mailmaint
Internet-Draft
Intended status: Informational
Expires: 8 May 2026

B. Bucksch
Beonex

4 November 2025

Mail Autoconfig
draft-ietf-mailmaint-autoconfig-04

Abstract

A protocol that allows email applications to set up mail accounts and related accounts with only the email address and password.

It defines how service providers can publish the account configuration, so that email applications can automatically find a working configuration. It reduces setup friction for their users, and calls to the support for the service provider.

Although the discovery process starts with an email address, the protocol is not limited setting up email accounts, but can also set up calendar, contact and file sync, video conference accounts and other accounts that are connected to the same user account.

This protocol uses a well-known address and DNS lookups, based on the email address domain, to find the XML configuration file for the service provider.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://benbucksch.github.io/autoconfig-spec/draft-ietf-mailmaint-autoconfig.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-mailmaint-autoconfig/.

Discussion of this document takes place on the mailmaint Working Group mailing list (mailto:mailmaint@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/mailmaint/. Subscribe at https://www.ietf.org/mailman/listinfo/mailmaint/.

Source for this draft and an issue tracker can be found at https://github.com/benbucksch/autoconfig-spec.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Configuring email, calendar and contacts client applications for a given user account at a specific service provider is often a tedious, error-prone and unnerving process. Even technical users struggle to find the right combination of hostname, port number, security protocols, authentication methods and forms of username. Less technical users often abort entirely. This difficulty is one of the primary factors why many users use provider-specific applications which use proprietary internal protocols instead of generic provider-independent client applications that use open protocols specified by the IETF. This in turn leads to significantly less choice for end users in their everyday user experience for communication, already today. Long-term, this leads to an errosion of standards support and the ability for end users to use the software of their choice.

This protocol allows users to set up their existing email account in a email client application, by entering only their name, email address, and password. The application, by means of the Autoconfig protocol specified here, will determine all the other parameters that are required, including IMAP or POP3 hostname, TLS configuration, form of username, authentication method, and other settings, and likewise for SMTP. Calendar, contact and file sync, video conference accounts and other accounts that are connected to the same user account can be set up at the same time.

The protocol works by first determining the domain from the email address, and then querying well-known URLs at the email provider, which return the configuration parameters in computer-readable form. Failing that, various fallback sources can be applied, like a common database of configurations for large email providers who do not directly support this protocol, or other mechanisms to determine the configuration.

While this Autoconfig protocol was originally conceived for configuring mail clients, it can also be used for accounts of other types, like contacts and calendar sync, chat, video conference, or online publishing. The primary concept and limitation here is that these accounts are hosted by the same provider as the email address.

This protocol is in active production use since 15 years by major email clients, and the configuration database contains configurations for over 80% of all email accounts.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Implementations

Currently, this protocol or parts of it has been implemented by:

* Thunderbird (https://thunderbird.net)
* Parula (https://parula.beonex.com)
* Evolution (https://projects.gnome.org/evolution/)
* KMail (https://userbase.kde.org/KMail)
* Kontact (https://www.kontact.org)
* K9 Mail (https://k9mail.app) and Thunderbird Mobile (https://www.thunderbird.net/mobile/)
* FairEmail (https://email.faircode.eu)
* NextCloud email (https://apps.nextcloud.com/apps/mail)
* Delta Chat (https://delta.chat/)

and likely other mail clients.

A large number of email domains already use AutoConfig to provide the configuration to mail clients and allow an automatic setup.

Some mail servers automatically provide AutoConfig files that follow this specification, including Stalwart (https://stalw.art/), Mailcow (https://mailcow.email/) and many others.

4. Data format

Whether the ISP or a common central database returns the configuration, the resulting document MUST have the following data format and qualities.

The format is in [XML]. The MIME type is text/xml.

The seconds below define the XML and its meaning. Most sections start with a concrete example of the elements that they define.

4.1. XML configuration file example

The following example shows the syntax of the XML configuration file.

```
<?xml version="1.0"?>
<clientConfig version="1.2">
  <emailProvider id="example.com">
    <domain>example.com</domain>
    <domain>example.net</domain>

    <displayName>Google Workspace</displayName>
    <displayShortName>GMail</displayShortName>

      <!-- type=
      "imap": IMAP
      "pop3": POP3
      "jmap": JMAP
      "ews": Microsoft Exchange Web Services
      "activesync": Microsoft ActiveSync
      -->
    <incomingServer type="imap">
      <hostname>imap.example.com</hostname>
      <port>993</port>
        <!--
        "plain": no encryption
        "SSL": TLS on TLS-specific port
        "STARTTLS": mandatory upgrade to TLS via STARTTLS
        -->
      <socketType>SSL</socketType>
        <!-- Authentication methods:
        "password-cleartext": SASL PLAIN, LOGIN or protocol-native login.
        "password-encrypted": SASL CRAM-MD5, DIGEST-MD5 etc. Not TLS.
        "TLS-client-cert": TLS client certificate on TLS layer
        "OAuth2": Provider MUST adhere to section "OAuth2 requirements".
        "none": No authentication

        Multiple <authentication> elements per server
        configuration are valid. Clients will pick the first
        one that they support.
        -->
      <authentication>password-cleartext</authentication>
      <username>%EMAILADDRESS%</username>
    </incomingServer>

    <!-- There can be multiple incoming servers,
```

```
      and even multiple IMAP server configs.
      The first configuration is the preferred one, but the user or
      or client can choose the alternative configs. -->
    <incomingServer type="pop3">
      <hostname>pop.example.com</hostname>
      <port>995</port>
      <socketType>SSL</socketType>
      <authentication>password-cleartext</authentication>
      <username>%EMAILADDRESS%</username>
    </incomingServer>

    <!-- Needed only for IMAP or POP3 -->
    <outgoingServer type="smtp">
      <hostname>smtp.googlemail.com</hostname>
      <port>587</port>
      <socketType>STARTTLS</socketType>
        <!-- smtp-auth (RFC 2554, 4954) or other auth mechanism. -->
      <authentication>password-cleartext</authentication>
      <username>%EMAILADDRESS%</username>
    </outgoingServer>

    <incomingServer type="jmap">
      <url>https://jmap.example.com</url>
        <!-- Authentication methods
        "basic": RFC 7617
        "digest": RFC 7616
        "OAuth2": Provider MUST adhere to section "OAuth2 requirements".
        -->
      <authentication>OAuth2</authentication>
      <authentication>basic</authentication>
      <username>%EMAILADDRESS%</username>
    </incomingServer>

    <incomingServer type="ews">
      <url>https://mail.example.com/EWS/Exchange.asmx</url>
      <username>%EMAILADDRESS%</username>
      <authentication>basic</authentication>
    </incomingServer>

    <incomingServer type="activesync">
      <url>https://mail.example.com/Microsoft-Server-ActiveSync</url>
      <username>%EMAILADDRESS%</username>
      <authentication>OAuth2</authentication>
    </incomingServer>

    <documentation url="https://www.example.com/help/mail/">
      <descr lang="en">Configure mail app for IMAP</descr>
      <descr lang="de">Email mit IMAP konfigurieren</descr>
```

```
    </documentation>

  </emailProvider>

  <addressbook type="carddav">
    <url>https://contacts.example.com/remote.php/dav</url>
    <authentication>basic</authentication>
    <username>%EMAILADDRESS%</username>
  </addressbook>

  <calendar type="caldav">
    <url>https://calendar.example.com/remote.php/dav</url>
    <authentication>basic</authentication>
    <username>%EMAILADDRESS%</username>
  </calendar>

    <!-- Upload files, allowing the user to share them.
    This can be used to send links instead of attachments,
    or to set up a file sync folder on the user's desktop.
    -->
  <fileShare type="webdav">
    <url>https://share.example.com/remote.php/dav</url>
    <authentication>basic</authentication>
    <username>%EMAILADDRESS%</username>
  </fileShare>

  <chatServer type="xmpp">
    <url>wss://example.com:5281/xmpp-websocket</url>
    <authentication>basic</authentication>
    <username>%EMAILADDRESS%</username>
  </chatServer>

  <chatServer type="xmpp">
    <hostname>xmpp.example.com</hostname>
    <port>5223</port>
    <socketType>TLS</socketType>
    <authentication>password-cleartext</authentication>
    <username>%EMAILADDRESS%</username>
  </chatServer>

  <videoConference type="opentalk">
    <url>https://talk.example.com/login</url>
    <authentication>OAuth2</authentication>
    <username>%EMAILADDRESS%</username>
  </videoConference>

    <!-- OAuth2 configuration for native public client apps.
    Gives e.g. clientID, expiry, and login page.
```

```
    The provider MUST adhere to "Open Client OAuth2 profile".
    -->
  <oAuth2>
    <authURL>https://login.example.com/auth</authURL>
    <tokenURL>https://login.example.com/token</tokenURL>
    <issuer>login.example.com</issuer>
    <scope>IMAP POP3 SMTP CalDAV CardDAV WebDAV offline_access</scope>
    <clientID>open</clientID>
    <!-- optional -->
    <clientSecret>give-me-your-password</clientSecret>
  </oAuth2>

  <clientConfigUpdate url="https://www.example.com/config/mail.xml" />

</clientConfig>
```

4.2. Global elements

The file starts with an XML header, e.g. <?xml version="1.0"?>, and is encoded in UTF-8 without BOM.

4.2.1. clientConfig

The root element of the XML file is <clientConfig version="1.2">.

The version is 1.2 for the version defined in this specification. 1.1 is a compatible previous version of this protocol: [Autoconfig1.1]. Higher versions are for future specifications. The client MUST NOT reject a configuration file solely based on the version number.

4.2.2. emailProvider

<emailProvider id="example.com">

Element <emailProvider> is within the root element. This element has no semantic purpose and exists for legacy reasons only, but its content is significant.

The id is a unique string that typically matches the primary domain of the provider. The syntax of the ID is the same as for domain and hostname.

Within <emailProvider> are <domain>, <displayName> and <displayShortName>, <documentation>, <incomingServer> and <outgoingServer>.

4.2.3. domain

```
<domain>example.com</domain>
<domain>example.net</domain>
<domain purpose="mx">example-hosting.com</domain>
```

The content of the <domain> element defines which email addresses this config is valid for. For example, a configuration with <domain>example.com</domain> is valid for email address fred@example.com.

Multiple <domain> elements may be included, which means that the configuration is valid for all of these domains. Their order has no meaning - they may be sorted by number of users, importance to the provider, or alphabethically.

<domain purpose="mx"> specifies the domain name of the MX server of this provider. It is used during the configuration file lookup using MX server names, as specified in Section 5.3 _MX_. If the email address that is to be configured has an MX server that is within the domain given by <domain purpose="mx">, then this configuration is applicable for that email address. Adding the purpose attribute is RECOMMENDED.

4.2.4. displayName and displayShortName

```
<displayName>Google Workspace</displayName>
<displayShortName>GMail</displayShortName>
```

The <displayName> element contains the name of the provider, e.g. as preferred by the marketing of the provider itself. It SHOULD be no longer than 30 characters, but MUST be no longer than 60 characters.

The <displayShortName> element contains the name of the provider, as typically used by end users. It SHOULD be no longer than 12 characters, and it MUST NOT be longer than 20 characters.

Both elements may used to display the provider or account name by the mail client UI, which has reasonable expectations on length and expressiveness.

The <displayShortName> in particular may be used by the mail client together with the email address as partial base for the account name, and during everyday use of the app to represent the account, so it SHOULD be short, distinctive and represent the provider in the way end users refer and think of it.

4.3. documentation

```
<documentation url="https://www.example.com/help/mail/">
  <descr lang="en">Configure mail app for IMAP</descr>
  <descr lang="de">Email mit IMAP konfigurieren</descr>
</documentation>
```

This is purely informational and not required for the automatic setup.

This element contains the end-user help webpage at the provider that describes the mail server settings. The configuration may be based on that page, but does not necessarily have to match it, e.g. when a better configuration is available than the one described on the webpage.

The url attribute contains the URL of the webpage. The <descr> content describes the content and purpose of the page and why it's referenced here. Multiple <descr> elements with different lang attributes are allowed, whereby the lang attribute contains the 2-letter ISO language code, like the HTML lang attribute.

4.4. Server sections

<incomingServer type="jmap">, <calendar type="carddav"> etc.

* The type attribute specifies the wire protocol that this server uses. See Section 4.5 _type_ below.

* <incomingServer> specifies the server that the mail client retrieves email from and submits changes to. In many protocols, this server is also used for sending email.

* <outgoingServer> is used for sending, if <incomingServer> does not support that directly. This is currently used only for SMTP in combination with IMAP and POP3.

* <incomingServer> and <outgoingServer> are within element <emailProvider>, whereas <calendar>, <addressbook>, <fileShare>, <chatServer> and <videoConference> are within the root element <clientConfig>, i.e. one level higher. Other than that, they work the same.

* In some protocols, the <incomingServer> server additionally provides calendars, addressbooks, and other data. For such protocols, the same server is not repeated in other specific server sections like <calendar>. Calendar-only clients supporting such multi-purpose protocols MUST read the <incomingServer> (nonewithstanding that it’s within legacy element <emailProvider>) and test and use the parts of the protocol needed for their functionality.

4.4.1. Multiple server sections

Server sections of the same type (like <incomingServer> or <calendar>) may appear multiple times in the same configuration file. In this case, the one listed first is preferred by the configuration provider. Unless the client has specific other requirements, it SHOULD pick the first config.

The client may deviate from this recommendation, because

* the client doesn’t support a higher-priority protocol, e.g. a JMAP configuraion is listed first and is the most preferred, but the client does not support JMAP yet, or

* the client doesn’t support a configuration setting, e.g. it doesn’t support STARTTLS, or the configuration specifies only an OAuth2 authentication and the client either doesn’t implement OAuth2, or there is a problem in the OAuth2 flow with this provider, or

* the client has a specific policy to prefer another configuration, e.g. a STARTTLS configuration is listed before a direct TLS config, and the client has a policy of preferring direct TLS, or likewise the client prefers IMAP over POP3.

Server types, elements and protocols that the client does not support MUST be ignored and the client MUST continue to parse the other server sections, which may contain configs that the client understands and supports. The client ignores the file only if there is no supported and working configuration found.

4.5. type

The type attribute on the server section element specifies the wire protocol that this server uses.

Element	Type	Base	Name	Specification
incomingServer	jmap	URL	JMAP	[JMAP-Core], [JMAP-Mail], [JMAP-WebSocket], [JMAP-Contacts] et al
incomingServer	imap	TCP	IMAP	[IMAP4rev2] or [IMAP4rev1], et al
incomingServer	pop3	TCP	POP3	[POP3], [POP3-SASL]
outgoingServer	smtp	TCP	SMTP	[SMTP], [EMail]
calendar	caldav	URL	CalDAV	[CalDAV]
addressbook	carddav	URL	CardDav	[CardDav]
fileShare	webdav	URL	WebDAV	[WebDAV]
chatServer	xmpp	URL	XMPP	[XMPP], [XMPP-IM], [XMPP-WebSocket]
chatServer	xmpptcp	TCP	XMPP	[XMPP], [XMPP-IM]
chatServer	matrix	URL	Matrix	[Matrix]
setupServer	managesieve	TCP	ManageSieve	[ManageSieve], [Sieve]
incomingServer	ews	URL	Exchange Web Services	
incomingServer	activeSync	URL	ActiveSync	
incomingServer	graph	URL	Microsoft Graph	

Table 1

Other protocol names can be added using an IANA registry. See the corresponding section below.

4.6. URL-based protocols

```
<incomingServer type="jmap">
  <url>https://jmap.example.com/session</url>
  <authentication>basic</authentication>
  <username>%EMAILADDRESS%</username>
</incomingServer>
```

For server sections with protocols that are based on HTTPS or other URLs, the following elements are supported:

4.6.1. url

Example: <url>https://jmap.example.com/session</url>

The content of the <url> element contains the [URL] where to contact the server.

The URL scheme will normally be HTTPS and the URL starts with https:// ([HTTP], Section 4.2.2). Some protocols may use other schemes, e.g. WebSockets wss:// ([WebSocket], Section 3).

4.6.2. authentication

Example: <authentication system="http">basic</authentication> or <authentication>OAuth2</authentication>

The content of the <authentication> element defines which HTTP authentication method to use. The system="http" attribute signals that the value refers to a WWW-Authenticate mechanism, but the attribute is optional when a https: or wss: URL is used.

* basic: Authenticate to the HTTP server using WWW-Authenticate: Basic. See [HTTP-Basic-Auth].

* digest: Authenticate to the HTTP server using WWW-Authenticate: Digest. See [HTTP-Digest-Auth]

* OAuth2: Authenticate to the HTTP server using WWW-Authenticate: Bearer. See [OAuth2], Section 3. The provider MUST adhere to the requirements defined in Section 7.4 _OAuth2 requirements_. Note: The XML element for OAuth2 is <authentication>OAuth2</authentication> without system attribute. <authentication system="http">Bearer</authentication> is invalid.

The rules as specified in Section 4.7.4.2 _Multiple authentication alternatives_ and Section 4.7.4.3 _Authentication verification and fallback_ apply here as well.

4.6.3. username

<username>%EMAILADDRESS%</username> or <username>fred</username>

The username to use for the authentication method.

Placeholders MUST be replaced before using the actual value. See Section 4.8 _Placeholders_.

4.7. TCP-based protocols

For server sections with protocols that are based on TCP, the following elements are supported:

```
<incomingServer type="imap">
  <hostname>imap.example.com</hostname>
  <port>993</port>
  <socketType>SSL</socketType>
  <authentication>password-cleartext</authentication>
  <username>%EMAILADDRESS%</username>
</incomingServer>
```

4.7.1. hostname

<hostname>imap.example.com</hostname>

The content of the <hostname> element contains the fully qualified hostname of the server.

4.7.2. port

<port>993</port>

The content of the <port> element is an integer and contains the TCP port number at the hostname. The port is typically specific to the combination of the wire protocol and socketType.

4.7.3. socketType

<socketType>SSL</socketType>

The content of the <socketType> element specifies whether to use direct TLS, STARTTLS, or none of these.

* SSL: Directly contact the TCP port using TLS. TLS version 1.2 [TLSv1.2] or higher SHOULD be used. Higher versions may be required based on security situation, server support, and client policy decisions.

* STARTTLS: Contact the TCP port first using an unencrypted plain socket, then upgrade to TLS using the protocol-specific STARTTLS specification [STARTTLS]. With this configuration, STARTTLS MUST be used and TLS MUST be used after the STARTTLS upgrade. If the upgrade to TLS fails for whatever reason, the client MUST disconnect and MUST NOT try to authenticate. This prevents downgrade attacks that could otherwise steal passwords, user data, and impersonate users.

* plain: Unencrypted connection, with neither TLS nor STARTTLS. May be needed for local servers. Deprecated.

4.7.3.1. TLS validation

In all cases where TLS is used, either directly or using STARTTLS, the client MUST validate the TLS certificate and ensure that the certificate is valid for the hostname given in this config. If not, or if the TLS certificate is otherwise invalid, the client MUST either disconnect or MAY warn the user of the dangers and ask for user confirmation. Such fallback with warning and confirmation is allowed only at original configuration and MUST NOT be allowed during normal everyday connection.

If the server had a valid TLS certificate during original configuration and the TLS certificate is later invalid during normal connection, the client MUST disconnect.

As an exception, if the problem is that the TLS certificate expired recently, the client MAY choose to not consider that a failure during normal connection and MAY use other recovery mechanisms.

4.7.4. authentication

<authentication>password-cleartext</authentication>

The content of the <authentication> element defines which authentication method to use. This can be either an authentication defined by the wire protocol, or a SASL scheme, or a successor to SASL.

* password-cleartext: Send password in the clear. Uses either the native authentication method defined by the wire protocol (if that is based on plaintext passwords), or a SASL authentication scheme like SASL PLAIN or SASL LOGIN, or a successor.

* password-encrypted: An encrypted or hashed password mechanism. Includes SASL CRAM-MD5 [CRAM-MD5], DIGEST-MD5 [DIGEST-MD5], SCRAM-SHA-256-PLUS [SCRAM], and successors. TLS by itself does not qualify as password-encrypted.

* NTLM: Legacy Windows login mechanisms NTLM or NTLMv2.

* GSSAPI: [Kerberos] or [GSSAPI], a single-signon mechanism based on TCP.

* TLS-client-cert: On the SSL/TLS layer, after server request, the client sends a TLS client certificate for the user, possibly after letting the user select confirm it. Uses SASL EXTERNAL scheme [SASL], Appendix A.

* OAuth: OAuth. SASL OAUTHBEARER [SASL-OAuth2] (current) or XOAUTH2 (deprecated) or successors. The provider MUST adhere to the requirements defined in Section 7.4 _OAuth2 requirements_.

* client-IP-address: Server can be used without any explicit authentication, and the client is admitted based on its IP address. This may be the case for some SMTP servers on local networks. Not supported on the Internet. Deprecated, because it breaks mobile devices.

4.7.4.1. Recommending specific SASL schemes

<authentication system="sasl">SCRAM-SHA-256-PLUS</authentication>

A specific SASL scheme [SASL] MAY be specified using the specific SASL authentication scheme name, e.g. SCRAM-SHA-256-PLUS [SCRAM]. To signal that, the <authentication> element SHOULD have the system attribute set to value sasl, i.e. <authentication system="sasl">.

In such a case, the server configuration section SHOULD also specify a more generic authentication mechanism as a lower priority alternative. That would make clients use the specific authentication mechanism, if they support it, and other clients will use the more generic authentication mechanism.

4.7.4.2. Multiple authentication alternatives

The <authentication> element may appear multiple times within a server section. In this case, they are ordered from the most to the least preferred method, based on the policy of the provider.

If a client does not support a specific authentication scheme, or does not have the conditions to use it, e.g. the client does not have a Client ID for this OAuth2 server, then the client MUST skip this <authentication> element and use the next in the list instead.

If none of the authentication methods are supported by the client, the client MUST ignore that server section and use the remaining server sections.

4.7.4.3. Authentication verification and fallback

The client SHOULD test the configuration during setup, with an actual authentication attempt.

If the authentication fails, the client decides based on the authentication error code how to proceed. For example, if the authentiocation method itself failed, or the error code indicates a systemic failure, the client SHOULD use a lower-priority authentication method from the list.

If the authentication method is working, but the error code indicated that the username or password was wrong, then the client MAY ask the user to correct the password.

For that reason, the server SHOULD be specific in the error codes and allow the client to distinguish between

* an unsupported or non-working authentication method or other systemic failures
* the client being rejected by the server
* the user being blocked from login
* the user authentication failing due to wrong password or username
* other reasons

If the server were to return the same error code for all these cases, the client might tell the user that the password is wrong, and the user starts attempting other passwords, potentially revealing passwords to other higher-value assets, which is highly dangerous.

If the authentification succeeded, the client SHOULD take note of the working configutation and use that for all subsequent connections, until an explicit reconfiguration occurs. During normal everyday operation, the client SHOULD NOT fallback nor attempt multiple different authentication methods.

4.7.5. username

<username>%EMAILADDRESS%</username> or <username>fred</username>

The username to use for the authentication method.

Placeholders MUST be replaced before using the actual value. See next Section 4.8 _Placeholders_.

4.8. Placeholders

The <username> value may contain placeholders.

The following special substrings in the value MUST be replaced by the client, before the value is actually used.

Placeholder	Replace with	Example
%EMAILADDRESS%	E-Mail-Address of the user	fred@example.com
%EMAILLOCALPART%	Part before @ in the E-Mail-Address	fred
%EMAILDOMAIN%	Part after @ in the E-Mail-Address	example.com

Table 2

Some clients MAY also support the same placeholders for the fields <hostname>, <url>, <authURL>, <tokenURL>, <issuer>, <displayName> and <displayShortName>.

4.9. XML validation

The client SHOULD validate that the configuration file is valid XML, and if the XML syntax is invalid, the client SHOULD ignore the entire file. In contrast, if there are merely unknown elements or attributes, the client MUST NOT ignore the file.

The client SHOULD regard only the elements and attributes that are supported by the client, and MUST ignore the others that are unknown to the client.

The client may optionally want to validate the XML before parsing it. This is not required. If the client choses to validate, the validation MUST ignore unknown elements and attributes and MUST NOT drop or ignore a configuration that contains unknown elements and attributes. This is required to allow future extensions of the format without breaking existing clients.

5. Configuration retrieval by mail clients

The mail client application, when it needs the configuration for a given email address, will perform several steps to retrieve the configuration from various sources.

The steps are ordered by priority. They may all be requested at the same time, but a higher priorty result that is available SHOULD be preferred over a lower priority one, even if the lower priority one is available earlier. Exceptions apply when a higher priority result is either invalid or outdated, or the fetch method is less secure. Lower priority requests MAY be cancelled, if a valid higher priority result has been successfully received. The priority is expressed below with the number before the URL or location, with lower numbers meaning higher priority, e.g. 1.2 has higher priority than 4.1.

In the URLs below, %EMAILADDRESS% SHALL be replaced with the email address that the user entered and wishes to use, and %EMAILDOMAIN% SHALL be replaced with the email domain extracted from the email address. For example, for fred@example.com, the email domain is example.com, and for fred@test.cs.example.net, the email domain is test.cs.example.net.

For full support of this specification, all "Required" and "Recommended" mechanisms MUST be implemented and working. For partial support of this specification, all "Required" mechanisms MUST be implemented and working, and in this case, the implementor SHALL make explicit when advertizing or referring to Autoconfig that there is only partial support of this specification.

5.1. Mail provider

The first step is to directly ask the mail provider and allow it to return the configuration. This step ensures that the protocol is decentralized and the mail provider is in control of the configuration issued to mail clients.

* 1.1. https://autoconfig.%EMAILDOMAIN%/mail/config-v1.1.xml?emailaddress=%EMAILADDRESS% (Required. emailaddress is Optional)

* 1.2. https://%EMAILDOMAIN%/.well-known/autoconfig/mail/config-v1.1.xml (Optional)

* 1.3. http://autoconfig.%EMAILDOMAIN%/mail/config-v1.1.xml (Optional)

For example:

* 1.1. https://autoconfig.example.com/mail/config-v1.1.xml?emailaddress=fred@example.com

* 1.2. https://example.com/.well-known/autoconfig/mail/config-v1.1.xml

* 1.3. http://autoconfig.example.com/mail/config-v1.1.xml

Step 1.3. is mainly for legacy servers. Many current deployments use this HTTP URL.

5.1.1. Customzing the configuration for a specific user

To allow the mail provider to return a configuration adjusted for that mailbox, the client sends the email address as query parameter in URL 1.1.

For example, a global company might want to locate the mailbox geographically close to the user, in the same country or even in the same office building.

However, while the protocol allows for such heterogenous configurations, mail providers are discouraged from doing so, and are instead encouraged to provide one single configuration for all their users. For example, DNS resolution based on location, mail proxy servers, or other techniques as necessary, can be used to route the traffic and host the mail efficiently.

This mechanism also allows the Autoconfig server to map the email address to a username that cannot be expressed using the Placeholders (see Section 4.8). However, this method is discouraged. Instead, the email server login should accept email addresses as username, and doing the mapping to internal usernames at login time, which avoids the need for the client to know a different username.

Autoconfig servers that customize the returned configuration file to the email address should avoid that email addresses can be tested for validity by hostile parties like spammers or attackers. For that reason, Autoconfig servers SHOULD return a real configuration, even if the email address sent as URL parameter does not exist. If the query contains a non-existing email address, the server should not return an error nor a fake configuration, but rather a random real configuation, e.g. a random host out of the pool of real hosts. Supporting mail clients should test the login before completing setup, so spelling mistakes in the email address will be signaled to the user as login error in later stages of the setup process.

5.2. Central database

The ISPDB is a central database that contains the configurations for most mail providers with a market share larger than 0.1%, and contains configurations for half of the email accounts in the world.

This is a useful fallback that allows mail clients to support mail providers which do not host a configuration server described in the previous step. This can increase the success rate of finding a valid configuration up to 10-fold.

The mail client application may choose the mail configuration database provider. A public mail configuration database is available at base URL https://v1.ispdb.net/.

%ISPDB% below is the base URL of that database.

* 2.1. %ISPDB%%EMAILDOMAIN% (Recommended)

For example:

* 2.1. https://v1.ispdb.net/geologist.com (https://v1.ispdb.net/geologist.com)

5.3. MX

Many companies do not maintain their own mail server, but let their email be hosted by a hosting company, which is then responsible for the email of dozens or thousands of domains. For these hosters, it may be difficult to set up the configuration server (step 1.1.) with valid TLS certificate for each of their customers, and to convince their customers to modify their root DNS specifically for Autoconfig. On the other side, the ISPDB can only contain the hosting company and cannot know all their customers. To handle such domains, the protocol first needs to find the server hosting the email.

If the previous mechanisms yield no result, the client SHOULD perform a DNS MX lookup on the email domain, and retrieve the MX server (incoming SMTP server) for that domain. Only the highest priority MX hostname is considered. From that MX hostname, 2 values are extracted:

* Extract only the second-level domain from the MX hostname, and use that as value for %MXBASEDOMAIN%. To determine the second-level domain, use the Public Suffic List (https://publicsuffix.org) or a similarly suited method, to correctly handle domains like .co.uk and .com.au.

* Remove the first component from the MX hostname, i.e. everything up to and including the first ., and use that as value for %MXFULLDOMAIN%. Use it only if it is longer than %MXBASEDOMAIN%.

For example:

* For mx.example.com, the MXFULLDOMAIN and MXBASEDOMAIN are both example.com.

* For mx.example.co.uk, the MXFULLDOMAIN and MXBASEDOMAIN are both example.co.uk.

* For mx.premium.europe.example.com, the MXFULLDOMAIN is premium.europe.example.com and the MXBASEDOMAIN is example.com.

Then, attempt to retrieve the configuration for these MX domains, using the previous methods:

* 3.1. https://autoconfig.%MXFULLDOMAIN%/mail/config-v1.1.xml?emailaddress=%EMAILADDRESS% (Required. emailaddress is Optional)

* 3.2. https://autoconfig.%MXBASEDOMAIN%/mail/config-v1.1.xml?emailaddress=%EMAILADDRESS% (Recommended. emailaddress is Optional)

* 3.3. %ISPDB%%MXFULLDOMAIN% (Recommended)

* 3.4. %ISPDB%%MXBASEDOMAIN% (Recommended)

For example:

* 3.1. https://autoconfig.premium.europe.example.com/mail/config-v1.1.xml?emailaddress=fred@example.com

* 3.2. https://autoconfig.example.com/mail/config-v1.1.xml?emailaddress=fred@example.com

* 3.3. https://v1.ispdb.net/premium.europe.example.com

* 3.4. https://v1.ispdb.net/example.com

5.4. Local disk

For testing purposes, a mail client may want to define a location on the disk, relative to the application installation directory, or relative to the user configuration directory, which may contain a configuration file for a specific domain, and which the mail client will use, if the above methods fail.

* 4.1. %USER_CONFIGURATION_DIR%/isp/%EMAILDOMAIN%.xml (Optional)

* 4.2. %APP_INSTALL_DIR%/isp/%EMAILDOMAIN%.xml (Optional)

For example:

* 4.1. /home/fred/.config/acmemailapp/isp/example.com.xml

* 4.2. /opt/acmemailapp/isp/example.com.xml

5.5. Other mechanisms

A mail client may want to implement other mechanisms to find a configuration, for example Exchange AutoDiscover, DNS-SRV [RFC6186], or heuristic guessing. If the mail client implements such alternative methods, and if they are less secure than some of the mechanisms provided here, the alternative methods SHOULD be considered only with lower priority (as defined above) than the more secure mechanisms defined here. For evaluating other mechanisms, use similar criteria as outlined in Section 8 _Security considerations_.

5.6. Manual configuration

If the above mechanisms fail to provide a working configuration, or if the user explicitly chooses so, the mail client SHOULD give the end user the ability to manually enter a configuration, and use that configuration to configure the account.

6. Configuration validation

6.1. User approval

Independent of the mechanisms used to find the configuration, before using that configuration, the mail client SHOULD display that configuration to the end user and let him confirm it. While doing so:

* At least the second-level domain name(s) of the hostnames involved MUST be shown clearly and with high prominence.

* To avoid spoofing, the client MUST NOT cut off parts of long second-level domains. At least 63 characters MUST be displayed.

* Care SHOULD be taken with international characters that look like ASCII characters, but have a different code.

6.2. Login test

After the user confirmed the configuration, the mail client SHOULD test the configuration, by attempting a login to each server configured. Only if the login succeeded, and the server is working, should the configuration be saved and retrieving and sending mail be started.

6.3. OAuth2 windows

If the configuration contains OAuth2 authentication, or any other kind of authentication that uses a web browser with URL redirects, the mail client MUST show the full URL or the second-level domain of the current page to the end user, at all times, including after page changes, URL changes, or redirects. The authentication start URL may be the email hoster, but it redirects to a corporate server for login, and then back to the hoster. This allows for setups where the hoster is not allowed to see the plaintext passwords.

Showing the URL or hostname allows the end user to verify that he is logging in at the expected page, e.g. the login server of their company, instead of the email hoster’s page. It is important that the user verifies that he enters the passwords on the right domain.

7. Configuration publishing by mail providers

Mail service providers who want to support this specification and publish the mail configuration for their own mail service, so that mail client apps can be automatically configured, MUST follow this section as guideline and MUST respect the definitions in this specification.

* Configuration fields MUST NOT contain invalid or non-working configuration data.

* The provided configuration MUST be working, and SHOULD use state-of-the-art security.

* Configurations MUST be public and MUST NOT require authentication (see below).

7.1. Configuration location for single domain

The configuration file SHOULD be published at the URL for step 1.1., i.e.

* https://autoconfig.%EMAILDOMAIN%/mail/config-v1.1.xml

e.g. for fred@example.com

* https://autoconfig.example.com/mail/config-v1.1.xml

7.2. Configuration location for domain hosters

For mail providers which host entire domains for their business customers, the same URL as listed in the previous section is preferred.

Alternatively, the configuration file SHOULD be published at the locations for step 3.1. and 3.2., i.e.

* https://autoconfig.%MXFULLDOMAIN%/mail/config-v1.1.xml

* https://autoconfig.%MXBASEDOMAIN%/mail/config-v1.1.xml

For example, if the MX server for customer domain example.net is mx.premium.europe.example.com, then the configuration file should be at both

* https://autoconfig.premium.europe.example.com/mail/config-v1.1.xml

* https://autoconfig.example.com/mail/config-v1.1.xml

7.3. No authentication for config

Any of the above URLs for retrieving the configuration file MUST NOT require authentication, but MUST be public.

This is because the configuration information in the Autoconfig file includes the authentication method. Without the Autoconfig file, the client does not know which authentication method is required and which username form to use (e.g. username fred or fred@example.com or fred\EXAMPLE). Given that this information is required for authentication, the Autoconfig file itself cannot require authentication.

7.4. OAuth2 requirements

If OAuth2 is used, the OAuth2 server MUST adhere either to the [OAuth2Client] specification, including all SHOULD requirements stated in those.

The provider MUST allow any client application that acts on behalf of the end user who the mailbox is for. Failure to do so implies a cartell or monopolistic behavior to lock out competing email applications from fulfilling their purpose on behalf of end users, which may be contrary to laws in multiple countries.

The OAuth2 server MUST

* accept the client ID that is given in the configuration file, or if that is not given,

* implement Dynamic Client Registration [RFC7591] in the way as defined by [OAuth2Client] and accept the resulting Client ID,

without client secret. It MAY support multiple of those methods.

The server MUST NOT employ any methods at any point to block or hinder clients applications that are acting on behalf of end users.

The specifications above contain requirements for expiry times and the login page, which are needed for mail client applications to work, and MUST be followed.

The OAuth2 scope MUST include all services defined in this configuration file, so that a single user login is sufficient for all services. The resulting refresh and access tokens MUST be valid for all services defined in the configuration file, including for all URL-based protocols like CalDAV and all TCP-based protocols like IMAP.

8. Security Considerations

8.1. Risk

If an attacker can provide a forged configuration, the provided mail hostname and authentication server can be controlled by the attacker, and the attacker can get access to the plain text password of the user. The attack can be implemented as man-in-the-middle, so the end user might receive mail as expected and never notice the attack.

An attacker gaining the plain text password of a real user is a very significant threat for the organization, not only because mail itself can contain sensitive information and can be used to issue orders within the organization that have wide-ranging impact, but given single-sign-on solutions, the same username and password may give access to other resources at the organization, including other computers or, in the case of admin users, even adminstrative access to the core of the entire organization.

Multi-factor authentication might not defend against such attacks, because the user may believe to be logging into his email and therefore comply with any multi-factor authentication steps required.

8.2. DNS

Any protocol that relies on DNS without further validation, e.g. http, should be considered insecure. This also applies to the DNS MX lookup and the https calls that base on its results, as described in Section 5.3 _MX_.

One possible mitigation is to use multiple different DNS servers and verify that the results match, e.g. to use the native DNS resolver of the operating system, and additionally also query a hardcoded DoH (DNS over HTTPS) server.

Nonetheless, the result should be used with care. If such configs are used, the end user MUST explicitly confirm the config, particularly the resulting second-level domains. See Section 6.1 _User approval_.

8.3. HTTP

HTTP requests may be intercepted, redirected, or altered at the network level. See Section 8.1 _Risk_ above.

Even if an http URL redirects to a https URL, and the domain of the https URL cannot be validated against the email domain, that is still insecure.

For that reason, clients MUST prefer HTTPS over HTTP during configuration retrieval, within the same retrieval method.

If such configs from HTTP are used, the end user MUST explicitly confirm the config, particularly the resulting second-level domains. See Section 6.1 _User approval_.

8.4. Configuration updates

Part of the security properties of this protocol assume that the timeframe of possible attack is limited to the moment when the user manually sets up a new mail client. This moment is triggered by the end user, and a rare action - e.g. maybe once per year or even less, for typical setups -, so an attacker has limited chances to run an attack. While not a complete protection on its own, this reduces the risk significantly.

However, if the mail client does regular configuration updates using this protocol, this security property is no longer given. For regular configuration updates, the mail client MUST use only mechanisms that are secure and cannot be tampered with by an active attacker. Furthermore, the user SHOULD still approve configuration changes.

But even with all these protections implemented, the mail client vendor should make a security assessment for the risks of making such regular updates. The mail client vendor should consider that servers can be hacked, and most users simply approve changes proposed by the app, so these give only a limited protection.

8.5. Server security

Given that mail clients will trust the configuration, the server delivering the configuration file needs to be secure. A static web server offers better security. The server software SHOULD be updated regularly and hosted on a dedicated secure server with all unnecessary services and server features turned off.

For the ISPDB, additions and modifications to the configurations are applicable to all respective users and must be made with care. The authenticity of the configuration MUST be verified from authorative sources. Server hostnames MUST be compared with the email domain names they are serving, and if they differ, the ownership of the server hostnames MUST be validated.

The risk is mitigated to some degree by Section 6.1 _User approval_.

9. Alternatives considered

9.1. DNSSEC

Due to their top-level domain, some domains do not have [DNSSEC] available to them, even if they would like to deploy it.

Even where the top-level domain supports it, DNSSEC is currently deployed in only 1% of domains, with adoption rates falling instead of rising, due to the difficulties of administrating it correctly.

Therefore, DNSSEC cannot be relied on in this specification, and DNS must be considered insecure for the purposes of this specification.

9.2. DNS SRV

DNS SRV protocols [DNS-SRV] [RFC6186] are not used here, for 2 reasons:

1. DNS SRV relies on insecure DNS and the configuration can therefore be trivially spoofed by an attacker. See also DNSSEC above.

2. DNS SRV does not provide all the necessary configuration parameters. For example, we need all of:

* the username form (fred@example.com, or fred, or fred\EXAMPLE, or even a username with no relation to the email address)

* authentication method (password, CRAM-MD5, OAuth2, SSL client certificate)

* authentication method parameters (e.g. OAuth parameters)

and other parameters. If any of these parameters are not configured right, the configuration won’t work. While these parameters could theoretically be added to DNS SRV, that would mean a new specification and render the idea void that this is a protocol that already exists, is standardized and deployed. It is unlikely that all DNS SRV records would be updated with the new values. Therefore, it does not solve the problem.

This specification was created as an answer to these deficiencies and provides an alternative to DNS SRV.

9.3. CAPABILITIES

Deployments in the wild from actual ISPs show that protocol-specific commands to find available authentication methods, e.g. IMAP CAPABILITIES or POP3 CAPA, are not reliable. Many email servers advertize authentication methods that do not work.

Some IMAP servers are default configured to list all SASL authentication methods that have corresponding libraries installed on the system, independent on whether they are actually configured to work. The client receives a long list of authentication methods, and many of them do not work. Additionally, the server response may be only "authentication failed" and may not indicate whether the method failed due to lack of configuration, or because the password was wrong. Because some authentication servers lock the account after 3 failed login attempts, and it may also fail due to unrelated reasons (e.g. username form, wrong password, etc.), the client cannot blindly issue countless login attempts. Locking the account must be avoided. So, simply attempting all methods and seeing which one works is not an option for the email client.

Additionally, some email servers advertize [Kerberos] / [GSSAPI], but when trying to use it, the method fails, and also runs into a long 2 minute timeout in some cases. End users consider that to be a broken app.

Additionally, such commands are protocol specific and have to be implemented in multiple different ways.

Finally, some non-mail protocols may not support capabilties commands that include authentication methods.

10. IANA Considerations

10.1. Registration

IANA will create the following registry in a new registry group called "Mail Autoconfig":

Registry Name: "Autoconfig Protocol Type Names"

Registration Procedure: Specification Required, per [RFC8126], Section 4

Designated Expert: The author of this document.

10.2. Contents

Table, with fields Element (alphanumeric), Type (alphanumeric), Base (URL or TCP or URL/TCP), Name, Specification, and Additional Elements

The registrations need to define:

* Element: The XML element wrapping the server section.
* Type: The type attribute value of the server section.
* Base: Whether the protocol is URL-based or TCP-based,
* Name: The commonly used name of the protocol
* Specification: Which RFCs or document specifies the protocol, and
* Additional Elements: (Optional) Protocol-specific XML elements and their meaning.

10.3. Initial registration

Element	Type	Base	Name	Specification	Additional Elements
incomingServer	jmap	URL	JMAP	RFC 8620, RFC 8621, RFC 8887, RFC 9610 et al	
incomingServer	imap	TCP	IMAP	RFC 9051 or RFC 3501, et al	
incomingServer	pop3	TCP	POP3	RFC 1939, RFC 5034	
outgoingServer	smtp	TCP	SMTP	RFC 5321, RFC 2822	
calendar	caldav	URL	CalDAV	RFC 4791	
addressbook	carddav	URL	CardDav	RFC 6352	
fileShare	webdav	URL	WebDAV	RFC 4918	
chatServer	xmpp	URL	XMPP	RFC 6120, RFC 6121, RFC 7395	
chatServer	xmpptcp	TCP	XMPP	RFC 6120, RFC 6121	
chatServer	matrix	URL	Matrix	https://spec.matrix.org	

				(https://spec.matrix.org)	
setupServer	managesieve	TCP	ManageSieve	RFC 5804, RFC 5228	
incomingServer	ews	URL	Exchange Web Services		
incomingServer	activeSync	URL	ActiveSync		
incomingServer	graph	URL	Microsoft Graph		

Table 3

The Additional Elements field is empty in all of the initial values.

11. References

11.1. Normative References

[CalDAV] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, DOI 10.17487/RFC4791, March 2007, <https://www.rfc-editor.org/rfc/rfc4791>.

[CardDav] Daboo, C., "CardDAV: vCard Extensions to Web Distributed Authoring and Versioning (WebDAV)", RFC 6352, DOI 10.17487/RFC6352, August 2011, <https://www.rfc-editor.org/rfc/rfc6352>.

[CRAM-MD5] Klensin, J., Catoe, R., and P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response", RFC 2195, DOI 10.17487/RFC2195, September 1997, <https://www.rfc-editor.org/rfc/rfc2195>.

[DIGEST-MD5] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, DOI 10.17487/RFC2831, May 2000, <https://www.rfc-editor.org/rfc/rfc2831>.

[EMail] Resnick, P., Ed., "Internet Message Format", RFC 2822, DOI 10.17487/RFC2822, April 2001, <https://www.rfc-editor.org/rfc/rfc2822>.

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[HTTP-Basic-Auth] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <https://www.rfc-editor.org/rfc/rfc7617>.

[HTTP-Digest-Auth] Shekh-Yusef, R., Ed., Ahrens, D., and S. Bremer, "HTTP Digest Access Authentication", RFC 7616, DOI 10.17487/RFC7616, September 2015, <https://www.rfc-editor.org/rfc/rfc7616>.

[IMAP4rev1] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, DOI 10.17487/RFC3501, March 2003, <https://www.rfc-editor.org/rfc/rfc3501>.

[IMAP4rev2] Melnikov, A., Ed. and B. Leiba, Ed., "Internet Message Access Protocol (IMAP) - Version 4rev2", RFC 9051, DOI 10.17487/RFC9051, August 2021, <https://www.rfc-editor.org/rfc/rfc9051>.

[JMAP-Contacts] Jenkins, N., Ed., "JSON Meta Application Protocol (JMAP) for Contacts", RFC 9610, DOI 10.17487/RFC9610, December 2024, <https://www.rfc-editor.org/rfc/rfc9610>.

[JMAP-Core] Jenkins, N. and C. Newman, "The JSON Meta Application Protocol (JMAP)", RFC 8620, DOI 10.17487/RFC8620, July 2019, <https://www.rfc-editor.org/rfc/rfc8620>.

[JMAP-Mail] Jenkins, N. and C. Newman, "The JSON Meta Application Protocol (JMAP) for Mail", RFC 8621, DOI 10.17487/RFC8621, August 2019, <https://www.rfc-editor.org/rfc/rfc8621>.

[JMAP-WebSocket] Murchison, K., "A JSON Meta Application Protocol (JMAP) Subprotocol for WebSocket", RFC 8887, DOI 10.17487/RFC8887, August 2020, <https://www.rfc-editor.org/rfc/rfc8887>.

[ManageSieve]
Melnikov, A., Ed. and T. Martin, "A Protocol for Remotely Managing Sieve Scripts", RFC 5804, DOI 10.17487/RFC5804, July 2010, <https://www.rfc-editor.org/rfc/rfc5804>.

[Matrix] "Matrix protocol specification", n.d., <https://spec.matrix.org>.

[OAuth2] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/rfc/rfc6750>.

[OAuth2Client]
Jenkis, N. and B. Bucksch, "OAuth Profile for Open Public Clients", 2025, <https://datatracker.ietf.org/doc/draft-ietf-mailmaint-oauth-public/>.

[POP3] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, DOI 10.17487/RFC1939, May 1996, <https://www.rfc-editor.org/rfc/rfc1939>.

[POP3-SASL]
Siemborski, R. and A. Menon-Sen, "The Post Office Protocol (POP3) Simple Authentication and Security Layer (SASL) Authentication Mechanism", RFC 5034, DOI 10.17487/RFC5034, July 2007, <https://www.rfc-editor.org/rfc/rfc5034>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <https://www.rfc-editor.org/rfc/rfc7591>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[SASL] Melnikov, A., Ed. and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, DOI 10.17487/RFC4422, June 2006, <https://www.rfc-editor.org/rfc/rfc4422>.

[SASL-OAuth2]
Mills, W., Showalter, T., and H. Tschofenig, "A Set of Simple Authentication and Security Layer (SASL) Mechanisms for OAuth", RFC 7628, DOI 10.17487/RFC7628, August 2015, <https://www.rfc-editor.org/rfc/rfc7628>.

[SCRAM] Hansen, T., "SCRAM-SHA-256 and SCRAM-SHA-256-PLUS Simple Authentication and Security Layer (SASL) Mechanisms", RFC 7677, DOI 10.17487/RFC7677, November 2015, <https://www.rfc-editor.org/rfc/rfc7677>.

[Sieve] Guenther, P., Ed. and T. Showalter, Ed., "Sieve: An Email Filtering Language", RFC 5228, DOI 10.17487/RFC5228, January 2008, <https://www.rfc-editor.org/rfc/rfc5228>.

[SMTP] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/rfc/rfc5321>.

[STARTTLS] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, DOI 10.17487/RFC2595, June 1999, <https://www.rfc-editor.org/rfc/rfc2595>.

[TLSv1.2] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/rfc/rfc5246>.

[URL] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[WebDAV] Dusseault, L., Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, DOI 10.17487/RFC4918, June 2007, <https://www.rfc-editor.org/rfc/rfc4918>.

[WebSocket]
Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <https://www.rfc-editor.org/rfc/rfc6455>.

[XMPP] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/rfc/rfc6120>.

[XMPP-IM] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, DOI 10.17487/RFC6121, March 2011, <https://www.rfc-editor.org/rfc/rfc6121>.

[XMPP-WebSocket] Stout, L., Ed., Moffitt, J., and E. Cestari, "An Extensible Messaging and Presence Protocol (XMPP) Subprotocol for WebSocket", RFC 7395, DOI 10.17487/RFC7395, October 2014, <https://www.rfc-editor.org/rfc/rfc7395>.

11.2. Informative References

[Autoconfig1.1] Bucksch, B., "Autoconfig version 1.1", 2020, <https://www.bucksch.org/1/projects/autoconfiguration/config-file-format.html>.

[DNS-SRV] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/rfc/rfc2782>.

[DNSSEC] Hoffman, P., "DNS Security Extensions (DNSSEC)", BCP 237, RFC 9364, DOI 10.17487/RFC9364, February 2023, <https://www.rfc-editor.org/rfc/rfc9364>.

[GSSAPI] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <https://www.rfc-editor.org/rfc/rfc2743>.

[Kerberos] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, DOI 10.17487/RFC4120, July 2005, <https://www.rfc-editor.org/rfc/rfc4120>.

[RFC6186] Daboo, C., "Use of SRV Records for Locating Email Submission/Access Services", RFC 6186, DOI 10.17487/RFC6186, March 2011, <https://www.rfc-editor.org/rfc/rfc6186>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[XML] Hollenbeck, S., Rose, M., and L. Masinter, "Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols", BCP 70, RFC 3470, DOI 10.17487/RFC3470, January 2003, <https://www.rfc-editor.org/rfc/rfc3470>.

Author’s Address

Ben Bucksch
Beonex
Email: ben.bucksch@beonex.com