XEP-0390: Entity Capabilities 2.0

Jonas Schäfer
mailto:jonas@wielicki.name
xmpp:jonas@wielicki.name

2020-04-28
Version 0.3.2

Status	Type	Short Name
Deferred	Standards Track	ecaps2

This document overhauls the XMPP protocol extension Entity Capabilities (XEP-0115). It defines an XMPP protocol extension for broadcasting and dynamically discovering client, device, or generic entity capabilities. In order to minimize network impact, the transport mechanism is standard XMPP presence broadcast (thus forestalling the need for polling related to service discovery data), the capabilities information can be cached either within a session or across sessions, and the format has been kept as small as possible.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

XMPP applications often face choices based on the disco#info (see Service Discovery (XEP-0030) [1]) exposed by other entities. For example, for a client, knowledge about whether a roster entry is a Mediated Information eXchange (MIX) (XEP-0369) [2] entity or a normal client is important for user experience. It may also be desirable to provide indicators on the type of client a contact is using (mobile or not).
The canonical way to do so has been issueing Service Discovery (XEP-0030) [3] requests to the entities emitting presence. This, with the evergrowing featureset of XMPP, induces a lot of traffic for all involed parties, especially during startup. This is a waste of resources, as XEP-0030 information rarely changes and even more, common client configurations and versions share exactly the same information.
Entity Capabilities (XEP-0115) [4] has provided the XMPP ecosystem with a way to share this information with less bandwith. Entities using that protocol send a hash of their disco#info result along with presence or stream features. As those hashes can be cached, entities receiving these hashes only need to query the information for each hash once, greatly reducing the Service Discovery traffic.
However, Entity Capabilities (XEP-0115) [5] has two main flaws:

- The hash agility mechanism is underspecified. While it is possible to change the hash function, there is no clearly defined way to send multiple hashes at once to allow for a transition period. Even though it is technically not forbidden to send multiple Entity Capabilities (XEP-0115) [6] <c/> elements with different hashes at once, it is unclear how implementations behave when this happens. Possible issues lie in the use of caps optimization, as well as clients expecting only one <c/> element.
- The algorithm to generate the input for the hash function has flaws as pointed out by Waqas Hussain [7]. Even though these flaws have partially been fixed and worked around, the fundamental problem that the structural information of the individual strings from the disco response is lost persists.

2 Requirements

The *Entity Capabilities 2.0* protocol aims to satisfy the following requirements:

[1] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[2] XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>.
[3] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[4] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[5] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[6] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[7] org.jabber.security Mailing List Archive: '[Security] Trivial preimage attack against the entity capabilities protocol)' from 2009-07-22, <https://mail.jabber.org/pipermail/security/2009-July/000812.html>.

1. Entities must be able to participate even if they support only XMPP Core [8], XMPP IM [9] and Service Discovery (XEP-0030) [10][11].
2. Entities must be able to participate without connectivity to services except their own XMPP server and without connectivity to specialized XMPP services, including cached information from those services.
3. Entities should be able to learn Service Discovery information without actively querying for it.
4. The bandwidth consumption should be as minimal as possible, while reusing existing specifications.
5. It must be possible to write Multi-User Chat (XEP-0045) [12] and Mediated Information eXchange (MIX) (XEP-0369) [13] implementations which can forward this protocol with negligible extra work.
6. Entities must be able to update their published information arbitrarily often in a single presence session.
7. Server infrastructure beyond XMPP Core and XMPP IM must not be required for this to work (but may be beneficial).
8. Entities must be able to be confident that the information obtained from the broadcast is equivalent to the information which would be obtained from querying the generating entity directly at the time the broadcast was generated.
9. The protocol must be able to coexist (but not necessarily exchange information) with Entity Capabilities (XEP-0115) [14].
10. No special XML features beyond what is needed to implement XMPP Core itself should be required.
11. Obsoletion of hash functions should not need a new version of the specification.
12. Support for pushing Entity Capabilities to the clients server without sending presence.

[8] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[9] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[10] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[11] While elements of XEP-0300 are re-used here, full support of XEP-0300 is not formally required to implement this specification.
[12] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[13] XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>.
[14] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

3 Glossary

Capability Hash A tuple of hash function and hash value generated as described in the Hash Function Input section.

Capability Hash Cache A mapping which maps Capability Hashes to disco#info <query/> responses with an empty 'node' attribute.

Cabability Hash Node The name of a Service Discovery (XEP-0030) XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>. 'node' for a given Capability Hash. See Construction of Capability Hash Nodes.

Capability Hash Set A set of Capability Hashes which cover the same Service Discovery (XEP-0030) XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>. response, possibly in the form of a <c/> element with Use of Cryptographic Hash Functions in XMPP (XEP-0300) XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>. <hash/> children.

Generating Entity An entity which emits a Capability Hash Set to other entities.

Processing Entity An entity which receives and processes a Capability Hash Set from a Generating Entity.

Query Interception Server-side processing of disco#info queries directed to a resource based on the Capability Hash Sets published by that resource.

Gratuitous Capabilities The sending of a Capability Hash Set to a server before initial presence has been sent and without being asked by the server.

4 Algorithms

The following algorithms provide data which is sent using this protocol.

4.1 Hash Function Input

The input to this algorithm is a Service Discovery (XEP-0030) [15] disco#info <query/> response. The output is an octet string which can be used as input to a hash function or an error.
General remarks:

- The algorithm strongly distinguishes between character data (sequences of Unicode code points) and octet strings (sequences of 8-bit bytes). Whenever character data is encoded to octet strings in the following algorithm, the UTF-8 as specified in RFC 3629

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

[16] encoding is used. Whenever octet strings are sorted in the following algorithm, the i;octet collation as specified in RFC 4790 [17] is used.

- The algorithm uses the xml:lang attribute. Implementations must take implicit values for the xml:lang attribute into account, for example those inherited from the disco#info, the IQ element, or from the root <stream> tag.

1. If the <query/> element contains any elements except <identity/>, <feature/> (both from the Service Discovery (XEP-0030) [18] disco#info namespace) or Service Discovery Extensions (XEP-0128) [19] data forms, abort with an error.
2. If any Service Discovery Extensions (XEP-0128) [20] <x/> element contains a data form which contains a <reported/> or <item/> element, abort with an error.
3. If any Service Discovery Extensions (XEP-0128) [21] <x/> element does not adhere to the "FORM_TYPE" protocol specified by Field Standardization for Data Forms (XEP-0068) [22], abort with an error.
4. Processing of <feature/> elements:

   a) For each <feature/> element: Encode the character data of the 'var' attribute and append an octet of value 0x1f (ASCII Unit Separator)
   b) Join the resulting octet strings together, ordered from lesser to greater.
   c) Append an octet of value 0x1c (ASCII File Separator).

   The result of this step is referenced as *Features String* later.

5. Processing of <identity/> nodes:

   a) For each <identity/> node:

      i. Encode the character data of the 'category', 'type', 'xml:lang' and 'name' attributes.
      ii. Append an octet of value 0x1f (ASCII Unit Separator) to each resulting octet string.

[16] RFC 3629: UTF-8, a transformation format of ISO 10646 <http://tools.ietf.org/html/rfc3629>.
[17] RFC 4790: Internet Application Protocol Collation Registry <http://tools.ietf.org/html/rfc4790>.
[18] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[19] XEP-0128: Service Discovery Extensions <https://xmpp.org/extensions/xep-0128.html>.
[20] XEP-0128: Service Discovery Extensions <https://xmpp.org/extensions/xep-0128.html>.
[21] XEP-0128: Service Discovery Extensions <https://xmpp.org/extensions/xep-0128.html>.
[22] XEP-0068: Field Data Standardization for Data Forms <https://xmpp.org/extensions/xep-0068.html>.

iii. Join the resulting octet strings together, in the order of 'category', 'type', 'xml:lang' and 'name', resulting in a single octet string for the <identity/> node.

iv. Append an octet of value 0x1e (ASCII Record Separator).

b) Join the resulting octet strings together, ordered from lesser to greater.

c) Append an octet of value 0x1c (ASCII File Separator).

The result of this step is referenced as *Identities String* later.

6. Processing of Service Discovery Extensions (XEP-0128) [23] <x/> elements:

a) For each <x/> element:

i. For each <field/> element:

A. Encode the character data of each <value/> element and append an octet of value 0x1f (ASCII Unit Separator)

B. Join the resulting octet strings together, ordered from lesser to greater.

C. Encode the character data of the 'var' attribute and append an octet of value 0x1f (ASCII Unit Separator) and the result from the previous step.

D. Append an octet of value 0x1e (ASCII Record Separator).

ii. Join the resulting octet strings together, ordered from lesser to greater.

iii. Append an octet of value 0x1d (ASCII Group Separator).

b) Join the resulting octet strings together, ordered from lesser to greater.

c) Append an octet of value 0x1c (ASCII File Separator).

The result of this step is referenced as *Extensions String* later.

7. Join the *Features String*, *Identities String* and *Extensions String* together, in this order. Return the resulting string as result of the algorithm.

4.2 Construction of Capability Hash Sets

The entity picks a set of hash functions it wishes to use. The set of hash functions MUST include at least one hash function which MUST be implemented according to Use of Cryptographic Hash Functions in XMPP (XEP-0300) [24] and SHOULD NOT include any hash functions which MUST NOT be supported according to Use of Cryptographic Hash Functions in XMPP

[23] XEP-0128: Service Discovery Extensions <https://xmpp.org/extensions/xep-0128.html>.
[24] XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>.

(XEP-0300) [25].
Using the algorithm from the previous subsection, the entity calculates the input for the hash functions. It then runs the input through each hash function individually. The resulting tuples of hash algorithm and hash values constitute the *Capability Hash Set*.

4.3 Construction of Capability Hash Nodes

The *Capability Hash Node* is obtained from a *Capability Hash* with the following simple algorithm:

1. To the namespace prefix "urn:xmpp:caps#", append the name of the hash function as per Use of Cryptographic Hash Functions in XMPP (XEP-0300) [26].
2. Append a FULL STOP character (U+002E, ".").
3. Append the Base64 encoded (as specified in RFC 3548 [27]) hash value.

The *Capability Hash Node* can be decomposed into its original components with the following algorithm:

1. Remove the namespace prefix "urn:xmpp:caps#" from the input.
2. From the *end* of the string, start searching for the FULL STOP character (U+002E, ".") separator.
3. Split the string into the hash function and the Base64-encoded hash value at the position found in the previous step.

4.4 Verification of a Capability Hash Set

The algorithm takes a *Capability Hash Set* as input and returns successfully if the hash matches and an error otherwise.

1. Pick a *Capability Hash* from the *Capability Hash Set*.
2. Query the *Generating Entity* for disco#info on the *Capability Hash Node* for the chosen hash like described above. If the entity returns an error, abort with an error.
3. Locally calculate the *Capability Hash* using the same hash function as in the input as described in the algorithm. If the algorithm exits with an error, abort with an error.
4. If the hashes do not match, abort with an error.
5. Exit successfully, the hash is verified.

[25] XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>.
[26] XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>.
[27] RFC 3548: The Base16, Base32, and Base64 Data Encodings <http://tools.ietf.org/html/rfc3548>.

4.5 Examples

The two examples walk through the process of constructing a *Capability Hash Set* for SHA-256 and SHA3-256. The full algorithm for generating the hash function input is explained.

4.5.1 Simple Example

Listing 1: disco#info payload for the simple example; no XEP-0128 forms

```
<query xmlns="http://jabber.org/protocol/disco#info">
  <identity category="client" name="BombusMod" type="mobile"/>
  <feature var="http://jabber.org/protocol/si"/>
  <feature var="http://jabber.org/protocol/bytestreams"/>
  <feature var="http://jabber.org/protocol/chatstates"/>
  <feature var="http://jabber.org/protocol/disco#info"/>
  <feature var="http://jabber.org/protocol/disco#items"/>
  <feature var="urn:xmpp:ping"/>
  <feature var="jabber:iq:time"/>
  <feature var="jabber:iq:privacy"/>
  <feature var="jabber:iq:version"/>
  <feature var="http://jabber.org/protocol/rosterx"/>
  <feature var="urn:xmpp:time"/>
  <feature var="jabber:x:oob"/>
  <feature var="http://jabber.org/protocol/ibb"/>
  <feature var="http://jabber.org/protocol/si/profile/file-transfer"/>
  <feature var="urn:xmpp:receipts"/>
  <feature var="jabber:iq:roster"/>
  <feature var="jabber:iq:last"/>
</query>
```

The data from the example was the first entry in the capsdb [28] hashes subdirectory which had no data forms at the time of writing. The features have been shuffled to show the sorting step in the algorithm.
The algorithm starts by constructing the *Features String*. For this, the values of the 'var' attributes of the feature nodes are encoded as UTF-8 and suffixed with 0x1f (ASCII Unit Separator). The first three of those features are shown as a hexdump below:

```
00000000  68 74 74 70 3a 2f 2f 6a  61 62 62 65 72 2e 6f 72  |http://
   jabber.or|
00000010  67 2f 70 72 6f 74 6f 63  6f 6c 2f 73 69 1f        |g/
   protocol/si.|
0000001e

00000000  68 74 74 70 3a 2f 2f 6a  61 62 62 65 72 2e 6f 72  |http://
   jabber.or|
```

[28] https://github.com/xnyhps/capsdb/

```
00000010  67 2f 70 72 6f 74 6f 63  6f 6c 2f 62 79 74 65 73  |g/
   protocol/bytes|
00000020  74 72 65 61 6d 73 1f                              |treams.|
00000027

00000000  68 74 74 70 3a 2f 2f 6a  61 62 62 65 72 2e 6f 72  |http://
   jabber.or|
00000010  67 2f 70 72 6f 74 6f 63  6f 6c 2f 63 68 61 74 73  |g/
   protocol/chats|
00000020  74 61 74 65 73 1f                                 |tates.|
00000026
```

Note the appended 0x1f octet for each of the three strings. Now the strings are ordered using the i;octet collation and concatenated. The result is suffixed with 0x1c (ASCII File Separator), which gives the following hexdump of the final *Features String*:

```
00000000  68 74 74 70 3a 2f 2f 6a  61 62 62 65 72 2e 6f 72  |http://
   jabber.or|
00000010  67 2f 70 72 6f 74 6f 63  6f 6c 2f 62 79 74 65 73  |g/
   protocol/bytes|
00000020  74 72 65 61 6d 73 1f 68  74 74 70 3a 2f 2f 6a 61  |treams\
   path{.http://ja|}
00000030  62 62 65 72 2e 6f 72 67  2f 70 72 6f 74 6f 63 6f  |bber.org/
   protoco|
00000040  6c 2f 63 68 61 74 73 74  61 74 65 73 1f 68 74 74  |l/
   chatstates.htt|
00000050  70 3a 2f 2f 6a 61 62 62  65 72 2e 6f 72 67 2f 70  |p://
   jabber.org/p|
00000060  72 6f 74 6f 63 6f 6c 2f  64 69 73 63 6f 23 69 6e  |rotocol/
   disco#in|
00000070  66 6f 1f 68 74 74 70 3a  2f 2f 6a 61 62 62 65 72  |fo.http:
   //jabber|
00000080  2e 6f 72 67 2f 70 72 6f  74 6f 63 6f 6c 2f 64 69  |.org/
   protocol/di|
00000090  73 63 6f 23 69 74 65 6d  73 1f 68 74 74 70 3a 2f  |sco#items
   .http:/|
000000a0  2f 6a 61 62 62 65 72 2e  6f 72 67 2f 70 72 6f 74  |/jabber.
   org/prot|
000000b0  6f 63 6f 6c 2f 69 62 62  1f 68 74 74 70 3a 2f 2f  |ocol/ibb.
   http://|
000000c0  6a 61 62 62 65 72 2e 6f  72 67 2f 70 72 6f 74 6f  |jabber.
   org/proto|
000000d0  63 6f 6c 2f 72 6f 73 74  65 72 78 1f 68 74 74 70  |col/
   rosterx.http|
000000e0  3a 2f 2f 6a 61 62 62 65  72 2e 6f 72 67 2f 70 72  |://jabber
   .org/pr|
000000f0  6f 74 6f 63 6f 6c 2f 73  69 1f 68 74 74 70 3a 2f  |otocol/si
   .http:/|
```

```
00000100  2f 6a 61 62 62 65 72 2e  6f 72 67 2f 70 72 6f 74  |/jabber.
   org/prot|
00000110  6f 63 6f 6c 2f 73 69 2f  70 72 6f 66 69 6c 65 2f  |ocol/si/
   profile/|
00000120  66 69 6c 65 2d 74 72 61  6e 73 66 65 72 1f 6a 61  |file-
   transfer.ja|
00000130  62 62 65 72 3a 69 71 3a  6c 61 73 74 1f 6a 61 62  |
   bber:iq:last.jab|
00000140  62 65 72 3a 69 71 3a 70  72 69 76 61 63 79 1f 6a  |
   ber:iq:privacy.j|
00000150  61 62 62 65 72 3a 69 71  3a 72 6f 73 74 65 72 1f  |
   abber:iq:roster.|
00000160  6a 61 62 62 65 72 3a 69  71 3a 74 69 6d 65 1f 6a  |
   jabber:iq:time.j|
00000170  61 62 62 65 72 3a 69 71  3a 76 65 72 73 69 6f 6e  |
   abber:iq:version|
00000180  1f 6a 61 62 62 65 72 3a  78 3a 6f 6f 62 1f 75 72  |.
   jabber:x:oob.ur|
00000190  6e 3a 78 6d 70 70 3a 70  69 6e 67 1f 75 72 6e 3a  |
   n:xmpp:ping.urn:|
000001a0  78 6d 70 70 3a 72 65 63  65 69 70 74 73 1f 75 72  |
   xmpp:receipts.ur|
000001b0  6e 3a 78 6d 70 70 3a 74  69 6d 65 1f 1c           |
   n:xmpp:time..|
000001bd
```

For the *Identities String*, first the character data of the 'category', 'type', 'xml:lang' and 'name' attributes is encoded as UTF-8 and suffixed with 0x1f (ASCII Unit Separator). The resulting individual strings have the following hexdumps:

```
00000000  63 6c 69 65 6e 74 1f                              |client.|
00000007

00000000  6d 6f 62 69 6c 65 1f                              |mobile.|
00000007

00000000  1f                                                |.|
00000001

00000000  42 6f 6d 62 75 73 4d 6f  64 1f                    |BombusMod
   .|
0000000a
```

The strings are now joined together and the result is suffixed with 0x1e (ASCII Record Separator):

```
00000000  63 6c 69 65 6e 74 1f 6d  6f 62 69 6c 65 1f 1f 42  |client.
   mobile..B|
```

```
00000010  6f 6d 62 75 73 4d 6f 64  1f 1e                    |ombusMod
    ..|
0000001a
```

Normally, a sorting step would occur here. As the example only has a single string, the sorting and joining is a no-op. The string is now suffixed with 0x1c (ASCII File Separator) to get the *Identities String*:

```
00000000  63 6c 69 65 6e 74 1f 6d  6f 62 69 6c 65 1f 1f 42  |client.
    mobile..B|
00000010  6f 6d 62 75 73 4d 6f 64  1f 1e 1c                 |ombusMod
    ...|
0000001b
```

The *Extensions String* is simply the 0x1c (ASCII File Separator) used to terminate it as no extensions are contained in the example. Thus, the final input for the hash function is, as hexdump:

```
00000000  68 74 74 70 3a 2f 2f 6a  61 62 62 65 72 2e 6f 72  |http://
    jabber.or|
00000010  67 2f 70 72 6f 74 6f 63  6f 6c 2f 62 79 74 65 73  |g/
    protocol/bytes|
00000020  74 72 65 61 6d 73 1f 68  74 74 70 3a 2f 2f 6a 61  |treams\
    path{.http://ja|}
00000030  62 62 65 72 2e 6f 72 67  2f 70 72 6f 74 6f 63 6f  |bber.org/
    protoco|
00000040  6c 2f 63 68 61 74 73 74  61 74 65 73 1f 68 74 74  |l/
    chatstates.htt|
00000050  70 3a 2f 2f 6a 61 62 62  65 72 2e 6f 72 67 2f 70  |p://
    jabber.org/p|
00000060  72 6f 74 6f 63 6f 6c 2f  64 69 73 63 6f 23 69 6e  |rotocol/
    disco#in|
00000070  66 6f 1f 68 74 74 70 3a  2f 2f 6a 61 62 62 65 72  |fo.http:
    //jabber|
00000080  2e 6f 72 67 2f 70 72 6f  74 6f 63 6f 6c 2f 64 69  |.org/
    protocol/di|
00000090  73 63 6f 23 69 74 65 6d  73 1f 68 74 74 70 3a 2f  |sco#items
    .http:/|
000000a0  2f 6a 61 62 62 65 72 2e  6f 72 67 2f 70 72 6f 74  |/jabber.
    org/prot|
000000b0  6f 63 6f 6c 2f 69 62 62  1f 68 74 74 70 3a 2f 2f  |ocol/ibb.
    http://|
000000c0  6a 61 62 62 65 72 2e 6f  72 67 2f 70 72 6f 74 6f  |jabber.
    org/proto|
000000d0  63 6f 6c 2f 72 6f 73 74  65 72 78 1f 68 74 74 70  |col/
    rosterx.http|
000000e0  3a 2f 2f 6a 61 62 62 65  72 2e 6f 72 67 2f 70 72  |://jabber
    .org/pr|
```

```
000000f0  6f 74 6f 63 6f 6c 2f 73  69 1f 68 74 74 70 3a 2f  |otocol/si
    .http:/|
00000100  2f 6a 61 62 62 65 72 2e  6f 72 67 2f 70 72 6f 74  |/jabber.
    org/prot|
00000110  6f 63 6f 6c 2f 73 69 2f  70 72 6f 66 69 6c 65 2f  |ocol/si/
    profile/|
00000120  66 69 6c 65 2d 74 72 61  6e 73 66 65 72 1f 6a 61  |file-
    transfer.ja|
00000130  62 62 65 72 3a 69 71 3a  6c 61 73 74 1f 6a 61 62  |
    bber:iq:last.jab|
00000140  62 65 72 3a 69 71 3a 70  72 69 76 61 63 79 1f 6a  |
    ber:iq:privacy.j|
00000150  61 62 62 65 72 3a 69 71  3a 72 6f 73 74 65 72 1f  |
    abber:iq:roster.|
00000160  6a 61 62 62 65 72 3a 69  71 3a 74 69 6d 65 1f 6a  |
    jabber:iq:time.j|
00000170  61 62 62 65 72 3a 69 71  3a 76 65 72 73 69 6f 6e  |
    abber:iq:version|
00000180  1f 6a 61 62 62 65 72 3a  78 3a 6f 6f 62 1f 75 72  |.
    jabber:x:oob.ur|
00000190  6e 3a 78 6d 70 70 3a 70  69 6e 67 1f 75 72 6e 3a  |
    n:xmpp:ping.urn:|
000001a0  78 6d 70 70 3a 72 65 63  65 69 70 74 73 1f 75 72  |
    xmpp:receipts.ur|
000001b0  6e 3a 78 6d 70 70 3a 74  69 6d 65 1f 1c 63 6c 69  |
    n:xmpp:time..cli|
000001c0  65 6e 74 1f 6d 6f 62 69  6c 65 1f 1f 42 6f 6d 62  |ent.
    mobile..Bomb|
000001d0  75 73 4d 6f 64 1f 1e 1c  1c                       |usMod
    ....|
000001d9
```

Running this octet string through the hash functions leads as to the following *Capability Hash Set*:

```
<c xmlns="urn:xmpp:caps">
  <hash xmlns="urn:xmpp:hashes:2" algo="sha-256">
     kzBZbkqJ3ADrj7v08reD1qcWUwNGHaidNUgD7nHpiw8=</hash>
  <hash xmlns="urn:xmpp:hashes:2" algo="sha3-256">79
     mdYAfU9rEdTOcWDO7UEAt6E56SUzk/g6TnqUeuD9Q=</hash>
</c>
```

4.5.2 Complex Example

Listing 2: disco#info payload for the complex example with XEP-0128 forms

```
<query xmlns="http://jabber.org/protocol/disco#info">
  <identity category="client" name="Tkabber" type="pc" xml:lang="en"/>
```

```
<identity category="client" name="      " type="pc" xml:lang="ru"/>
<feature var="games:board"/>
<feature var="http://jabber.org/protocol/activity"/>
<feature var="http://jabber.org/protocol/activity+notify"/>
<feature var="http://jabber.org/protocol/bytestreams"/>
<feature var="http://jabber.org/protocol/chatstates"/>
<feature var="http://jabber.org/protocol/commands"/>
<feature var="http://jabber.org/protocol/disco#info"/>
<feature var="http://jabber.org/protocol/disco#items"/>
<feature var="http://jabber.org/protocol/evil"/>
<feature var="http://jabber.org/protocol/feature-neg"/>
<feature var="http://jabber.org/protocol/geoloc"/>
<feature var="http://jabber.org/protocol/geoloc+notify"/>
<feature var="http://jabber.org/protocol/ibb"/>
<feature var="http://jabber.org/protocol/iqibb"/>
<feature var="http://jabber.org/protocol/mood"/>
<feature var="http://jabber.org/protocol/mood+notify"/>
<feature var="http://jabber.org/protocol/rosterx"/>
<feature var="http://jabber.org/protocol/si"/>
<feature var="http://jabber.org/protocol/si/profile/file-transfer"/>
<feature var="http://jabber.org/protocol/tune"/>
<feature var="http://www.facebook.com/xmpp/messages"/>
<feature var="http://www.xmpp.org/extensions/xep-0084.html#ns-
    metadata+notify"/>
<feature var="jabber:iq:avatar"/>
<feature var="jabber:iq:browse"/>
<feature var="jabber:iq:dtcp"/>
<feature var="jabber:iq:filexfer"/>
<feature var="jabber:iq:ibb"/>
<feature var="jabber:iq:inband"/>
<feature var="jabber:iq:jidlink"/>
<feature var="jabber:iq:last"/>
<feature var="jabber:iq:oob"/>
<feature var="jabber:iq:privacy"/>
<feature var="jabber:iq:roster"/>
<feature var="jabber:iq:time"/>
<feature var="jabber:iq:version"/>
<feature var="jabber:x:data"/>
<feature var="jabber:x:event"/>
<feature var="jabber:x:oob"/>
<feature var="urn:xmpp:avatar:metadata+notify"/>
<feature var="urn:xmpp:ping"/>
<feature var="urn:xmpp:receipts"/>
<feature var="urn:xmpp:time"/>
<x xmlns="jabber:x:data" type="result">
  <field type="hidden" var="FORM_TYPE">
    <value>urn:xmpp:dataforms:softwareinfo</value>
  </field>
  <field var="software">
```

```
      <value>Tkabber</value>
    </field>
    <field var="software_version">
      <value>0.11.1-svn-20111216-mod (Tcl/Tk 8.6b2)</value>
    </field>
    <field var="os">
      <value>Windows</value>
    </field>
    <field var="os_version">
      <value>XP</value>
    </field>
  </x>
</query>
```

The data from the example is the shortest entry from the capsdb [29] hashes subdirectory which had data forms and multiple identities at the time of writing. The features have been shuffled to show the sorting step in the algorithm.
We skip over the process for the *Features String* and only present the final result encoded as base64 for reference:

```
Z2FtZXM6Ym9hcmQfaHR0cDovL2phYmJlci5vcmcvcHJvdG9jb2wvYWN0aXZpdHkfaHR0cDovL2ph
YmJlci5vcmcvcHJvdG9jb2wvYWN0aXZpdHkrbm90aWZ5H2h0dHA6Ly9qYWJiZXIub3JnL3Byb3Rv
Y29sL2J5dGVzdHJlYW1zH2h0dHA6Ly9qYWJiZXIub3JnL3Byb3RvY29sL2NoYXRzdGF0ZXMfaHR0
cDovL2phYmJlci5vcmcvcHJvdG9jb2wvY29tbWFuZHMfaHR0cDovL2phYmJlci5vcmcvcHJvdG9j
b2wvZGlzY28jaW5mbx9odHRwOi8vamFiYmVyLm9yZy9wcm90b2NvbC9kaXNjbyNpdGVtcx9odHRw
Oi8vamFiYmVyLm9yZy9wcm90b2NvbC9ldmlsH2h0dHA6Ly9qYWJiZXIub3JnL3Byb3RvY29sL2Zl
YXR1cmUtbmVnH2h0dHA6Ly9qYWJiZXIub3JnL3Byb3RvY29sL2dlb2xvYx9odHRwOi8vamFiYmVy
Lm9yZy9wcm90b2NvbC9nZW9sb2Mrbm90aWZ5H2h0dHA6Ly9qYWJiZXIub3JnL3Byb3RvY29sL2li
Yh9odHRwOi8vamFiYmVyLm9yZy9wcm90b2NvbC9pcWliYh9odHRwOi8vamFiYmVyLm9yZy9wcm90
b2NvbC9tb29kH2h0dHA6Ly9qYWJiZXIub3JnL3Byb3RvY29sL21vb2Qrbm90aWZ5H2h0dHA6Ly9q
YWJiZXIub3JnL3Byb3RvY29sL3Jvc3RlcngfaHR0cDovL2phYmJlci5vcmcvcHJvdG9jb2wvc2kf
aHR0cDovL2phYmJlci5vcmcvcHJvdG9jb2wvc2kvcHJvZmlsZS9maWxlLXRyYW5zZmVyH2h0dHA6
Ly9qYWJiZXIub3JnL3Byb3RvY29sL3R1bmUfaHR0cDovL3d3dy5mYWNlYm9vay5jb20veG1wcC9t
```

[29] https://github.com/xnyhps/capsdb/

```
ZXNzYWdlcx9odHRwOi8vd3d3LnhtcHAub3JnL2V4dGVuc2lvbnMveGVwLTAwODQuaHRtbCNucy1t

ZXRhZGF0YStub3RpZnkfamFiYmVyOmlxOmF2YXRhch9qYWJiZXI6aXE6YnJvd3NlH2phYmJlcjpp

cTpkdGNwH2phYmJlcjppcTpmaWxleGZlch9qYWJiZXI6aXE6aWJiH2phYmJlcjppcTppbmJhbmQf

amFiYmVyOmlxOmppZGxpbmsfamFiYmVyOmlxOmxhc3QfamFiYmVyOmlxOm9vYh9qYWJiZXI6aXE6

cHJpdmFjeR9qYWJiZXI6aXE6cm9zdGVyH2phYmJlcjppcTp0aW1lH2phYmJlcjppcTp2ZXJzaW9u

H2phYmJlcjp4OmRhdGEfamFiYmVyOng6ZXZlbnQfamFiYmVyOng6b29iH3Vybjp4bXBwOmF2YXRh

cjptZXRhZGF0YStub3RpZnkfdXJuOnhtcHA6cGluZx91cm46eG1wcDpyZWNlaXB0cx91cm46eG1w

cDp0aW1lHxw=
```

In the previous example, it was already shown how the individual parts of each <identity/> element are combined. We get the following octet strings as hexdumps:

```
00000000  63 6c 69 65 6e 74 1f 70  63 1f 72 75 1f d0 a2 d0  |client.pc
   .ru....|
00000010  ba d0 b0 d0 b1 d0 b1 d0  b5 d1 80 1f 1e
   |.............|
0000001d

00000000  63 6c 69 65 6e 74 1f 70  63 1f 65 6e 1f 54 6b 61  |client.pc
   .en.Tka|
00000010  62 62 65 72 1f 1e                                 |bber..|
00000016
```

The second string is ordered before the first string in the i;octet collation and afterwards the strings are joined and the result is suffixed with 0x1c (ASCII File Separator) to close the identities part of the input. The final *Identities String* is thus, as hexdump:

```
00000000  63 6c 69 65 6e 74 1f 70  63 1f 65 6e 1f 54 6b 61  |client.pc
   .en.Tka|
00000010  62 62 65 72 1f 1e 63 6c  69 65 6e 74 1f 70 63 1f  |bber..
   client.pc.|
00000020  72 75 1f d0 a2 d0 ba d0  b0 d0 b1 d0 b1 d0 b5 d1  |ru
   ..............|
00000030  80 1f 1e 1c                                       |....|
00000034
```

The example has a Service Discovery Extensions (XEP-0128) [30] form. For each field, a string consisting of the 'var' attributes character data and the values is created as per the algorithm:

[30] XEP-0128: Service Discovery Extensions <https://xmpp.org/extensions/xep-0128.html>.

```
00000000  46 4f 52 4d 5f 54 59 50  45 1f 75 72 6e 3a 78 6d  |FORM_TYPE
    .urn:xm|
00000010  70 70 3a 64 61 74 61 66  6f 72 6d 73 3a 73 6f 66  |
    pp:dataforms:sof|
00000020  74 77 61 72 65 69 6e 66  6f 1f 1e                 |twareinfo
    ..|
0000002b

00000000  73 6f 66 74 77 61 72 65  1f 54 6b 61 62 62 65 72  |software.
    Tkabber|
00000010  1f 1e                                            |..|
00000012

00000000  73 6f 66 74 77 61 72 65  5f 76 65 72 73 69 6f 6e  |
    software_version|
00000010  1f 30 2e 31 31 2e 31 2d  73 76 6e 2d 32 30 31 31  |.0.11.1-
    svn-2011|
00000020  31 32 31 36 2d 6d 6f 64  20 28 54 63 6c 2f 54 6b  |1216-mod
    (Tcl/Tk|
00000030  20 38 2e 36 62 32 29 1f  1e                       | 8.6b2)
    ..|
00000039

00000000  6f 73 1f 57 69 6e 64 6f  77 73 1f 1e              |os.
    Windows..|
0000000c
```

The strings need to be sorted using i;octet and joined together. The result is suffixed with 0x1d (ASCII Group Separator), which closes the form. As this is the only form, the resulting *Extensions String* is obtained by adding a 0x1c (ASCII File Separator) to close the extensions section of the hash input:

```
00000000  46 4f 52 4d 5f 54 59 50  45 1f 75 72 6e 3a 78 6d  |FORM_TYPE
    .urn:xm|
00000010  70 70 3a 64 61 74 61 66  6f 72 6d 73 3a 73 6f 66  |
    pp:dataforms:sof|
00000020  74 77 61 72 65 69 6e 66  6f 1f 1e 6f 73 1f 57 69  |twareinfo
    ..os.Wi|
00000030  6e 64 6f 77 73 1f 1e 6f  73 5f 76 65 72 73 69 6f  |ndows..
    os_versio|
00000040  6e 1f 58 50 1f 1e 73 6f  66 74 77 61 72 65 1f 54  |n.XP..
    software.T|
00000050  6b 61 62 62 65 72 1f 1e  73 6f 66 74 77 61 72 65  |kabber..
    software|
00000060  5f 76 65 72 73 69 6f 6e  1f 30 2e 31 31 2e 31 2d  |_version
    .0.11.1-|
00000070  73 76 6e 2d 32 30 31 31  31 32 31 36 2d 6d 6f 64  |svn
    -20111216-mod|
```

```
00000080  20 28 54 63 6c 2f 54 6b  20 38 2e 36 62 32 29 1f  | (Tcl/Tk
   8.6b2).|
00000090  1e 1d 1c                                          |...|
00000093
```

Note the "os" field is now before the other fields but after "FORM_TYPE", due to the sorting. The final hash function input is obtained by concatenating the *Features String*, *Identities String* and *Extensions String*:

```
00000000  67 61 6d 65 73 3a 62 6f  61 72 64 1f 68 74 74 70  |
   games:board.http|
00000010  3a 2f 2f 6a 61 62 62 65  72 2e 6f 72 67 2f 70 72  |://jabber
   .org/pr|
00000020  6f 74 6f 63 6f 6c 2f 61  63 74 69 76 69 74 79 1f  |otocol/
   activity.|
00000030  68 74 74 70 3a 2f 2f 6a  61 62 62 65 72 2e 6f 72  |http://
   jabber.or|
00000040  67 2f 70 72 6f 74 6f 63  6f 6c 2f 61 63 74 69 76  |g/
   protocol/activ|
00000050  69 74 79 2b 6e 6f 74 69  66 79 1f 68 74 74 70 3a  |ity+
   notify.http:|
00000060  2f 2f 6a 61 62 62 65 72  2e 6f 72 67 2f 70 72 6f  |//jabber.
   org/pro|
00000070  74 6f 63 6f 6c 2f 62 79  74 65 73 74 72 65 61 6d  |tocol/
   bytestream|
00000080  73 1f 68 74 74 70 3a 2f  2f 6a 61 62 62 65 72 2e  |s\path{.
   http://jabber.|}
00000090  6f 72 67 2f 70 72 6f 74  6f 63 6f 6c 2f 63 68 61  |org/
   protocol/cha|
000000a0  74 73 74 61 74 65 73 1f  68 74 74 70 3a 2f 2f 6a  |tstates\
   path{.http://j|}
000000b0  61 62 62 65 72 2e 6f 72  67 2f 70 72 6f 74 6f 63  |abber.org
   /protoc|
000000c0  6f 6c 2f 63 6f 6d 6d 61  6e 64 73 1f 68 74 74 70  |ol/
   commands.http|
000000d0  3a 2f 2f 6a 61 62 62 65  72 2e 6f 72 67 2f 70 72  |://jabber
   .org/pr|
000000e0  6f 74 6f 63 6f 6c 2f 64  69 73 63 6f 23 69 6e 66  |otocol/
   disco#inf|
000000f0  6f 1f 68 74 74 70 3a 2f  2f 6a 61 62 62 65 72 2e  |o.http://
   jabber.|
00000100  6f 72 67 2f 70 72 6f 74  6f 63 6f 6c 2f 64 69 73  |org/
   protocol/dis|
00000110  63 6f 23 69 74 65 6d 73  1f 68 74 74 70 3a 2f 2f  |co#items\
   path{.http://|}
00000120  6a 61 62 62 65 72 2e 6f  72 67 2f 70 72 6f 74 6f  |jabber.
   org/proto|
00000130  63 6f 6c 2f 65 76 69 6c  1f 68 74 74 70 3a 2f 2f  |col/evil.
   http://|
```

```
00000140  6a 61 62 62 65 72 2e 6f  72 67 2f 70 72 6f 74 6f  |jabber.
   org/proto|
00000150  63 6f 6c 2f 66 65 61 74  75 72 65 2d 6e 65 67 1f  |col/
   feature-neg.|
00000160  68 74 74 70 3a 2f 2f 6a  61 62 62 65 72 2e 6f 72  |http://
   jabber.or|
00000170  67 2f 70 72 6f 74 6f 63  6f 6c 2f 67 65 6f 6c 6f  |g/
   protocol/geolo|
00000180  63 1f 68 74 74 70 3a 2f  2f 6a 61 62 62 65 72 2e  |c.http://
   jabber.|
00000190  6f 72 67 2f 70 72 6f 74  6f 63 6f 6c 2f 67 65 6f  |org/
   protocol/geo|
000001a0  6c 6f 63 2b 6e 6f 74 69  66 79 1f 68 74 74 70 3a  |loc+
   notify.http:|
000001b0  2f 2f 6a 61 62 62 65 72  2e 6f 72 67 2f 70 72 6f  |//jabber.
   org/pro|
000001c0  74 6f 63 6f 6c 2f 69 62  62 1f 68 74 74 70 3a 2f  |tocol/ibb
   .http:/|
000001d0  2f 6a 61 62 62 65 72 2e  6f 72 67 2f 70 72 6f 74  |/jabber.
   org/prot|
000001e0  6f 63 6f 6c 2f 69 71 69  62 62 1f 68 74 74 70 3a  |ocol/
   iqibb.http:|
000001f0  2f 2f 6a 61 62 62 65 72  2e 6f 72 67 2f 70 72 6f  |//jabber.
   org/pro|
00000200  74 6f 63 6f 6c 2f 6d 6f  6f 64 1f 68 74 74 70 3a  |tocol/
   mood.http:|
00000210  2f 2f 6a 61 62 62 65 72  2e 6f 72 67 2f 70 72 6f  |//jabber.
   org/pro|
00000220  74 6f 63 6f 6c 2f 6d 6f  6f 64 2b 6e 6f 74 69 66  |tocol/
   mood+notif|
00000230  79 1f 68 74 74 70 3a 2f  2f 6a 61 62 62 65 72 2e  |y.http://
   jabber.|
00000240  6f 72 67 2f 70 72 6f 74  6f 63 6f 6c 2f 72 6f 73  |org/
   protocol/ros|
00000250  74 65 72 78 1f 68 74 74  70 3a 2f 2f 6a 61 62 62  |terx.
   http://jabb|
00000260  65 72 2e 6f 72 67 2f 70  72 6f 74 6f 63 6f 6c 2f  |er.org/
   protocol/|
00000270  73 69 1f 68 74 74 70 3a  2f 2f 6a 61 62 62 65 72  |si.http:
   //jabber|
00000280  2e 6f 72 67 2f 70 72 6f  74 6f 63 6f 6c 2f 73 69  |.org/
   protocol/si|
00000290  2f 70 72 6f 66 69 6c 65  2f 66 69 6c 65 2d 74 72  |/profile/
   file-tr|
000002a0  61 6e 73 66 65 72 1f 68  74 74 70 3a 2f 2f 6a 61  |ansfer.
   http://ja|
000002b0  62 62 65 72 2e 6f 72 67  2f 70 72 6f 74 6f 63 6f  |bber.org/
   protoco|
```

```
000002c0  6c 2f 74 75 6e 65 1f 68  74 74 70 3a 2f 2f 77 77  |l/tune.
   http://ww|
000002d0  77 2e 66 61 63 65 62 6f  6f 6b 2e 63 6f 6d 2f 78  |w.
   facebook.com/x|
000002e0  6d 70 70 2f 6d 65 73 73  61 67 65 73 1f 68 74 74  |mpp/
   messages.htt|
000002f0  70 3a 2f 2f 77 77 77 2e  78 6d 70 70 2e 6f 72 67  |p://www.
   xmpp.org|
00000300  2f 65 78 74 65 6e 73 69  6f 6e 73 2f 78 65 70 2d  |/
   extensions/xep-|
00000310  30 30 38 34 2e 68 74 6d  6c 23 6e 73 2d 6d 65 74  |0084.html
   #ns-met|
00000320  61 64 61 74 61 2b 6e 6f  74 69 66 79 1f 6a 61 62  |adata+
   notify.jab|
00000330  62 65 72 3a 69 71 3a 61  76 61 74 61 72 1f 6a 61  |
   ber:iq:avatar.ja|
00000340  62 62 65 72 3a 69 71 3a  62 72 6f 77 73 65 1f 6a  |
   bber:iq:browse.j|
00000350  61 62 62 65 72 3a 69 71  3a 64 74 63 70 1f 6a 61  |
   abber:iq:dtcp.ja|
00000360  62 62 65 72 3a 69 71 3a  66 69 6c 65 78 66 65 72  |
   bber:iq:filexfer|
00000370  1f 6a 61 62 62 65 72 3a  69 71 3a 69 62 62 1f 6a  |.
   jabber:iq:ibb.j|
00000380  61 62 62 65 72 3a 69 71  3a 69 6e 62 61 6e 64 1f  |
   abber:iq:inband.|
00000390  6a 61 62 62 65 72 3a 69  71 3a 6a 69 64 6c 69 6e  |
   jabber:iq:jidlin|
000003a0  6b 1f 6a 61 62 62 65 72  3a 69 71 3a 6c 61 73 74  |k.
   jabber:iq:last|
000003b0  1f 6a 61 62 62 65 72 3a  69 71 3a 6f 6f 62 1f 6a  |.
   jabber:iq:oob.j|
000003c0  61 62 62 65 72 3a 69 71  3a 70 72 69 76 61 63 79  |
   abber:iq:privacy|
000003d0  1f 6a 61 62 62 65 72 3a  69 71 3a 72 6f 73 74 65  |.
   jabber:iq:roste|
000003e0  72 1f 6a 61 62 62 65 72  3a 69 71 3a 74 69 6d 65  |r.
   jabber:iq:time|
000003f0  1f 6a 61 62 62 65 72 3a  69 71 3a 76 65 72 73 69  |.
   jabber:iq:versi|
00000400  6f 6e 1f 6a 61 62 62 65  72 3a 78 3a 64 61 74 61  |on.
   jabber:x:data|
00000410  1f 6a 61 62 62 65 72 3a  78 3a 65 76 65 6e 74 1f  |.
   jabber:x:event.|
00000420  6a 61 62 62 65 72 3a 78  3a 6f 6f 62 1f 75 72 6e  |
   jabber:x:oob.urn|
00000430  3a 78 6d 70 70 3a 61 76  61 74 61 72 3a 6d 65 74  |
   :xmpp:avatar:met|
```

```
00000440  61 64 61 74 61 2b 6e 6f  74 69 66 79 1f 75 72 6e  |adata+
    notify.urn|
00000450  3a 78 6d 70 70 3a 70 69  6e 67 1f 75 72 6e 3a 78  |
    :xmpp:ping.urn:x|
00000460  6d 70 70 3a 72 65 63 65  69 70 74 73 1f 75 72 6e  |
    mpp:receipts.urn|
00000470  3a 78 6d 70 70 3a 74 69  6d 65 1f 1c 63 6c 69 65  |
    :xmpp:time..clie|
00000480  6e 74 1f 70 63 1f 65 6e  1f 54 6b 61 62 62 65 72  |nt.pc.en.
    Tkabber|
00000490  1f 1e 63 6c 69 65 6e 74  1f 70 63 1f 72 75 1f d0  |..client.
    pc.ru..|
000004a0  a2 d0 ba d0 b0 d0 b1 d0  b1 d0 b5 d1 80 1f 1e 1c
    |................|
000004b0  46 4f 52 4d 5f 54 59 50  45 1f 75 72 6e 3a 78 6d  |FORM_TYPE
    .urn:xm|
000004c0  70 70 3a 64 61 74 61 66  6f 72 6d 73 3a 73 6f 66  |
    pp:dataforms:sof|
000004d0  74 77 61 72 65 69 6e 66  6f 1f 1e 6f 73 1f 57 69  |twareinfo
    ..os.Wi|
000004e0  6e 64 6f 77 73 1f 1e 6f  73 5f 76 65 72 73 69 6f  |ndows..
    os_versio|
000004f0  6e 1f 58 50 1f 1e 73 6f  66 74 77 61 72 65 1f 54  |n.XP..
    software.T|
00000500  6b 61 62 62 65 72 1f 1e  73 6f 66 74 77 61 72 65  |kabber..
    software|
00000510  5f 76 65 72 73 69 6f 6e  1f 30 2e 31 31 2e 31 2d  |_version
    .0.11.1-|
00000520  73 76 6e 2d 32 30 31 31  31 32 31 36 2d 6d 6f 64  |svn
    -20111216-mod|
00000530  20 28 54 63 6c 2f 54 6b  20 38 2e 36 62 32 29 1f  | (Tcl/Tk
    8.6b2).|
00000540  1e 1d 1c                                          |...|
00000543
```

Feeding the concatenated octet string as input to the hash functions yields the following *Capability Hash Set:*

```
<c xmlns="urn:xmpp:caps">
  <hash xmlns="urn:xmpp:hashes:2" algo="sha-256">
     u79ZroNJbdSWhdSp311mddz44oHHPsEBntQ5b1jqBSY=</hash>
  <hash xmlns="urn:xmpp:hashes:2" algo="sha3-256">
     XpUJzLAc93258sMECZ3FJpebkzuyNXDzRNwQog8eycg=</hash>
</c>
```

5 Use Cases

5.1 Advertising Support

If an entity supports *Entity Capabilities 2.0*, it MUST advertise the fact by returning a feature of "urn:xmpp:caps".

Listing 3: Response to a disco#info request

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

5.2 Advertisement of Support and Capabilities by Servers

A server MAY advertise its support for this protocol as well as the current hashes in the stream features.

Listing 4: Stream Features of a server

```
<stream:features>
  ...
  <c xmlns="urn:xmpp:caps">
    <hash xmlns="urn:xmpp:hashes:2" algo="sha-256">
       K1Njy3HZBThlo4moOD5gBGhn0U0oK7/CbfLlIUDi6o4=</hash>
    <hash xmlns="urn:xmpp:hashes:2" algo="sha3-256">+sDTQqBmX6iG/
       X3zjt06fjZMBBqL/723knFIyRf0sg8=</hash>
  </c>
  ...
</stream:features>
```

When a connected client or peer server sends a service discovery information request to determine the entity capabilities of a server that advertises capabilities via the stream feature, the requesting entity MUST send the disco#info request to the server's JID as provided in the 'from' attribute of the response stream header. To enable this functionality, a server that advertises support for entity capabilities MUST provide a 'from' address in its response stream headers, in accordance with RFC 6120 [31].

[31] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

5.3 Advertising Support of Caps Optimizations

If a server supports Caps Optimizations, it MUST advertise the fact by returning a feature of "urn:xmpp:caps:optimize".

Listing 5: Response to a disco#info request

```
<iq from='montague.lit'
    id='disco2'
    to='romeo@montague.lit/chamber'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='urn:xmpp:caps'/>
    <feature var='urn:xmpp:caps:optimize'/>
    ...
  </query>
</iq>
```

5.4 Broadcasting Entity Capabilities

An entity publishes the current *Capability Hash Set* in presence stanzas it sends:

Listing 6: Presence broadcast with hashes

```
<presence from='juliet@capulet.lit'>
  <c xmlns="urn:xmpp:caps">
    <hash xmlns="urn:xmpp:hashes:2" algo="sha-256">
       u79ZroNJbdSWhdSp311mddz44oHHPsEBntQ5b1jqBSY=</hash>
    <hash xmlns="urn:xmpp:hashes:2" algo="sha3-256">
       XpUJzLAc93258sMECZ3FJpebkzuyNXDzRNwQog8eycg=</hash>
  </c>
</presence>
```

The <hash/> element is specified by Use of Cryptographic Hash Functions in XMPP (XEP-0300) [32] and is used to transport the *Capability Hashes*.

5.5 Service Discovery Query for a Specific Hash Value

To query the Service Discovery (XEP-0030) [33] information for a specific *Capability Hash* value, an entity MUST query a Service Discovery node equal to the *Capability Hash Node*[34].
An entity is free to choose for which *Capability Hash* of a *Capability Hash Set* the request is sent.

[32] XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>.
[33] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[34] As outlined in the Business Rules, this statement does not oblige an entity to actually perform this query.

Listing 7: Service Discovery request in response to a broadcast Capability Hash Set

```
<presence from='juliet@capulet.lit/chamber' to='romeo@montague.lit/
    orchard'>
  <c xmlns="urn:xmpp:caps">
    <hash xmlns="urn:xmpp:hashes:2" algo="sha-256">
       u79ZroNJbdSWhdSp311mddz44oHHPsEBntQ5b1jqBSY=</hash>
    <hash xmlns="urn:xmpp:hashes:2" algo="sha3-256">
       XpUJzLAc93258sMECZ3FJpebkzuyNXDzRNwQog8eycg=</hash>
  </c>
</presence>

<iq from='romeo@montague.lit/orchard'
    id='disco3'
    to='juliet@capulet.lit/chamber'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'
         node='urn:xmpp:caps#sha-256.
            u79ZroNJbdSWhdSp311mddz44oHHPsEBntQ5b1jqBSY='/>
</iq>

<iq from='juliet@capulet.lit/chamber'
    id='disco3'
    to='romeo@montague.lit/orchard'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'
         node='urn:xmpp:caps#sha-256.
            u79ZroNJbdSWhdSp311mddz44oHHPsEBntQ5b1jqBSY='>
    <identity category="client" name="Tkabber" type="pc" xml:lang="en"
       />
    <identity category="client" name="      " type="pc" xml:lang="ru"/
       >
    <feature var="games:board"/>
    <feature var="http://jabber.org/protocol/activity"/>
    <feature var="http://jabber.org/protocol/activity+notify"/>
    <feature var="http://jabber.org/protocol/bytestreams"/>
    <feature var="http://jabber.org/protocol/chatstates"/>
    <feature var="http://jabber.org/protocol/commands"/>
    <feature var="http://jabber.org/protocol/disco#info"/>
    <feature var="http://jabber.org/protocol/disco#items"/>
    <feature var="http://jabber.org/protocol/evil"/>
    <feature var="http://jabber.org/protocol/feature-neg"/>
    <feature var="http://jabber.org/protocol/geoloc"/>
    <feature var="http://jabber.org/protocol/geoloc+notify"/>
    <feature var="http://jabber.org/protocol/ibb"/>
    <feature var="http://jabber.org/protocol/iqibb"/>
    <feature var="http://jabber.org/protocol/mood"/>
    <feature var="http://jabber.org/protocol/mood+notify"/>
    <feature var="http://jabber.org/protocol/rosterx"/>
    <feature var="http://jabber.org/protocol/si"/>
```

```
    <feature var="http://jabber.org/protocol/si/profile/file-transfer"
       />
    <feature var="http://jabber.org/protocol/tune"/>
    <feature var="http://www.facebook.com/xmpp/messages"/>
    <feature var="http://www.xmpp.org/extensions/xep-0084.html#ns-
       metadata+notify"/>
    <feature var="jabber:iq:avatar"/>
    <feature var="jabber:iq:browse"/>
    <feature var="jabber:iq:dtcp"/>
    <feature var="jabber:iq:filexfer"/>
    <feature var="jabber:iq:ibb"/>
    <feature var="jabber:iq:inband"/>
    <feature var="jabber:iq:jidlink"/>
    <feature var="jabber:iq:last"/>
    <feature var="jabber:iq:oob"/>
    <feature var="jabber:iq:privacy"/>
    <feature var="jabber:iq:roster"/>
    <feature var="jabber:iq:time"/>
    <feature var="jabber:iq:version"/>
    <feature var="jabber:x:data"/>
    <feature var="jabber:x:event"/>
    <feature var="jabber:x:oob"/>
    <feature var="urn:xmpp:avatar:metadata+notify"/>
    <feature var="urn:xmpp:ping"/>
    <feature var="urn:xmpp:receipts"/>
    <feature var="urn:xmpp:time"/>
    <x xmlns="jabber:x:data" type="result">
      <field type="hidden" var="FORM_TYPE">
        <value>urn:xmpp:dataforms:softwareinfo</value>
      </field>
      <field var="software">
        <value>Tkabber</value>
      </field>
      <field var="software_version">
        <value>0.11.1-svn-20111216-mod (Tcl/Tk 8.6b2)</value>
      </field>
      <field var="os">
        <value>Windows</value>
      </field>
      <field var="os_version">
        <value>XP</value>
      </field>
    </x>
  </query>
</iq>
```

5.6 Gratuitous Capabilities

A server MAY support pushing of *Capability Hashes* from clients before sending initial presence. This allows servers to discover capabilities of clients before those have sent initial presence, which may be useful or important for some protocols (such as Mediated Information eXchange (MIX) (XEP-0369) [35]). This feature is called *Gratuitous Capabilities*.
To advertise support, the server publishes the urn:xmpp:caps:gratuitous feature:

Listing 8: Response to a disco#info request if the server supports Gratuitous Capabilities

```
<iq from='montague.lit'
    id='disco3'
    to='romeo@montague.lit/chamber'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='urn:xmpp:caps'/>
    <feature var='urn:xmpp:caps:gratuitous'/>
    ...
  </query>
</iq>
```

After determining server support, a client can send *Capability Hashes* via *Gratuitous Capabilities* before sending initial presence:

Listing 9: Sending Gratuitous Capabilities

```
<iq from='romeo@montague.lit/chamber'
    to='montague.lit'
    id='grat1'
    type='set'>
  <c xmlns="urn:xmpp:caps">
    <hash xmlns="urn:xmpp:hashes:2" algo="sha-256">
       u79ZroNJbdSWhdSp311mddz44oHHPsEBntQ5b1jqBSY=</hash>
    <hash xmlns="urn:xmpp:hashes:2" algo="sha3-256">
       XpUJzLAc93258sMECZ3FJpebkzuyNXDzRNwQog8eycg=</hash>
  </c>
</iq>

<iq from='montague.lit'
    to='romeo@montague.lit/chamber'
    id='grat1'
    type='result'>
</iq>
```

The server replies with an empty result on success.
The server MUST NOT broadcast the *Capability Hashes* submitted via *Gratuitous Capabilities*

[35] XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>.

using presence.
Clients SHOULD NOT send *Gratuitous Capabilities* after they have sent initial presence; instead, they SHOULD re-send presence to update the *Capability Hashes*. Otherwise, entities subscribed to the presence will not receive the updated *Capability Hashes*.

6 Business Rules

6.1 Rules for Generating Entities

- Entities MUST respond to disco#info queries for all *Capability Hash Nodes* of at least the most recent 3 *Capability Hash Sets* emitted.
- Entities MUST broadcast the *Capability Hash Set* of the current disco#info it publishes in every non-directed "available" <presence/> they send and SHOULD do so for directed "available" <presence/>.
- After initial presence has been sent, entities MUST re-broadcast the *Capability Hash Set* after their disco#info response changes, but MAY limit the rate at which presences are emitted solely for the purpose of sending new *Capability Hash Sets*.
- Before initial presence has been sent and if the server supports *Gratuitous Capabilities*, entities SHOULD send *Gratuitous Capabilities* after their disco#info response changes, but MAY limit the rate at which *Gratuitous Capabilities* are sent. (For example, a client may load and enable additional functionality (thus changing its features) based on server support and only send *Gratuitous Capabilities* once all functionality has been set up, not after each individual feature.)
- Entities MAY assume that another entity supports *Entity Capabilities 2.0* after receiving a *Capability Hash Set* from that entity.
- Entities MAY also send Entity Capabilities (XEP-0115) [36] capabilities to support legacy entities.

6.2 Rules for Processing Entities

- Entities MAY limit the rate at which they process incoming *Capability Hash Sets*.
- Entities MUST be able to process *Capability Hash Nodes* which use a hash function whose name includes the FULL STOP character (U+002E, ".").
- Entities MAY verify incoming *Capability Hash Sets*.
- Entities MUST NOT expect to receive *Capability Hash Sets* on every presence sent by an entity supporting *Entity Capabilities 2.0*.

[36] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

6.2.1 Caching

A *Capability Hash* MAY be stored alongside with its disco#info in a *Capability Hash Cache*. A received *Capability Hash* which has not been verified MUST NOT be stored.
Instead of issuing a Service Discovery (XEP-0030) [37] disco#info <query/> with absent 'node' attribute to a target entity, an entity MAY use a *Capability Hash Cache* to obtain the response.
To look up the disco#info response in the *Capability Hash Cache*, an entity MUST use a hash from the *Capability Hash Set* which was most recently received from the entity to which the <query/> would have been sent otherwise. If none of the most recently received *Capability Hashes* are found in the *Capability Hash Cache*, the entity MUST fall back to sending the request.
An entity MUST NOT use *Capability Hashes* which were not included in the most recent *Capability Hash Set* received from the target entity.
An entity MAY use external data sources to fill the *Capability Hash Cache*.
An entity MUST ensure that implicit values for xml:lang attributes is preserved when disco#info data is cached. This can for example happen by making the implicit values explicit in the storage.

6.3 Additional Rules for Clients and Servers implementing Caps Optimizations

- Servers MAY strip off the <c/> element if it has not changed since the previous presence broadcast.
- Servers MUST ensure that the first presence notification sent to each subscriber contains the most recent <c/> element, if any were sent in the current presence session.
- Servers MUST ensure that every change in the <c/> element is sent to all subscribers.
- Clients MAY omit the <c/> element if it has not changed since the last presence *iff* they determined that their server supports Caps Optimization.
- Servers MAY answer disco#info requests for *Capability Hash Nodes* on behalf of their and others clients if the disco#info response belonging to that *Capability Hash* is known to them.

6.4 Query Interception

Servers MAY implement *Query Interception* to further optimise bandwidth consumption. The idea is that servers intercept Service Discovery (XEP-0030) [38] disco#info queries sent to clients if they already know the answer from *Capability Hashes* published by the client. The rules for *Query Interception* are the following (to be applied in this order):

- Servers MUST NOT intercept disco#info queries except those with empty node or a node which refers to a *Capability Hash Node* known to the server.

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

- Servers MUST NOT intercept disco#info queries on behalf of the resource unless the query would be forwarded to the resource otherwise.
- Servers MUST NOT intercept disco#info queries to resources which do not support *Entity Capabilities 2.0* (clients not implementing *Entity Capabilities 2.0* may legitimately use disco#info nodes matching the format of *Capability Hash Nodes* for different purposes).
- Servers SHOULD intercept disco#info queries with empty node and answer them with the disco#info of the most recent *Capability Hash Set* published by the client.
- Servers SHOULD intercept disco#info queries a valid *Capability Hash Node* node, if the server knows the disco#info for the *Capability Hash Node*. Otherwise, the query MUST be forwarded to the addressed resource. Note that it is valid for a sevrer to reply for *Capability Hash Nodes* which have not been published by the resource.

7 Implementation Notes

7.1 Caching

It is RECOMMENDED that entities use the caching mechanisms outlined in the Caching Business Rules. Entities MAY share caches among connections and accounts.

7.2 Upgrading from XEP-0115

Generating Entities are encouraged to also emit Entity Capabilities (XEP-0115) [39] <c/> elements in their presence updates (as specified in Entity Capabilities (XEP-0115) [40]) for a reasonable transition period.
When receiving a *Capability Hash Set* along with Entity Capabilities (XEP-0115) [41] capabilities, a *Processing Entity* MAY obtain the disco#info <query/> for verification from a XEP-0115 based cache instead of querying the *Generating Entity* directly. A *Processing Entity* MUST NOT use disco#info data from a XEP-0115 cache without verification if a *Entity Capabilities 2.0* <c/> element is available.

8 Security Considerations

8.1 Hash Function Input Data Separators

The codepoints used for separating the different parts in the Hash Function Input Algortihm (0x1c (ASCII File Separator) through 0x1f (ASCII Unit Separator)) are not allowed in well-

[39] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[40] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[41] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

formed XML 1.0 character data[42]. As entities are, per XMPP Core [43], required to close a stream if non-well-formed XML 1.0 data is received, these codepoints cannot occur in the input to the algorithm and their use as separators is safe.

8.2 Caching

If the algorithm for constructing the input to the hash function or the used hash function itself allow for cheap collisions, caching the hashes will become dangerous as it allows for cache poisoning. This in turn allows entities to effectively fake disco#info responses of other entities.
This was an issue with Entity Capabilities (XEP-0115) [44] and has been addressed with a new algorithm for generating the hash function input which keeps the structural information of the disco#info input.
An entity MUST NOT ever use disco#info which has not been verified to belong to a *Capability Hash* obtained from a cache using that *Capability Hash*. Using cache contents from a trusted source (at the discretion of the entity) counts as verifying.
A malicious entity could send a large amount of *Capability Hash Sets* in short intervals, while making sure that it provides matching disco#info responses. If a *Processing Entity* uses caching, this can overflow or thrash the caches. *Processing Entities* should be aware of this risk and apply proper rate-limiting for processing *Capability Hash Sets*. To reduce the attack surface, an entity MAY choose to not cache *Capability Hashes* obtained from entities not in its roster.
As mentioned earlier, when storing disco#info data in a cache for later retrieval, implementations MUST ensure that implicit values for xml:lang attributes are reconstructed correctly when the disco#info is restored.

8.3 Directed Presence

Entities MAY choose to not send *Capability Hash Sets* with directed presence (for example to increase privacy). In that case, entities SHOULD also refuse direct Service Discovery (XEP-0030) [45] queries.

8.4 Query Interception

The server replies to certain disco#info queries on behalf of the client. This means that the client has no choice on to whom they reply. Otherwise, a client could choose to reply with <service-unavailable/> to mask its existence. We consider two effects of this:

[42] Note that the "ASCII Seperators" codepoints would be valid, although discouraged, characters of XML 1.1, but XMPP mandates XML 1.0.
[43] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[44] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[45] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

- A remote entity could attempt to detect that an entity exists behind a resource. For this, they send a disco#info query to the resource since nearly everyone implements disco#info. As the client responds with <service-unavailable/>, it looks as if no client was present at this resource.
  With *Query Interception*, the server would reply on behalf of the client. However, the consensus in the community is that by measuring the difference between the reply from the server of the resource and the reply from the actual resource, it would generally be possible to detect the existence of a resource.

- A remote entity can obtain the disco#info information of any resource which supports *Entity Capabilities 2.0* and of which the entity knows the resource.
  This cannot be mitigated with *Query Interception*. The risk is deemed acceptable considering that resources should generally be chosen randomly.

9 Design Considerations

The following alternatives to the custom algorithm were considered and eventually rejected:

9.1 Canonical XML

A common way to canonicalize XML which could be used is Canonical XML [46]. It was decided not to use Canonical XML for the following reasons:

- Implementing it is quite some effort and not all XML libraries come with an implementation.
- It is sensitive to the relative ordering of the elements. The relative ordering of children in disco#info <query/> elements, however, does not matter.
- Several children of Service Discovery Extensions (XEP-0128) [47] data forms are deliberately ignored, like instructions and other descriptive text. The descriptive text is not relevant for the information is being conveyed.

Thus, using Canonical XML would require additional, non-trivial software support and still require non-trivial additional canonicalization rules.

[46] Canonical XML 1.0 <http://www.w3.org/TR/xml-c14n>.
[47] XEP-0128: Service Discovery Extensions <https://xmpp.org/extensions/xep-0128.html>.

10 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [48].

11 XMPP Registrar Considerations

11.1 Protocol Namespaces

The XMPP Registrar [49] includes "urn:xmpp:caps" in its registry of protocol namespaces (see <https://xmpp.org/registrar/namespaces.html>).

```
<ns>
  <name>urn:xmpp:caps</name>
  <doc>&xep0390;</doc>
</ns>
```

11.2 Service Discovery Features

The XMPP Registrar includes "urn:xmpp:caps" and "urn:xmpp:caps:optimize" in its registry of service discovery features (see <https://xmpp.org/registrar/disco-features.html>).

```
<var>
  <name>urn:xmpp:caps</name>
  <desc>Indicate support for Entity Capabilities 2.0</desc>
  <doc>&xep0390;</doc>
</var>
<var>
  <name>urn:xmpp:caps:optimize</name>
  <desc>Indicate support for optimisation of Entity Capabilities 2.0
     broadcast.</desc>
  <doc>&xep0390;</doc>
</var>
```

11.3 Stream Features

The XMPP Registrar includes "urn:xmpp:caps" in its registry of stream features (see <https://xmpp.org/registrar/stream-features.html>).

[48] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

[49] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.

```
<feature>
  <ns>urn:xmpp:caps</ns>
  <name>ecaps2</name>
  <element>c</element>
  <desc>Indicate support for Entity Capabilities 2.0 and publish
     capabilities to peer.</desc>
  <doc>&xep0390;</doc>
</feature>
```

12 XML Schema

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='urn:xmpp:caps'
  xmlns='urn:xmpp:caps'
  elementFormDefault='qualified'
  xmlns:hashes='urn:xmpp:hashes:2'>

  <!-- FIXME: import of XEP-0300 schema, which isn't␣at␣https://xmpp.
     org/schemas/␣at␣the␣time␣of␣writing␣-->

␣␣<xs:annotation>
␣␣␣␣<xs:documentation>
␣␣␣␣␣␣The␣protocol␣documented␣by␣this␣schema␣is␣defined␣in␣XEP-0390:
␣␣␣␣␣␣http://www.xmpp.org/extensions/xep-0390.html
␣␣␣␣</xs:documentation>
␣␣</xs:annotation>

␣␣<xs:element␣name='c'>
␣␣␣␣<xs:complexType>
␣␣␣␣␣␣<xs:sequence␣minOccurs='1'>
␣␣␣␣␣␣␣␣<xs:element␣ref='hashes:hash'␣minOccurs='1'␣maxOccurs='
    unbounded'/>
␣␣␣␣␣␣</xs:sequence>
␣␣␣␣</xs:complexType>
␣␣</xs:element>

</xs:schema>
```

13 Acknowledgements

Thanks to the authors of Entity Capabilities (XEP-0115) [50] for coming up with the original idea of using presence broadcast to convey service discovery information, as well as the optimization strategies.
The note below the example in Advertisement of Support and Capabilities by Servers has been copied verbatimly from Entity Capabilities (XEP-0115) [51].
Thanks to Waqas Hussain for originally (to my knowledge) pointing out the security flaws in Entity Capabilities (XEP-0115) [52] (see [53]).
Thanks to Dave Cridland, Georg Lukas, Link Mauve, Sebastian Riese, Florian Schmaus and Sam Whited for their input, editorial and otherwise.

[50] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[51] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[52] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[53] org.jabber.security Mailing List Archive: '[Security] Trivial preimage attack against the entity capabilities protocol)' from 2009-07-22, <https://mail.jabber.org/pipermail/security/2009-July/000812.html>.