A Uniform Resource Identifier (URI) Scheme for the Extensible Messaging and Presence Protocol (XMPP)
draft-saintandre-xmpp-uri-08

Abstract

This document defines a Uniform Resource Identifier (URI) scheme for use 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. Description of xmpp: URI Scheme ............................... 3
   2.1 Rationale ................................................. 3
   2.2 Form .................................................... 4
   2.3 Authority Component ...................................... 5
   2.4 Path Component ........................................... 6
   2.5 Query Component .......................................... 6
   2.6 Fragment Identifier Component ............................ 8
   2.7 Generation of XMPP URIs ................................... 8
   2.8 Processing of XMPP URIs .................................... 11
   2.9 Internationalization ...................................... 14

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 URL scheme name ............................................ 15
   3.6 Security considerations ................................... 15
   3.7 Relevant publications ..................................... 16
   3.8 Person and email address to contact for further information ............................................. 16
   3.9 Author/change controller .................................. 16

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. [XMPP-CORE] specifies that on an XMPP network itself, the address of an XMPP entity MUST NOT be prepended with a Uniform Resource Identifier (URI) scheme (as defined in [URI]). However, many applications external to an XMPP network may need to identify XMPP entities as full URIs; examples include databases that need to store XMPP addresses and non-native user agents (e.g., web browsers and calendaring applications) that provide interfaces to XMPP services. In order to address the needs of such applications, this memo defines an xmpp: URI scheme that conforms to both the requirements in [URL-REG] and the recommendations in [URL-GUIDE].

1.1 Terminology

This document inherits terminology described in [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. Description of xmpp: URI Scheme

2.1 Rationale

Many types of application can be built using XMPP. As specified in [XMPP-IM], instant messaging and presence applications of XMPP MUST handle the im: and pres: URI schemes specified by [CPIM] and [CPP]. However, it is appropriate to define an XMPP-specific URI scheme for other applications of XMPP (such as network management, workflow applications, generic publish-subscribe, remote procedure calls, content syndication, gaming, and middleware) since these applications do not necessarily implement instant messaging and presence semantics. Therefore, this document defines a generic URI scheme that will enable applications to address as a URI any entity that can communicate via XMPP.

The xmpp: URI scheme is provided mainly 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 URI 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 (also known as a "JID") 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 ]

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, because a URI is allowed to contain [US-ASCII] characters only and certain characters are reserved in URIs (the "reserved" rule defined in [URI]), an XMPP address must be properly handled when transformed into an XMPP URI (see Section 2.7 of this memo) and the ABNF syntax needs to be adjusted in order to accurately capture the form of an XMPP URI as opposed to a native XMPP address. Furthermore, it is desirable to take advantage of more advanced aspects of URI syntax and semantics in XMPP URIs, such as authority components, query components, and fragment identifier components. Therefore, using the "fragment", "host", "pct-encoded", "query", "sub-delims", and "unreserved" rules defined in [URI], the ABNF syntax for an XMPP URI is defined as follows:
xmppuri    = "xmpp:" hier-xmpp [ "?" querycomp ] [ "#" fragment ]
hier-xmpp = authpath / path-xmpp
authpath  = "//" auth-xmpp [ "/" path-xmpp ]
auth-xmpp = nodeid "@" host
path-xmpp = [ nodeid "@" ] host [ "/" resid ]
nodeid    = * ( unreserved / pct-encoded / nodeallow )
nodeallow = '!/' / '%$' / '{' / '/' / '*'/ '+'/ ',' / ';' / '=' / '^' / '`' / '{' / '|' / '}')
resid     = * ( unreserved / pct-encoded / sub-delims )
querycomp = querytype [ *pair ]
querytype = *( ALPHA / DIGIT / '-' / '_' / '.' / ':' )
pair      = "&" key "=" value
key       = *unreserved
value     = * (unreserved / pct-encoded )

(Note: It would have been desirable to re-use the "userinfo" rule from [URI]; however, this was not possible since the "userinfo" rule allows characters that conform to the "sub-delims" rule, but the "&" and """ characters (which are allowed by the "sub-delims" rule) are disallowed in XMPP node identifiers by the Nodeprep profile of [STRINGPREP] as specified in Appendix A of [XMPP-CORE]. Furthermore, there is no need to refer to punycode in the URI syntax itself, since any punycode representation would occur only inside an XMPP application in order to represent internationalized domain names.)

The following is an example of a basic XMPP URI used for purposes of identifying a node associated with an XMPP server (an IM user is one type of such a node, but by no means the only type):

xmpp:juliet@example.com

Further descriptions of the various components of an XMPP URI are provided in the following sections.

2.3 Authority Component

As explained in Section 2.8 of this memo, in the absence of an authority component the processing application would authenticate as a configured user at a configured XMPP server. The presence of an authority component (always preceded by "/" /) signals the processing application to authenticate as the node@domain specified in the authority component, rather than as a configured node@domain. Thus, the following XMPP URI indicates to authenticate as "guest@example.com":

xmpp://guest@example.com

Note well that this is quite different from the following XMPP 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 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 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

(Note: It is unlikely that the authority component will be included in most XMPP URIs; however, the scheme allows for inclusion of the authority component if appropriate.)

2.4 Path Component

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

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

xmpp:juliet@example.com

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

xmpp:juliet@example.com/balcony

Inclusion of a node is optional in XMPP addresses, so that the following XMPP 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 URI; examples include but are not limited to:

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

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 described in [HTML]).

As an example, an XMPP URI intended to launch an interface for sending a message to the XMPP entity "juliet@example.com" might be represented as follows:

```
xmpp:juliet@example.com?message
```

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

```
xmpp:juliet@example.com?message&subject=Hello%20World
```

If included, the query component MUST first be encoded as a [UTF-8] string; any [UTF-8] encoded octets MUST then be converted into US-ASCII characters, making sure to represent any reserved character (i.e., any character that conforms to the "reserved" rule defined in [URI]) and any character that is outside the range of the US-ASCII coded character set as a percent-encoded octet (see Section 2.1 of [URI]).

If the processing application does not understand query components, it MUST ignore the query component and treat the URI as consisting of, for example, `<xmpp:juliet@example.com>` rather than `<xmpp:juliet@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.

In pursuit of interoperability, it may be valuable to maintain a registry of query types and perhaps even of keys for use in the query component portion of XMPP URIs. 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 is described 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 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 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 URI, it MUST ignore the fragment identifier component.

If included, the fragment identifier component MUST first be encoded as a [UTF-8] string; any [UTF-8] encoded octets MUST then be converted into US-ASCII characters, making sure to represent any reserved character (i.e., any character that conforms to the "reserved" rule defined in [URI]) and any character that is outside the range of the US-ASCII coded character set as a percent-encoded octet (see Section 2.1 of [URI]).

2.7 Generation of XMPP URIs

2.7.1 URI Generation Method

When generating a conformant XMPP URI from an XMPP address, it is necessary to use consistent methods for transforming (1) an XMPP "node identifier" into a string of US-ASCII characters that conforms to the "nodeid" rule, (2) an XMPP "domain identifier" into a string of US-ASCII characters that conforms to the "host" rule, and (3) an XMPP "resource identifier" into a string of US-ASCII characters that conforms to the "resid" rule; such methods are described below.
Naturally, if the XMPP address exists in a non-UTF-8 form (e.g., having been written on a piece of paper or having been represented internally in a computer program as UTF-16), it MUST first be converted to [UTF-8] before the XMPP URI is generated.

In order to transform an XMPP "node identifier" into a string of US-ASCII characters that conforms to the "nodeid" rule, the node identifier MUST first be constructed in accordance with the rules specified in [XMPP-CORE], including application of the Nodeprep profile of [STRINGPREP] (see Appendix A of [XMPP-CORE]) and encoding as a [UTF-8] string; any [UTF-8] encoded octets of the XMPP "node identifier" MUST then be converted into US-ASCII characters, making sure to represent any reserved character (i.e., any character that conforms to the "reserved" rule defined in [URI]) and any character that is outside the range of the US-ASCII coded character set as a percent-encoded octet (see Section 2.1 of [URI]).

In order to transform an XMPP "domain identifier" into a string of US-ASCII characters that conforms to the "host" rule, the domain identifier MUST first be constructed in accordance with the rules specified in [XMPP-CORE], including application of the [NAMEPREP] profile of [STRINGPREP] and encoding as a [UTF-8] string; any [UTF-8] encoded octets of the XMPP "domain identifier" MUST then be converted into US-ASCII characters, making sure to represent any reserved character (i.e., any character that conforms to the "reserved" rule defined in [URI]) and any character that is outside the range of the US-ASCII coded character set as a percent-encoded octet (see Section 2.1 of [URI]).

In order to transform an XMPP "resource identifier" into a string of US-ASCII characters that conforms to the "resid" rule, the resource identifier MUST first be constructed in accordance with the rules specified in [XMPP-CORE], including application of the Resourceprep profile of [STRINGPREP] (see Appendix B of [XMPP-CORE]) and encoding as a [UTF-8] string; any [UTF-8] encoded octets of the XMPP "resource identifier" MUST then be converted into US-ASCII characters, making sure to represent any reserved character (i.e., any character that conforms to the "reserved" rule defined in [URI]) and any character that is outside the range of the US-ASCII coded character set as a percent-encoded octet (see Section 2.1 of [URI]).

In order to form an XMPP URI from the foregoing components, the generating application MUST concatenate:

1. the "xmpp:" scheme
2. optionally (if an authority component is to be included), 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 US-ASCII characters that conforms to the "nodeid" rule, followed by the "@" character
4. a string of US-ASCII characters that conforms to the "host" rule
5. optionally (if the XMPP address contained an XMPP "resource identifier"), the character "/" and a string of US-ASCII characters that conforms to the "resid" 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

2.7.2 URI Generation Example

Consider the following XMPP address:

<jiři@čechy.example/v Praze>

(Note: The string "&#x159;" stands for the Unicode character LATIN SMALL LETTER R WITH CARON and the string "&#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. 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 URI from this address:

1. First 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. Split the address into an XMPP "node identifier" ("jiři"), XMPP "domain identifier" ("čechy.example"), and XMPP "resource identifier" ("v Praze").
3. Transform the XMPP "node identifier" into a string of US-ASCII characters that conforms to the "nodeid" rule by converting the [UTF-8] string to US-ASCII, including conversion of the LATIN SMALL LETTER R WITH CARON character to its percent-encoded representation "%C5%99"; the result is the string "ji%C5%99i".
4. Transform the XMPP "domain identifier" into a string of US-ASCII characters that conforms to the "host" rule by converting the [UTF-8] string to US-ASCII, including conversion of the LATIN SMALL LETTER C WITH CARON character to its percent-encoded representation "%C4%8C"; the result is the string
5. Transform the XMPP "resource identifier" into a string of US-ASCII characters that conforms to the "resid" rule by converting the [UTF-8] string to US-ASCII, including conversion of the " " (SP) character to its percent-encoded representation "%20"; the result is the string "v%20Praze".

6. Concatenate the following:
   1. the "xmpp:" scheme
   2. a URI "authority component" if included (not shown in this example)
   3. the string of US-ASCII characters that conforms to the "nodeid" rule, followed by the "@" character
   4. the string of US-ASCII characters that conforms to the "host" rule
   5. the "/" character followed by the string of US-ASCII characters that conforms to the "resid" rule
   6. the "?" character followed by a URI "query component" if appropriate to the application (not shown in this example)
   7. the "#" character followed by a URI "fragment identifier component" if appropriate to the application (not shown in this example)

The result is this XMPP URI:

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

2.8 Processing of XMPP URIs

2.8.1 URI Processing Method

As with the generation of an XMPP URI from an XMPP address, so also with the processing of an XMPP URI (including the extraction of an XMPP address therefrom): it is necessary to use consistent methods; such methods are described below.

In order to decompose an XMPP URI, a processing application MUST separate:

1. the "xmpp:" scheme
2. optionally (if the XMPP URI contains an authority component), the authority component (the string of US-ASCII characters between the "/" characters and the first "/" character or the end of the URI)
3. optionally a string of US-ASCII characters that conforms to the "nodeid" rule (if any), using the "@" character as a separator
4. a string of US-ASCII characters that conforms to the "host" rule
5. optionally a string of US-ASCII characters that conforms to the "resid" rule (if any), using the "/" character as a separator
6. optionally the query component (if any), using the "?" character as a separator
7. optionally the fragment identifier component (if any), using the "#" character as a separator

In order to reconstruct the XMPP address from the foregoing components, the processing application MUST:

- Transform the string of US-ASCII characters that conforms to the "nodeid" rule into an XMPP "node identifier" by converting each sequence of percent-encoded octets into the appropriate sequence of reserved or non-US-ASCII octets by (1) decoding percent-encoded octets into actual octets, (2) interpreting the octets as [UTF-8], and (3) applying the Nodeprep profile of [STRINGPREP] as specified in Appendix A of [XMPP-CORE].
- Transform the string of US-ASCII characters that conforms to the "host" rule into an XMPP "domain identifier" by converting each sequence of percent-encoded octets into the appropriate sequence of reserved or non-US-ASCII octets by (1) decoding percent-encoded octets into actual octets, (2) interpreting the octets as [UTF-8], and (3) applying the [NAMEPREP] profile of [STRINGPREP].
- Transform the string of US-ASCII characters that conforms to the "resid" rule into an XMPP "resource identifier" by converting each sequence of percent-encoded octets into the appropriate sequence of reserved or non-US-ASCII octets by (1) decoding percent-encoded octets into actual octets, (2) interpreting the octets as [UTF-8], and (3) applying the Resourceprep profile of [STRINGPREP] as specified in Appendix B of [XMPP-CORE].
- Concatenate the following (ensuring that the resulting string is encoded as [UTF-8]):
  1. the XMPP "node identifier" and the "@" character (if a string of US-ASCII characters that conforms to the "nodeid" rule was included)
  2. the XMPP "domain identifier"
  3. the "/" character and XMPP "resource identifier" (if a string of US-ASCII characters that conforms to the "resid" rule was included)

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. Authenticating either as the user specified in the authority component or as the configured user at the configured XMPP server if not already so authenticated.
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.

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

2.8.2 URI Processing Example

Consider the XMPP URI that resulted from the previous example:

```xmpp:ji%C5%99i%C4%8Cechy.example/v%20Praze```

In accordance with the process specified above, the processing application would do the following to extract the XMPP address from this XMPP URI:

1. Split the URI into a string of US-ASCII characters that conforms to the "nodeid" rule ("ji%C5%99i"), a string of US-ASCII characters that conforms to the "host" rule ("%C4%8Cechy.example"), and a string of US-ASCII characters that conforms to the "resid" rule ("v%20Praze").
2. Transform the string of US-ASCII characters that conforms to the "nodeid" rule into an XMPP "node identifier" by converting the percent-encoded representation "ji" to its equivalent [UTF-8] character (LATIN SMALL LETTER R WITH CARON), making sure that the entire string is encoded as [UTF-8]; the result is an XMPP "node identifier" of "jiři".
3. Transform the string of US-ASCII characters that conforms to the "host" rule into an XMPP "domain identifier" by converting the percent-encoded representation "%C4%8Cechy.example" to its equivalent [UTF-8] character (LATIN SMALL LETTER C WITH CARON), making sure that the entire string is encoded as [UTF-8]; the result is an XMPP "domain identifier" of "čechy.example" (encoded as a [UTF-8] string).
4. Transform the string of US-ASCII characters that conforms to the "resid" rule into an XMPP "resource identifier" by converting the percent-encoded representation "v%20Praze" to its equivalent [UTF-8] character (SP), making sure that the entire string is encoded as
[UTF-8]; the result is an XMPP "resource identifier" of "v Praze".

5. Concatenate the following (ensuring that the resulting string is encoded as [UTF-8]):
   1. the XMPP "node identifier" and the "@" character
   2. the XMPP "domain identifier"
   3. the "/" character and the XMPP "resource identifier"

The result is this XMPP address:

<jii&#x159;i@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 and the conversion of syntax-relevant US-ASCII characters (e.g., "?"), an XMPP IRI can be constructed directly by prepending "xmpp:" to an XMPP address.

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]. The "fragment", "host", "pct-encoded", "query", "sub-delims", and "unreserved" rules are defined in [URI].
xmppuri   = "xmpp:" hier-xmpp [ "?" querycomp ] [ "#" fragment ]
hier-xmpp = authpath / path-xmpp
authpath = "//" auth-xmpp [ "/" path-xmpp ]
auth-xmpp = nodeid "@" host
path-xmpp = [ nodeid "@" ] host [ "/" resid ]
nodeid   = *( unreserved / pct-encoded / nodeallow )
nodeallow = "!" / ";" / "#" / "$" / "%" / "&" / "*" / ";" / "^" / "_" / "-" / ";" / "" / ";" / "" / "" / "" / "" / "" / "" / ""
resid     = *( unreserved / pct-encoded / sub-delims )
querycomp = querytype [ 'pair ]
querytype = *( ALPHA / DIGIT / '-' / '_' / '.' / ':' )
pair      = "&" key "=" value
key       = *unreserved
value     = *( unreserved / pct-encoded )

3.3 Character encoding considerations

Prior to any conversion into a URI and in accordance with [XMPP-CORE], an Extensible Messaging and Presence Protocol (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). The UTF-8 string MUST then be converted into a US-ASCII string in order to be included in a URI; as part of this conversion, non-US-ASCII octets MUST be percent-encoded as described in Section 2.1 of [URI].

3.4 Intended usage

The xmpp: URI 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, authenticate with the appropriate XMPP server, and send an appropriate XMPP "stanza" (XML fragment) to the XMPP address. There is no MIME type associated with this URI.

3.5 Applications and/or protocols which use this URL scheme name

The xmpp: URI 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.

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

Detailed security considerations for Uniform Resource Identifiers are given in [URI], and for the Extensible Messaging and Presence Protocol in [XMPP-CORE]. 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 URIs. Care must also be taken in exposing XMPP addresses in the authority and path components of XMPP URIs that are publicly accessible.

6. References

6.1 Normative References


[URL-GUIDE]
Informative References


[IRI] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", draft-duerst-iri-11 (work in
progress), December 2004.

[JEP-0009]

[JEP-0030]

[JEP-0045]

[JEP-0053]

[JEP-0060]

[JEP-0072]

[JEP-0077]
Saint-Andre, P., "In-Band Registration", JSF JEP 0077, August 2004.

[JEP-0147]


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 (2004). 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.