Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)
draft-saintandre-xmpp-iri-02

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on April 2, 2006.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This document defines the use of Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) in identifying or interacting with entities that can communicate via the Extensible Messaging and Presence Protocol (XMPP).
Table of Contents

1. Introduction ........................................... 3
   1.1 Terminology ......................................... 3
2. Use of XMPP IRIs and URIs ............................... 3
   2.1 Rationale ........................................... 4
   2.2 Form ............................................... 4
   2.3 Authority Component ................................. 6
   2.4 Path Component ..................................... 7
   2.5 Query Component .................................... 7
   2.6 Fragment Identifier Component ...................... 8
   2.7 Generation of XMPP IRIs/URIs ....................... 9
   2.8 Processing of XMPP IRIs/URIs ....................... 11
   2.9 Internationalization .............................. 13
3. IANA Registration of xmpp: URI Scheme ................ 14
   3.1 URI scheme name .................................... 14
   3.2 URI scheme syntax .................................. 14
   3.3 Character encoding considerations .................. 15
   3.4 Intended usage ..................................... 15
   3.5 Applications and/or protocols which use this scheme name .................................. 15
   3.6 Security considerations ............................. 15
   3.7 Relevant publications ............................... 15
   3.8 Person and email address to contact for further information .................................. 15
   3.9 Author/change controller ............................ 15
4. IANA Considerations .................................... 16
5. Security Considerations ............................... 16
6. References ............................................. 16
   6.1 Normative References ............................... 16
   6.2 Informative References ............................. 17
   Author’s Address ...................................... 19
   Intellectual Property and Copyright Statements ....... 20
1. Introduction

The Extensible Messaging and Presence Protocol (XMPP) is a streaming XML technology that enables any two entities on a network to exchange well-defined but extensible XML elements (called "XML stanzas") in close to real time.

As specified in [XMPP-CORE], entity addresses as used in communications over an XMPP network must not be prepended with a Uniform Resource Identifier (URI) scheme (as specified in [URI]). However, applications external to an XMPP network may need to identify XMPP entities either as URIs or, in a more modern fashion, as Internationalized Resource Identifiers (IRIs; see [IRI]). Examples of such external applications include databases that need to store XMPP addresses and non-native user agents such as web browsers and calendaring applications that provide interfaces to XMPP services.

The format for an XMPP address is defined in [XMPP-CORE]. Such an address may contain nearly any [UNICODE] character and must adhere to various profiles of [STRINGPREP]. The result is that an XMPP address is fully internationalizable and is very close to being an IRI without a scheme. However, given that there is no freestanding registry of IRI schemes, it is necessary to define XMPP identifiers primarily as URIs rather than as IRIs, and to register an xmpp: URI scheme rather than an IRI scheme. Therefore this document does the following:

- Specifies how to identify XMPP entities as IRIs or URIs.
- Specifies how to interact with XMPP entities as IRIs or URIs.
- Formally defines the syntax for XMPP IRIs and URIs.
- Specifies how to transform XMPP IRIs into URIs and vice-versa.
- Registers the xmpp: URI scheme.

1.1 Terminology

This document inherits terminology from [IRI], [URI], and [XMPP-CORE].

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

2. Use of XMPP IRIs and URIs
2.1 Rationale

As described in [XMPP-IM], instant messaging and presence applications of XMPP must handle im: and pres: URIs (as specified by [CPIM] and [CPP]). However, there are many other applications of XMPP (including network management, workflow systems, generic publish-subscribe, remote procedure calls, content syndication, gaming, and middleware), and these applications do not implement instant messaging and presence semantics. Neither does a generic XMPP entity implement the semantics of any existing URI scheme, such as the http:, ftp:, or mailto: scheme. Therefore, it is appropriate to define a new URI scheme that makes it possible to identify or interact with any XMPP entity (not just instant messaging and presence entities) as an IRI or URI.

XMPP IRIs and URIs are defined for use by non-native interfaces and applications, and primarily for the purpose of identification rather than interaction (on the latter distinction, see Section 1.2.2 of [URI]). In order to ensure interoperability on XMPP networks, when data is routed to an XMPP entity (e.g., when an XMPP address is contained in the 'to' or 'from' attribute of an XML stanza) or an XMPP entity is otherwise identified in standard XMPP protocol elements, the entity MUST be addressed as <[node@]domain[/resource]> (i.e., without a prepended scheme), where the "node identifier", "domain identifier", and "resource identifier" portions of an XMPP address conform to the definitions provided in Section 3 of [XMPP-CORE].

(Note: For historical reasons, the term "resource identifier" is used in XMPP to refer to the optional portion of an XMPP address that follows the domain identifier and the "/" separator character (for details, refer to Section 3.4 of [XMPP-CORE]); this use of the term "resource identifier" is not to be confused with the meanings of "resource" and "identifier" provided in Section 1.1 of [URI].)

2.2 Form

As described in [XMPP-CORE], an XMPP address used natively on an XMPP network is a string of Unicode characters that (1) conforms to a certain set of [STRINGPREP] profiles and [IDNA] restrictions, (2) follows a certain set of syntax rules, and (3) is encoded as [UTF-8]. The form of such an address can be represented using Augmented Backus-Naur Form ([ABNF]) as:

```
[ node "@" ] domain [ "/" resource ]
```

In this context, the "node" and "resource" rules rely on distinct profiles of [STRINGPREP] and the "domain" rule relies on the concept
of an internationalized domain name as described in [IDNA] (however, there is no need to refer to punycode in the IRI syntax itself, since any punycode representation would occur only inside an XMPP application in order to represent internationalized domain names).

Naturally, in order to be converted into an IRI or URI, an XMPP address must be prepended with a scheme (specifically the xmpp: scheme) and may also need to undergo transformations that adhere to the rules defined in [IRI] and [URI]. Furthermore, in order to enable more advanced interaction with an XMPP entity rather than simple identification, it is desirable to take advantage of additional aspects of URI syntax and semantics, such as authority components, query components, and fragment identifier components.

Therefore, the ABNF syntax for an XMPP IRI is defined as shown below using Augmented Backus-Naur Form as specified by [ABNF], where the "ifragment", "ihost", and "iunreserved" rules are defined in [IRI] and the "pct-encoded" rule is defined in [URI]:

```
xmppiri   = "xmpp:" ihierxmpp [ "?" iquerycomp ] [ "#" ifragment ]

ihierxmpp = iauthpath / ipathxmpp

iauthpath = "//" iauthxmpp [ "/" ipathxmpp ]

iauthxmpp = [ inodeid "@" ihost ]

inodeid  = *( iunreserved / pct-encoded / nodeallow )

nodeallow = "!" / "$" / "(" / ")" / ";" / "," / "/" / ":" / ":" / ":" / "*" / "+" / "+" / "-" / "#" / "_" / "\" / "^" / "'" / "{" / "|"

iresid   = *( iunreserved / pct-encoded / resallow )

resallow = "!" / "$" / "(" / ")" / ";" / "," / "/" / ":" / ":" / ":" / "*" / "+" / "+" / "+" / "+" / "+" / "+" / "+" / "^" / "_" / "\" / "^" / "_" / "\" / "^" / "_" / "\"

iquerycomp = iquerytype [ *ipair ]

iquerytype = *iunreserved

ipair     = ";" ikey "=" ivalue

ikey      = *iunreserved

ivalue    = *( iunreserved / pct-encoded )
```

However, the foregoing syntax is not appropriate for inclusion in the registration of the xmpp: URI scheme, since the IANA recognizes only URI schemes rather than IRI schemes. Therefore, the ABNF syntax for an XMPP URI rather than IRI is defined as shown in Section 3.2 section of this document (see below under "IANA Registration"). If it is necessary to convert the IRI syntax into URI syntax, an application MUST adhere to the mapping procedure specified in Section 3.1 of [IRI].

The following is an example of a basic XMPP IRI/URI used for purposes
of identifying a node associated with an XMPP server:

xmpp:node@example.com

Descriptions of the various components of an XMPP IRI/URI are provided in the following sections.

2.3 Authority Component

As explained in Section 2.8 of this document, in the absence of an authority component the processing application would authenticate as a configured user at a configured XMPP server. That is, the authority component section is unnecessary and should be ignored if the processing application has been configured with a set of default credentials.

In accordance with Section 3.2 of RFC 3986, the authority component is preceded by a double slash (“//”) and is terminated by the next slash (“/”), question mark (“?”), or number sign (“#”) character, or by the end of the IRI/URI. As explained more fully in Section 2.8.1 of this document, the presence of an authority component signals the processing application to authenticate as the node@domain specified in the authority component, rather than as a configured node@domain. (While it is unlikely that the authority component will be included in most XMPP IRIs or URIs, the scheme allows for its inclusion if appropriate.) Thus, the following XMPP IRI/URI indicates to authenticate as "guest@example.com":

xmpp://guest@example.com

Note well that this is quite different from the following XMPP IRI/URI, which identifies a node "guest@example.com" but does not signal the processing application to authenticate as that node:

xmpp:guest@example.com

Similarly, using a possible query component of "?message" to trigger an interface for sending a message, the following XMPP IRI/URI signals the processing application to authenticate as "guest@example.com" and send a message to "support@example.com":

xmpp://guest@example.com/support@example.com?message

By contrast, the following XMPP IRI/URI signals the processing application to authenticate as its configured default account and send a message to "support@example.com":

xmpp:support@example.com?message
2.4 Path Component

The path component of an XMPP IRI/URI identifies an XMPP address or specifies the XMPP address to which an XML stanza shall be directed at the end of IRI/URI processing.

For example, the following XMPP IRI/URI identifies a node associated with an XMPP server:

```
xmpp:example-node@example.com
```

The following XMPP IRI/URI identifies a node associated with an XMPP server along with a particular XMPP resource identifier associated with that node:

```
xmpp:example-node@example.com/some-resource
```

Inclusion of a node is optional in XMPP addresses, so that the following XMPP IRI/URI simply identifies an XMPP server:

```
xmpp:example.com
```

2.5 Query Component

There are many potential use cases for encapsulating information in the query component of an XMPP IRI/URI; examples include but are not limited to:

- Sending an XMPP message stanza (see [XMPP-IM]).
- Adding a roster item (see [XMPP-IM]).
- Sending a presence subscription (see [XMPP-IM]).
- Probing for current presence information (see [XMPP-IM]).
- Triggering a remote procedure call (see [JEP-0009]).
- Discovering the identity or capabilities of another entity (see [JEP-0030]).
- Joining an XMPP-based text chat room (see [JEP-0045]).
- Interacting with publish-subscribe channels (see [JEP-0060]).
- Providing a SOAP interface (see [JEP-0072]).
- Registering with another entity (see [JEP-0077]).

Many of these potential use cases are application-specific, and the full range of such applications cannot be foreseen in advance given the continued expansion in XMPP development; however, there is agreement within the Jabber/XMPP developer community that all of the uses envisioned to date can be encapsulated via a "query type", optionally supplemented by one or more "key-value" pairs (this is similar to the "application/x-www-form-urlencoded" MIME type...
As an example, an XMPP IRI/URI intended to launch an interface for sending a message to the XMPP entity "example-node@example.com" might be represented as follows:

```
xmpp:example-node@example.com?message
```

Similarly, an XMPP IRI/URI intended to launch an interface for sending a message to the XMPP entity "example-node@example.com" with a particular subject might be represented as follows:

```
xmpp:example-node@example.com?message;subject=Hello%20World
```

If included in an XMPP IRI, the query component MUST first be encoded as a [UTF-8] string and then (if necessary) transformed to conform to the "iquerycomp" rule specified above; if included in an XMPP URI, the query component MUST be transformed to conform to the "querycomp" rule specified in Section 3.2 of this document.

If the processing application does not understand query components, it MUST ignore the query component and treat the IRI/URI as consisting of, for example, `<xmpp:example-node@example.com>` rather than `<xmpp:example-node@example.com?query>`. If the processing application does not understand a particular key within the query component, it MUST ignore that key and its associated value.

(Note: In pursuit of interoperability, it may be helpful to maintain a registry of query types and perhaps even of keys for use in XMPP query components. Given that such values will most likely be specific to particular applications of XMPP rather than core to XMPP itself, it seems reasonable that such a registry, if created, would be maintained by the Jabber Registrar function of the Jabber Software Foundation as described in [JEP-0053], rather than by the IANA. A proposal for creating such a registry can be found in [JEP-0147].)

2.6 Fragment Identifier Component

As stated in Section 3.5 of [URI], "The fragment identifier component of a URI allows indirect identification of a secondary resource by reference to a primary resource and additional identifying information." Because the resource identified by an XMPP IRI/URI does not make available any media type (see [MIME]) and therefore (in the terminology of [URI]) no representation exists at an XMPP resource, the semantics of the fragment identifier component in XMPP IRIs/URIs are to be "considered unknown and, effectively, unconstrained" (ibid.). Particular XMPP applications MAY make use of the fragment identifier component for their own purposes. However,
if a processing application does not understand fragment identifier components or the syntax of a particular fragment identifier component included in an XMPP IRI/URI, it MUST ignore the fragment identifier component.

If included in an XMPP IRI, the fragment identifier component MUST first be encoded as a [UTF-8] string and then (if necessary) transformed to conform to the "ifragment" rule defined in [IRI]; if included in an XMPP URI, the fragment identifier component MUST be transformed to conform to the "fragment" rule defined in [URI].

2.7 Generation of XMPP IRIs/URIs

2.7.1 Generation Method

In order to form an XMPP IRI from an XMPP node identifier, domain identifier, and resource identifier, the generating application MUST concatenate:

1. the "xmpp:" scheme
2. optionally (if an authority component is to be included before the node identifier), the characters "//", an authority component of the form node@domain, and the character "/
3. optionally (if the XMPP address contained an XMPP "node identifier"), a string of Unicode characters that conforms to the "inodeid" rule, followed by the "@" character
4. a string of Unicode characters that conforms to the "ihost" rule
5. optionally (if the XMPP address contained an XMPP "resource identifier"), the character "/" and a string of Unicode characters that conforms to the "iresid" rule
6. optionally (if a query component is to be included), the "?" character and query component
7. optionally (if a fragment identifier component is to be included), the "#" character and fragment identifier component

In order to form an XMPP URI from the resulting IRI, an application MUST adhere to the mapping procedure specified in Section 3.1 of [IRI].

2.7.2 Generation Notes

Certain characters are allowed in the node identifier, domain identifier, and resource identifier portions of a native XMPP address but prohibited by the "inodeid", "ihost", and "iresid" rules of an XMPP IRI. Specifically, the "#" and "?" characters are allowed in node identifiers and the "/", "?", "#", and "@" characters are allowed in resource identifiers, but these characters are used as delimiters in XMPP IRIs; in addition, the " " (US-ASCII space)
character is allowed in resource identifiers but prohibited in IRIs. Therefore, all of the foregoing characters MUST be percent-encoded when transforming an XMPP address into an XMPP IRI.

Consider the following nasty node in an XMPP address:

nasty!#$%()*+,-.;=?[\]^_'{|}~node@example.com

That address would be transformed into the following XMPP IRI:

xmpp:nasty!%23$%()*+,-.;=%3F[\]^_'{|}~node@example.com

Consider the following repulsive resource in an XMPP address (split into two lines for layout purposes):

node@example.com
/repulsive !#$%&'()*+,-./:;<=>?@\]^_'{|}~resource

That address would be transformed into the following XMPP IRI (split into two lines for layout purposes):

xmpp:node@example.com
/repulsive%20!%23$%&’()*+,-.%2F;<=%3F%40[\]^_'{|}~resource

Furthermore, virtually any non-US-ASCII character is allowed in an XMPP address and therefore also in an XMPP IRI, but URI syntax forbids such characters directly and specifies that such characters MUST be percent-encoded. In order to determine the URI associated with an XMPP IRI, an application MUST adhere to the mapping procedure specified in Section 3.1 of [IRI].

2.7.3 Generation Example

Consider the following XMPP address:

<jii&#x159;i@echy.example/v Praze>

(Note: The string "&amp;x159;" stands for the Unicode character LATIN SMALL LETTER R WITH CARON and the string "&amp;x10D;" stands for the Unicode character LATIN SMALL LETTER C WITH CARON, following the "XML Notation" used in [IRI] to represent characters that cannot be rendered in ASCII-only documents (note also that these characters are represented in their stringprep canonical form). The ‘<’ and ‘>’ characters are not part of the address itself, but are provided to set off the address for legibility. For those who do not read Czech, this example could be Anglicized as "george@czech-lands.example/In Prague").
In accordance with the process specified above, the generating application would do the following to generate a valid XMPP IRI from this address:

1. Ensure that the XMPP address conforms to the rules specified in [XMPP-CORE], including application of the relevant [STRINGPREP] profiles and encoding as a [UTF-8] string.
2. Concatenate the following:
   1. the "xmpp:" scheme
   2. an "authority component" if included (not shown in this example)
   3. a string of Unicode characters that represents the XMPP address, transformed in accordance with the "inodeid", "ihost", and "iresid" rules
   4. the "?" character followed by a "query component" if appropriate to the application (not shown in this example)
   5. the "#" character followed by a "fragment identifier component" if appropriate to the application (not shown in this example)

The result is this XMPP IRI:

<xmpp:ji%i159;i%10D;echy.example/v%20Praze>

In order to generate a valid XMPP URI from the foregoing IRI, the application MUST adhere to the procedure specified in Section 3.1 of [IRI], resulting in the following URI:

<xmpp:ji%C5%99i%C4%8Dechy.example/v%20Praze>

2.8 Processing of XMPP IRIs/URIs

2.8.1 Processing Method

If a processing application is presented with an XMPP URI rather than an XMPP IRI, it MUST first convert the URI into an IRI by following the procedure specified in Section 3.2 of [IRI].

In order to decompose an XMPP IRI for interaction with the entity it identifies, a processing application MUST separate:

1. the "xmpp:" scheme
2. the authority component if included (the string of Unicode characters between the "//" characters and the next "/" character, the "?" character, the "#" character, or the end of the IRI)
3. a string of Unicode characters that represents an XMPP address as transformed in accordance with the "inodeid", "ihost", and "iresid" rules
4. optionally the query component if included, using the "?" character as a separator
5. optionally the fragment identifier component if included, using the "#" character as a separator

At this point, the processing application would either (1) complete further XMPP handling itself or (2) invoke a helper application to complete XMPP handling; such XMPP handling would most likely consist of the following steps:

1. If not already connected to an XMPP server, connecting either as the user specified in the authority component or as the configured user at the configured XMPP server, normally by adhering to the XMPP connection procedures defined in [XMPP-CORE]. (Note: the processing application SHOULD ignore the authority component if it has been configured with a set of default credentials.)
2. Optionally determining the nature of the intended recipient (e.g., via [JEP-0030]).
3. Optionally presenting an appropriate interface to a user based on the nature of the intended recipient and/or the contents of the query component.
4. Generating an XMPP stanza that translates any user or application inputs into their corresponding XMPP equivalents.
5. Sending the XMPP stanza via the authenticated server connection for delivery to the intended recipient.

2.8.2 Processing Notes

It may help implementors to note that the first two steps of "further XMPP handling" as described at the end of Section 2.8.1 are similar to HTTP authentication ([HTTP-AUTH]), while the next three steps are similar to the handling of mailto: URIs ([MAILTO]).

As noted in Section 2.7.2 of this document, certain characters are allowed in the node identifier, domain identifier, and resource identifier portions of a native XMPP address but prohibited by the "inodeid", "ihost", and "iresid" rules of an XMPP IRI. The percent-encoded octets corresponding to these characters in XMPP IRIs MUST be transformed into the characters allowed in XMPP addresses when processing an XMPP IRI for interaction with the represented XMPP entity.

Consider the following nasty node in an XMPP IRI:
That IRI would be transformed into the following XMPP address:

nasty!#$%()*+,-.;=?\]^_'{|}~node@example.com

Consider the following repulsive resource in an XMPP IRI (split into two lines for layout purposes):

xmpp:node@example.com
/repulsive%20!%23"$%&'()*+,-./:;<=>?@\]^_'{|}~resource

That IRI would be transformed into the following XMPP address (split into two lines for layout purposes):

node@example.com
/repulsive !#$"%&'()*+,-./;<=?@\]^_'{|}~resource

2.8.3 Processing Example

Consider the XMPP URI that resulted from the previous example:

<xmpp:ji%C5%99i@%C4%8Dechy.example/v%20Praze>

In order to generate a valid XMPP IRI from that URI, the application MUST adhere to the procedure specified in Section 3.2 of [IRI], resulting in the following IRI:

<xmpp:ji&#x159;i@&#x10D;echy.example/v%20Praze>

In accordance with the process specified above, the processing application would remove the "xmpp:" scheme to extract the XMPP address from this XMPP IRI, converting any percent-encoded octets from the "inodeid", "ihost", and "iresid" rules into their character equivalents (e.g., "%20" into the space character).

The result is this XMPP address:

<ji&#x159;i@&#x10D;echy.example/v Praze>

2.9 Internationalization

Because XMPP addresses are [UTF-8] strings and because the non-US-ASCII octets in XMPP addresses can be easily converted to percent-encoded octets, XMPP addresses are designed to work well with Internationalized Resource Identifiers ([IRI]). In particular, with
the exception of stringprep verification, the conversion of syntax-
relevant US-ASCII characters (e.g., "?"), and the conversion of
percent-encoded octets from the "inodeid", "ihost", and "iresid"
rules into their character equivalents (e.g., "%20" into the US-ASCII
space character), an XMPP IRI can be constructed directly by
prepending "xmpp:" to an XMPP address. Furthermore, an XMPP IRI
can be converted into URI syntax by adhering to the procedure specified
in Section 3.1 of [IRI] and an XMPP URI can be converted into IRI
syntax by adhering to the procedure specified in Section 3.2 of
[IRI], thus ensuring interoperability with applications that are able
to process URIs but unable to process IRIs.

3. IANA Registration of xmpp: URI Scheme

This section provides the information required to register the xmpp:
URI scheme.

3.1 URI scheme name

xmpp

3.2 URI scheme syntax

The syntax for an xmpp: URI is defined below using Augmented Backus-
Naur Form as specified by [ABNF], where the "fragment", "host", "pct-
encoded", and "unreserved" rules are defined in [URI]:

xmppuri = "xmpp:" hierxmpp [ "?" querycomp ] [ "#" fragment ]

hierxmpp = authpath / pathxmpp
authpath = "//" authxmpp [ "/" pathxmpp ]
authxmpp = nodeid "@" host
pathxmpp = [ nodeid "@" ] host [ "/" resid ]
nodeid = *( unreserved / pct-encoded / nodeallow )
querycomp = querytype [ *pair ]
querytype = *( unreserved / pct-encoded )
pair = ";" key "=" value
key = *( unreserved / pct-encoded )
value = *( unreserved / pct-encoded )
3.3 Character encoding considerations

In addition to XMPP URIs, there will also be XMPP Internationalized Resource Identifiers (IRIs). Prior to converting an Extensible Messaging and Presence Protocol (XMPP) address into an IRI (and in accordance with [XMPP-CORE]), the XMPP address must be represented as UTF-8 by the generating application (e.g., by transforming an application’s internal representation of the address as a UTF-16 string into a UTF-8 string) and the UTF-8 string must then be prepended with an xmpp: scheme. However, because an XMPP URI must contain only US-ASCII characters, the UTF-8 string of an XMPP IRI must be transformed into URI syntax by adhering to the procedure specified in RFC 3987.

3.4 Intended usage

The xmpp: URI scheme identifies entities that natively communicate using the Extensible Messaging and Presence Protocol (XMPP), and is mainly used for identification rather than processing. However, an application that processes an xmpp: URI SHOULD reconstruct the encapsulated XMPP address, connect to the appropriate XMPP server, and send an appropriate XMPP "stanza" (XML fragment) to the XMPP address. There is no MIME type associated with the xmpp: URI scheme.

3.5 Applications and/or protocols which use this scheme name

The xmpp: URI scheme is intended to be used by interfaces to an XMPP network from non-native user agents such as web browsers, as well as by non-native applications that need to identify XMPP entities as full URIs or IRIs.

3.6 Security considerations

See Security Considerations (Section 5) of XXXX.

3.7 Relevant publications

[XMPP-CORE]

3.8 Person and email address to contact for further information

Peter Saint-Andre [mailto:stpeter@jabber.org]

3.9 Author/change controller

This scheme is registered under the IETF tree. As such, the IETF maintains change control.
4. IANA Considerations

This document registers a URI scheme. The registration template can be found in Section 3 of this document.

5. Security Considerations

The security considerations discussed in [IRI], [URI], and [XMPP-CORE] apply to XMPP IRIs, and the security considerations discussed in [URI] and [XMPP-CORE] apply to XMPP URIs.

Providing an interface to XMPP services from non-native applications introduces new security concerns. For example, the ability to interact with XMPP entities via a web browser may expose sensitive information to attacks that are not possible or that are unlikely on a native XMPP network. Due care must be taken in deciding what information is appropriate for representation in XMPP IRIs or URIs. Care must also be taken in exposing XMPP addresses in the authority and path components of XMPP IRIs or URIs that are publicly accessible.

Advertising XMPP IRIs/URIs in publicly accessible locations (e.g., on websites) may make it easier for malicious users to harvest XMPP addresses and therefore to send unsolicited bulk communications to the users or applications represented by those addresses. Care should be taken in balancing the benefits of open information exchange against the potential costs of unwanted communications.

Passwords and other user credentials are forbidden in the authority component (and in fact are not needed since the use of [SASL] in XMPP makes it possible to use the SASL ANONYMOUS mechanism if desired).

6. References

6.1 Normative References


6.2 Informative References


[URL-GUIDE]


Author’s Address

Peter Saint-Andre
Jabber Software Foundation

Email: stpeter@jabber.org
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.