XEP-0494: Client Access Management

Matthew Wild
mailto:mwild1@gmail.com

2024-09-17
Version 0.1.0

Status	Type	Short Name
Experimental	Standards Track	cam

This specification details how an XMPP account owner can view and control which applications and services have access to their account.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

A common feature of secure online services today is the ability for users to monitor and manage what software and services have access to their account. This is especially relevant for XMPP - a diverse ecosystem of software around an interoperable standard can lead to many and various types of applications having access to a user's account.
This specification provides a standard protocol to let a user view and manage what has access to their account.

2 Requirements

- It should be possible for an account owner to obtain a list of applications and entities that are permitted access to their account.
- Where possible, additional details of such access should be provided, to allow informed decisions. This may include things like when access was last used (so that access can removed when no longer used).
- Where possible, an account owner should be able to revoke access so that an application or entity can no longer access their account.

3 Use Cases

3.1 Listing clients

To list clients that have access to the user's account, send a <list/> payload element inside an <iq/> of type 'get':

Listing 1: Client requests list of clients from server

```
<iq id="5468616e6b73" type="get">
  <list xmlns="urn:xmpp:cam:0"/>
</iq>
```

The server will respond with a list of clients:

Listing 2: Client receives list of clients from server

```
<iq id="5468616e6b73" to="user@example.com/UYJKBHKT" type="result"
    xmlns="jabber:client">
  <clients xmlns="urn:xmpp:cam:0">
    <client connected="true" id="zeiP41HLglIu" type="session">
      <first-seen>2023-04-06T14:26:08Z</first-seen>
      <last-seen>2023-04-06T14:37:25Z</last-seen>
      <auth>
```

```
        <password/>
      </auth>
      <permission status=”unrestricted”/>
      <user-agent>
        <software>Gajim</software>
        <uri>https://gajim.org/</uri>
        <device>Juliet’s␣laptop</device>
␣␣␣␣␣␣</user-agent>
␣␣␣␣</client>
␣␣␣␣<client␣connected=”false”␣id=”HjEEr45_LQr”␣type=”access”>
␣␣␣␣␣␣<first-seen>2023-03-27T15:16:09Z</first-seen>
␣␣␣␣␣␣<last-seen>2023-03-27T15:37:24Z</last-seen>
␣␣␣␣␣␣<auth>
␣␣␣␣␣␣␣␣␣␣<grant/>
␣␣␣␣␣␣</auth>
␣␣␣␣␣␣<permission␣status=”normal”/>
␣␣␣␣␣␣<user-agent>
␣␣␣␣␣␣␣␣<software>REST␣client</software>
␣␣␣␣␣␣</user-agent>
␣␣␣␣</client>
␣␣</clients>
</iq>
```

The following attributes are defined on the <client/> tag:

- ’connected’: a boolean that reflects whether this client has an active session on the server (”active” includes connected and sessions that may be disconnected but may yet be reconnected, e.g. using Stream Management (XEP-0198) [1]).
- ’id’: an opaque reference for the client, which can be used to revoke access.
- ’type’: either ”session” if this client is known to have an active or inactive client session on the server, or ”access” if no session has been established (e.g. it may have been granted access to the account, but only used non-XMPP APIs or never logged in).

The <first-seen/> and <last-seen/> elements contain timestamps that reflect when a client was first granted access to the user’s account, and when it most recently used that access. For active sessions, it may reflect the current time or the time of the last login.
The <user-agent/> element contains information about the client software. It may contain any of three optional child elements, each containing text content:

- <software/> - the name of the software
- <uri/> - a URI/URL for the client, such as a homepage
- <device/> - a human-readable identifier/name for the device where the client runs

[1] XEP-0198: Stream Management <https://xmpp.org/extensions/xep-0198.html>.

The <auth/> element MUST be included, and lists the known authentication methods that the client has used to gain access to the account. The following child elements are defined:

- <password/> - the client has presented a valid password
- <grant/> - the client has a valid authorization grant (e.g. via OAuth). The <grant/> element may also contain details of the grant and the associated permissions (described below)
- <fast/> - the client has active FAST tokens

The <auth/> element is explicitly extensible - alternative/future authentication mechanisms may be included under appropriate namespaces.
The <permission/> element MUST also be present, and contains details of the client's level of access to the user's account. The 'status' attribute of the permission element MUST be present and MUST be one of the following values:

- "unrestricted" - the client has full unlimited access to the account.
- "normal" - the client has general access to the account, but some security-relevant features may be restricted (such as managing account access and changing the account password).
- "restricted" - the client has additional restrictions in place. In such a case the details of these restrictions SHOULD be included in an appropriate format (and namespace) within the <permission/> element.

3.2 Revoking access

To revoke a client's access, send a <revoke/> payload element with an 'id' attribute containing one of the client ids fetched from the list:

Listing 3: Client requests revocation of another client's access

```
<iq id="4e4c6e6574" type="set">
  <revoke xmlns="urn:xmpp:cam:0" id="HjEEr45_LQr" />
</iq>
```

The server will respond with an empty result if the revocation succeeds:

Listing 4: Server confirms successful revocation

```
<iq id="4e4c6e6574" type="result" />
```

If the identified client has previously authenticated with a password, there is no way to revoke access except by changing the user's password. If you request revocation of such a client, the server will respond with a 'service-unavailable' error, with the 'password-reset-required' application error:

Listing 5: Server indicates password reset required

```
<iq id="4e4c6e6574" type="error">
  <error type="cancel">
    <service-unavailable xmlns="xmlns='urn:ietf:params:xml:ns:xmpp-
        stanzas'">
    <password-reset-required xmlns="urn:xmpp:cam:0"/>
  </error>
</iq>
```

Changing the user's password can be performed using In-Band Registration (XEP-0077) [2].

4 Accessibility Considerations

This XEP is not deemed to require any additional accessibility considerations beyond those normally required for implementations.

5 Security Considerations

Servers MUST ensure that the provided client listing is an accurate representation of what has access to the user's account. It MUST ensure that the protocol described here is protected from unauthorized access by third parties, to avoid information leaks and denial of service.
In addition to the account owner, implementations MAY provide functionality for server administrators to view and revoke access on behalf of users. For example, if a popular third-party client is discovered to be compromised, an administrator may want to immediately revoke its access to all accounts on their server.

6 Privacy Considerations

This specification provides methods designed to enhance privacy, by allowing revocation of account access when it is no longer needed.
To allow users to make an informed decision, the client listing needs to contain useful information that can help them to identify clients that are suspicious, unwanted, or no longer needed. This may involve the server storing additional information about client sessions

[2] XEP-0077: In-Band Registration <https://xmpp.org/extensions/xep-0077.html>.

than it would otherwise. Implementations MUST make it possible for a deployment to choose whether to keep certain information, and how long for. This ensures that deployments can adapt to their own requirements, the needs and wishes of their users, and the jurisdictions they operate in.

The defaults should aim to strike a balance between privacy and security, and keep client session information for no longer than necessary.

7 IANA Considerations

None.

8 XMPP Registrar Considerations

None.

9 Acknowledgements

Thanks to Kim Alvefur for assisting in the review of this protocol, and the associated prototype implemention. Thanks to NLnet for providing the funding to work on XMPP authentication improvements.