# XEP-0389: Extensible In-Band Registration

Sam Whited
mailto:sam@samwhited.com
xmpp:sam@samwhited.com
https://blog.samwhited.com/

2017-03-16
Version 0.1.0

| Status | Type | Short Name |
|---|---|---|
| Experimental | Standards Track | ibr2 |

This specification defines an XMPP protocol extension for in-band registration with instant messaging servers and other services with which an XMPP entity may initiate a stream. It aims to improve upon the state of the art and replace XEP-0077: In-Band Registration by allowing multi-factor registration mechanisms, and account recovery.

# Legal

## Copyright



## Permissions



## Warranty



## Liability



## Conformance

## Contents

# 1 Introduction

Historically, registering with an XMPP service has been difficult. Each server either used customized out-of-band registration mechanisms such as web forms which were difficult to discover, or they used In-Band Registration (XEP-0077) [1] which could easily be abused by spammers to register large numbers of accounts and which allowed for only limited extensibility.
To solve these issues this specification provides a new in-band registration protocol that allows servers to present the user with a series of "challenges". This allows for both multi-stage proof-of-posession registration flows and spam prevention mechanisms such as proof-of-work functions.

# 2 Requirements

- The server MUST be able to present multiple challenges to the client.
- The server SHOULD be able reduce account registration spam.
- The server MAY present a challenge that requires the user to complete a step out-of-band.
- A client SHOULD be able to register an account without requiring the user to leave the client.
- A client MUST be able to use the same mechanism to register an account and to recover a forgotten password (subject to server policy).

# 3 Glossary

**Proof-of-work (PoW)** A proof-of-work protocol requires that a client perform a computationally intense task which is easily verified by the server.

**Proof-of-possession (PoP)** A proof-of-possession protocol requires that a client prove that they have posession of some resource (eg. a shared secret, or a valid mobile phone number).

# 4 Use Cases

- As a server operator, I want to prevent individual spammers from registering many accounts so I require registrants to perform a proof-of-work function before registration is completed.

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

- As a server operator I want to prevent zombie machines from registering for accounts so I require that registrants submit a form which requires user interaction.
- As a user I do not want to lose access to my account if I forget my password, so I provide my email and telephone number in response to the servers data form.
- As a server operator I do not want users to accidentally add an incorrect recovery address so I send an email with a unique link to the indicated account and require that they click the link before registration can continue.
- As a server operator I want to prevent SPIM using a proof-of-posession protocol so I present the user with a form asking for a mobile phone number and then send a verification code to that number via SMS and show another form requesting the verification code.

## 5 Discovering Support

If a server supports registering for or recovering an account using Extensible IBR, it MUST inform the connecting client when returning stream features during the stream negotiation process. This is done by including a <register/> element, qualified by the 'urn:xmpp:register:0' namespace for account registration, or a <recovery/> element qualified by the same namespace for account recovery. The register and recovery features are always voluntary-to-negotiate. The registration and recovery features MUST NOT be advertised before encryption has been negotiated, eg. using direct-TLS or STARTTLS. They SHOULD be advertised at the same time as the SASL authentication feature, meaning that after registration or recovery is completed SASL authentication can proceed.
For recovery or registration, the server MUST include a list of all challenge types which the client may receive during the course of registering or recovering an account. The purpose of this list is to allow clients to detect if registration requires a challenge type which the client does not support, so servers SHOULD only include each type once; the list is merely informative, and should not be relied upon by clients except to ensure that all mechanisms are supported. This list should comprise <challenge/> elements containing a string that uniquely identifies the type of challenge being issued.

Listing 1: Host Advertises Stream Features

```
<stream:features>
  <mechanisms xmlns='urn:xmpp:sasl:0'>
    <mechanism>EXTERNAL</mechanism>
    <mechanism>SCRAM-SHA-1-PLUS</mechanism>
    <mechanism>SCRAM-SHA-1</mechanism>
    <mechanism>PLAIN</mechanism>
  </mechanisms>
  <register xmlns='urn:xmpp:register:0'>
    <challenge>jabber:x:data'</challenge>
␣␣␣␣<challenge>pow-example</challenge>
```

```
␣␣</register>
␣␣<recovery␣xmlns='urn:xmpp:register:0'>
␣␣␣␣<challenge>jabber:x:oob</challenge>
␣␣</recovery>
</stream:features>
```

## 6 Challenges

A client selects the registration or recovery feature for negotiation by replying with an empty element of the same name and namespace. For example, to attempt account recovery the client would send a <recovery> element qualified by the 'urn:xmpp:register:0' namespace.
The server then replies with a challenge. Challenges take the form of a <challenge/> element qualified by the 'urn:xmpp:register:0' namespace with a 'type' attribute containing the challenge type and containing a challenge data payload.
Type type of a challenge is a value which identifes what sort of payload a client might expect. This document defines a type of 'jabber:x:data' which MUST always contain a data form (an 'x' element with type 'form') as defined by Data Forms (XEP-0004) [2]. Other types may be defined in the future. For example, a challenge containing a data form might look like the following:

Listing 2: Host Returns Registration Form to Entity

```
<challenge xmlns='urn:xmpp:register:0'
           type='jabber:x:data'>
  <x xmlns='jabber:x:data' type='form'>
    <title>Chat Registration</title>
    <instructions>
      Please provide the following information
      to sign up to view our chat rooms!
    </instructions>
    <field type='hidden' var='FORM_TYPE'>
      <value>urn:xmpp:register:0</value>
    </field>
    <field type='text-single' label='Given␣Name' var='first'/>
    <field type='text-single' label='Family␣Name' var='last'/>
    <field type='text-single' label='Nickname' var='nick'>
      <required/>
    </field>
    <field type='text-single' label='Recovery␣Email␣Address' var='
       email'>
      <required/>
    </field>
  </x>
</challenge>
```

[2] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.

After a challenge is received, the client replies to the challenge by sending a <response/> element qualified by the 'urn:xmpp:register:0' namespace or a cancelation as defined later in this document. If the client sends a response, it MUST also include a payload defined by the specific challenge type. In the case of a jabber:x:data challenge, the payload should be a form submission as defined by Data Forms (XEP-0004) [3] (an 'x' element of type 'submit'). For instance, to reply to the data form challenge from the previous example a client might send:

Listing 3: User Submits Registration Form

```
<response xmlns='urn:xmpp:register:0'>
  <x xmlns='jabber:x:data' type='submit'>
    <field type='hidden' var='FORM_TYPE'>
      <value>urn:xmpp:register:0</value>
    </field>
    <field type='text-single' label='Given Name' var='first'>
      <value>Juliet</value>
    </field>
    <field type='text-single' label='Family Name' var='last'>
      <value>Capulet</value>
    </field>
    <field type='text-single' label='Nickname' var='nick'>
      <value>Jule</value>
    </field>
    <field type='text-single' label='Recovery Email Address' var='
       email'>
      <value>juliet@capulet.com</value>
    </field>
  </x>
</response>
```

If after receiving a challenge a client does not wish to continue registration or recovery, it may send an empty <cancel> element qualified by the 'urn:xmpp:register:0' namespace. This informs the server that registration is complete. This is the same as submitting a data form of type 'cancel' in response to a data form challenge.

Listing 4: User Cancels Registration or Recovery

```
<cancel xmlns='urn:xmpp:register:0'/>
```

## 7 Completing Registration or Recovery

If the client submits invalid data, or the server wishes to cancel for some other reason, it may reply with an empty <cancel/> element qualified by the 'urn:xmpp:register:0' namespace. If the client successfully completes the challenge, the server MAY return an empty <success/>

[3] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.

element qualified by the 'urn:xmpp:register:0' namespace, at which point it may continue with the stream negotiation process. If the server needs more information, for example, in the previous challenge the user entered an email and now the server wishes to ask for a code that was sent to that email, the server MAY send another challenge.

## 8 Internationalization Considerations

When providing instructions in a data form the server SHOULD use the language specified in the XML stream's current xml:lang, or the closest language for which the server has a translation (eg. based on mutual intelligibility between scripts and languages).
For more information about language tags and matching, see BCP 47 [4]

## 9 Security Considerations

Servers that allow in-band registration need to take measures to prevent abuse. Common techniques to prevent spam registrations include displaying CAPTCHAs or requiring proof-of-posession of a valid email address or telephone number by sending a unique code (e.g. an HMAC that can later be verified as having originated at the server) to the users email and requiring that they enter the code before continuing. Servers that do not take such measures risk being black listed by other servers in the network.

## 10 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [5].

## 11 XMPP Registrar Considerations

### 11.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:register:0

[4] BCP 47: Tags for Identifying Languages <http://tools.ietf.org/html/bcp47>.
[5] 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/>.

Upon advancement of this specification from a status of Experimental to a status of Draft, the XMPP Registrar [6] shall add the foregoing namespace to the registry located at <https://xmpp.org/registrar/stream-features.html>, as described in Section 4 of XMPP Registrar Function (XEP-0053) [7].

## 11.2 IBR Challenge Types Registry

The XMPP Registrar shall maintain a registry of IBR challenge types. Challenge types defined within the XEP series MUST be registered with the XMPP Registrar.
In order to submit new values to this registry, the registrant shall define an XML fragment of the following form and either include it in the relevant XMPP Extension Protocol or send it to the email address <registrar@xmpp.org>:

```
<challenge>
  <name>The name of the challenge type.</name>
  <desc>A natural-language summary of the challenge.</desc>
  <payloaddoc>
    The document in which the IBR challenge payload is specified.
  </payloaddoc>
  <doc>
    The doucment in which the IBR challenge itself is specified (may
        be the same
    as <payloaddoc/>).
  </doc>
</challenge>
```

For an example registration, see the next section.

## 11.3 Challenge Types

This specification defines the following IBR challenge types:

- jabber:x:data

Upon advancement of this specification from a status of Experimental to a status of Draft, the XMPP Registrar [8] shall add the following definition to the IBR challenge types registry, as described in this document:

[6] 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/>.
[7] XEP-0053: XMPP Registrar Function <https://xmpp.org/extensions/xep-0053.html>.
[8] 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/>.

```
<challenge>
  <name>Data Forms Challenge</name>
  <desc>Requests that the client fill out an XEP-0004 data form.</desc
     >
  <payloaddoc>XEP-0004</payloaddoc>
  <doc>TODO: Insert this document once it is assigned a number</doc>
</profile>
```

### 11.4 Namespace Versioning

If the protocol defined in this specification undergoes a revision that is not fully backwards-compatible with an older version, the XMPP Registrar shall increment the protocol version number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.