, the revoking client MUST publish another XID with this ID, either by republishing an existing one with this ID (and removing the element with the old ID), or by generating a new one.

5.2.1 Revoking Example

Juliet wants to revoke her old XID because she suspects it may have been compromised. She first removes it from the active node, then publishes the revocation on the revoked node.

Listing 2: Juliet removes her old XID from the active node

```
<iq type='set'
    from='juliet@capulet.lit/balcony'
    to='juliet@capulet.lit'
    id='xidrev1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <retract node='urn:xmpp:xid'>
      <item id='current'/>
    </retract>
  </pubsub>
</iq>
```

Listing 3: Juliet publishes the revocation on the revoked node

```
<iq type='set'
    from='juliet@capulet.lit/balcony'
    to='juliet@capulet.lit'
    id='xidrev2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:xid:revoked'>
      <item id='0003
          a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc8664125531b8
          '>
        <revoked xmlns='urn:xmpp:xid:0'
                 created='2026-05-27T14:30:00Z'
                 revoked='2026-05-30T09:15:00Z'>
          0003
              a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc8664125531b8@id
              .internal
        </revoked>
      </item>
    </publish>
  </pubsub>
```

```
</iq>
```

6 Identity Challenge

To verify that an entity is allowed to use the XID, an XMPP entity can send a <message/> stanza with the default type (”chat”) to the bare JID of the entity to verify. This stanza MUST have a <challenge/> element qualified by the ’urn:xmpp:xid:0’ namespace. The <challenge/> MUST have a ’xid’ attribute with the XID to verify, a ’timestamp’ attribute with a DateTime value as specified in XMPP Date and Time Profiles (XEP-0082) [9] indicating when the challenge was issued, and MUST have a nonce for its content (i.e., a cryptographically strong random number used only once). The nonce MUST be encoded as a hex string (lowercase).
When the <challenge/> element is received by any of the devices having the private key corresponding to this XID, it MUST sign the requested nonce and send a message to the entity requesting the challenge with a <response/> element qualified by the ’urn:xmpp:xid:0’ namespace. The <response/> element MUST have a ’xid’ attribute with the XID being verified, a ’timestamp’ attribute matching the ’timestamp’ from the <challenge/> element, and MUST use the hex-encoded signature of the nonce as content. The entity verifier MUST then check that the signature is correct for this nonce and that the public key corresponds to the XID (i.e., the localpart of the JID from which the algorithm prefix byte has been stripped).
Notes:

1. <message/> is used here instead of <iq/> because it lets request the bare entity and it can be replied asynchronously if no device is connected at the moment of the challenge.

2. Because <message/> is used, several devices may receive the challenge thanks to Message Carbons (XEP-0280) [10]. If a device sees that the <response/> element has already been sent by another device, it SHOULD NOT send this element itself. However, the verifying entity MUST expect to receive several <response/> elements. Only the first one received MUST be verified, following ones MUST be ignored.

6.1 Verification Example

Romeo wants to verify that Juliet is the legitimate owner of her XID. He sends a challenge with a random nonce, and Juliet’s device signs it and returns the response.

Listing 4: Romeo verifies Juliet’s XID

```
<message type='chat'
         from='romeo@montague.lit/orchard'
         to='juliet@capulet.lit'>
  <challenge xmlns='urn:xmpp:xid:0'
```

[9] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.
[10] XEP-0280: Message Carbons <https://xmpp.org/extensions/xep-0280.html>.

```
            xid='0003
                a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc8664125531b8@id
                .internal'
            timestamp='2026-05-30T10:15:30Z'>
    a3f2c8b1e9d74560
  </challenge>
</message>
```

Listing 5: Juliet responds to the challenge

```
<message type='chat'
         from='juliet@capulet.lit/balcony'
         to='romeo@montague.lit/orchard'>
  <response xmlns='urn:xmpp:xid:0'
            xid='0003
                a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc8664125531b8@id
                .internal'
            timestamp='2026-05-30T10:15:30Z'>
    7
        f2be0038e2f62b4ab6688440e07cd5939549feb810fc2514a26282d35056d3aea60c8c102dd3db

  </response>
</message>
```

7 Private Key Synchronization

Devices may use several ways to share private keys. Two methods are described below: QR Code exchange and automatic synchronization.

7.1 QR Code Exchange

The private key can be encoded in an xmpp: URI as follows:

```
xmpp:<public_key>@id.internal?;xid-private=<private_key>;xid-created=<
    timestamp>
```

Both keys are hex-encoded as specified in XID Generation. The 'xid-private' query parameter contains the private key corresponding to the XID. The 'xid-created' query parameter matches the 'created' attribute of the published XID item and uses a DateTime value as specified in XMPP Date and Time Profiles (XEP-0082) [11]. Before importing the private key, the scanning device MUST verify that the public key in the URI matches one of the XIDs published on its own entity's PEP node, and MUST verify that the public key derived from the private key matches the public key in the URI.

[11] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.

Listing 6: QR Code URI for Juliet's XID private key

```
xmpp:0003a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc8664125531b8@id
    .internal?;xid-private=000102030405060708090
    a0b0c0d0e0f101112131415161718191a1b1c1d1e1f;xid-created=2026-05-27
    T14:30:00Z
```

7.2 Automatic Synchronization

This method provides an automatic private key synchronization between known devices, and works asynchronously (devices don't need to be connected at the same time, or in the same location). It uses OMEMO Encryption (XEP-0384) [12] with Stanza Content Encryption (XEP-0420) [13].
When a device discovers a XID linked to its own identity (e.g., by checking its own 'urn:xmpp:xid' PEP node as explained in Publishing and Revoking XID) and it doesn't know the private key, it can request it by sending a <message/> stanza with the <private-key-request/> element qualified by the 'urn:xmpp:xid:0' namespace. This element MUST have a 'xid' attribute set to the XID it wants the private key for, and MUST be sent to its own bare JID. The request MUST use OMEMO Encryption (XEP-0384) [14] encryption and the element MUST be encrypted with Stanza Content Encryption (XEP-0420) [15] as explained in XEP-0384 §5.5 Sending a Message. This message will then be copied to all entity devices thanks to Message Carbons (XEP-0280) [16].
When a <private-key-request/> is received by a device, it MUST:

- Check that the sender is from its own entity (i.e. same bare JID, and a different resource than itself).
- Check that the message has been correctly end-to-end encrypted as explained above.
- Check that the 'xid' attribute corresponds to a XID associated with this entity.
- Check that it knows the private key of this XID.
- Check that no other device has already replied to this request.

If all these conditions are respected, the entity SHOULD send a <message/> stanza with a <private-key/> element qualified by the 'urn:xmpp:xid:0' namespace. This element MUST have a 'xid' attribute with the XID corresponding to this private key. The content of the element MUST be the private key encoded as a hex string (lowercase). The message MUST be end-to-end encrypted with OMEMO Encryption (XEP-0384) [17] with Stanza Content Encryption (XEP-0420) [18] as explained in XEP-0384 §5.5 Sending a Message, and this message MUST be

[12] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.
[13] XEP-0420: Stanza Content Encryption <https://xmpp.org/extensions/xep-0420.html>.
[14] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.
[15] XEP-0420: Stanza Content Encryption <https://xmpp.org/extensions/xep-0420.html>.
[16] XEP-0280: Message Carbons <https://xmpp.org/extensions/xep-0280.html>.
[17] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.
[18] XEP-0420: Stanza Content Encryption <https://xmpp.org/extensions/xep-0420.html>.

encrypted only for known devices of this entity.
Note: A SHOULD is used here and not a MUST because this feature is optional.

7.2.1 Synchronization Example

Juliet connects a new device (her tablet) and it discovers her XID but doesn't have the private key. It sends a request to her own bare JID, and her phone responds with the key.

Listing 7: Juliet's new device requests the XID private key

```
<message type='chat'
         from='juliet@capulet.lit/tablet'
         to='juliet@capulet.lit'>
  <encrypted xmlns='urn:xmpp:omemo:2'>
    <header sid='12345'>
      <keys jid='juliet@capulet.lit'>
        <key rid='67890'><!-{}- base64 key -{}-></key>
        <key rid='09876'><!-{}- base64 key -{}-></key>
      </keys>
    </header>
    <payload>
      <!-{}- The payload is normally encrypted and unreadable;
        we show it here in plaintext to illustrate the protocol -{}->
      <envelope xmlns='urn:xmpp:sce:1'>
        <content>
          <private-key-request xmlns='urn:xmpp:xid:0'
                               xid='0003
                                   a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc8
                                   .internal'/>
        </content>
        <rpad>...</rpad>
        <from jid='juliet@capulet.lit/tablet'/>
      </envelope>
    </payload>
  </encrypted>
</message>
```

Listing 8: Juliet's phone responds with the private key

```
<message type='chat'
         from='juliet@capulet.lit/phone'
         to='juliet@capulet.lit'>
  <encrypted xmlns='urn:xmpp:omemo:2'>
    <header sid='67890'>
      <keys jid='juliet@capulet.lit'>
        <key rid='12345'><!-{}- base64 key -{}-></key>
        <key rid='09876'><!-{}- base64 key -{}-></key>
      </keys>
```

```
    </header>
    <payload>
      <!-{}- The payload is normally encrypted and unreadable;
        we show it here in plaintext to illustrate the protocol -{}->
      <envelope xmlns='urn:xmpp:sce:1'>
        <content>
          <private-key xmlns='urn:xmpp:xid:0'
                       xid='0003
                           a107bff3ce10be1d70dd18e74bc09967e4d6309ba50d5f1ddc866412553
                           .internal'>
            000102030405060708090
               a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
          </private-key>
        </content>
        <rpad>...</rpad>
        <from jid='juliet@capulet.lit/phone'/>
      </envelope>
    </payload>
  </encrypted>
</message>
```

8 Server Mapping

A server MAY support XID to and from JID mapping: if it has proven the identity and association between a JID and a XID (e.g., via the Identity Challenge mechanism), and an entity is using a XID in its 'from' attribute, it MUST convert any incoming stanza where the XID is used in the 'from' attribute to the corresponding JID, and it MUST convert any stanza sent to the corresponding entity by replacing the JID used in the 'to' attribute with the corresponding XID. A server that supports this feature MUST advertise it by including the 'urn:xmpp:xid:server-mapping:0' discovery feature in response to a Service Discovery (XEP-0030) [19] information request.

9 Business Rules

- In a serverless environment, if there is no mechanism to handle PEP service without a server available, the XID should be published by any other practical means available in the configuration. How this is done is dependent of the set-up and beyond the scope of this specification.
- The same XID can be associated with one or more DNS-based JIDs. This happens naturally when the same XID is published on the 'urn:xmpp:xid' PEP node of multiple accounts; for example after a server move, or to explicitly link several accounts belonging to the same

[19] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

entity (e.g., work and personal, or several personal accounts on different servers). How a client manages sharing a single XID across multiple accounts is beyond the scope of this specification and is left to client-specific implementation or other XEPs (e.g., Moved (XEP-0283) [20]).

- If Private Key Synchronization is used, the XMPP client SHOULD show a notification indicating that the XID private key has been shared, and SHOULD show which devices it has been shared with (if possible, including information such as the device type (e.g., "phone", "web") and the client used on each device). This allows users to identify which device it is, and detect more easily if it is one of their own, or a suspicious one.
- If Atomically Compare-And-Publish PubSub Items (XEP-0395) [21] is supported by the PEP service, it SHOULD be used to prevent accidental overwriting of the current XID in case of a race condition.
- XID MAY be used in place of DNS-based JIDs to identify users (e.g., for the publisher attribute in Publish-Subscribe (XEP-0060) [22]). That way, if the user changes server (including the pubsub service in this case), the items stay valid.

10 Discovering Support

If a client supports XID, it MUST advertise it by including the 'urn:xmpp:xid:0' discovery feature in response to a Service Discovery (XEP-0030) [23] information request:

Listing 9: Service Discovery information request

```
<iq type='get'
    from='juliet@capulet.lit/balcony'
    to='romeo@montague.lit/orchard'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 10: Service Discovery information response

```
<iq type='result'
    from='romeo@montague.lit/orchard'
    to='juliet@capulet.lit/balcony'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='urn:xmpp:xid:0'/>
    ...
```

[20] XEP-0283: Moved <https://xmpp.org/extensions/xep-0283.html>.
[21] XEP-0395: Atomically Compare-And-Publish PubSub Items <https://xmpp.org/extensions/xep-0395.html>.
[22] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[23] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

```
  </query>
</iq>
```

A server that supports XID to and from JID mapping (see Server Mapping) MUST advertise the 'urn:xmpp:xid:server-mapping:0' discovery feature:

Listing 11: Service Discovery information response from a server with XID mapping support

```
<iq type='result'
    from='capulet.lit'
    to='juliet@capulet.lit/balcony'
    id='disco2'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='urn:xmpp:xid:server-mapping:0'/>
    ...
  </query>
</iq>
```

11 Accessibility Considerations

This specification does not define any user interface elements and therefore has no accessibility considerations.

12 Privacy Considerations

XID is designed to be independent of DNS-based identifiers, which can help reduce correlation by third parties. However, the XID itself is a stable identifier that can be used to track entities across different JIDs and servers. Entities should be aware that publishing their XID (especially with an "open" access model) makes this association publicly discoverable, and could be used for any kind of undesired tracking (advertisement, profiling, etc.).

13 Security Considerations

Many OMEMO implementations use TOFU authentication scheme by default, or similar ones. If user devices have not been manually and correctly verified out of band, there is a risk that a malicious entity has set a compromised key (e.g., if the server is compromised). Among the problems involved (end-to-end encryption compromised), that would mean that Private Key Synchronization would potentially leak the private key to compromised devices. This is a problem of balance between UX and security.
Challenge/response pairs are protected against replay by the nonce mechanism. A verifying

entity MUST NOT accept a response to the same nonce more than once.

14 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA).

15 XMPP Registrar Considerations

The namespaces 'urn:xmpp:xid:0' and 'urn:xmpp:xid:server-mapping:0' should be included in the registry of protocol namespaces (see <https://xmpp.org/registrar/namespaces.html>).
The well-known PEP nodes 'urn:xmpp:xid' and 'urn:xmpp:xid:revoked' should be included in the registry of nodes for service discovery and publish-subscribe (see <https://xmpp.org/registrar/nodes.html>).

15.1 URI Query Types

As authorized by XMPP URI Query Components (XEP-0147) [24], the XMPP Registrar maintains a registry of queries and key-value pairs for use in XMPP URIs (see <https://xmpp.org/registrar/querytypes.html>).
As described below, the registered querytypes for XID private key synchronization are "xid-private" and "xid-created".

15.1.1 xid-private

The registered querytype "xid-private" contains the private key corresponding to the XID, encoded as a hex string (lowercase).

15.1.2 xid-created

The registered querytype "xid-created" contains the creation datetime of the XID, using a DateTime value as specified in XMPP Date and Time Profiles (XEP-0082) [25].

[24] XEP-0147: XMPP URI Query Components <https://xmpp.org/extensions/xep-0147.html>.
[25] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.

16 Acknowledgements

This work has been done for the Serverless and Metadata Reduction for XMPP project. Many thanks to NLNet foundation and NGI Zero Core for funding the work on this specification.

17 XML Schema

TODO