OPSEC Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2016

S. Winter
RESTENA

March 18, 2016

A Configuration File Format for Network Services on Leaf Devices
draft-winter-opsec-netconfig-metadata-00

Abstract

This document specifies a YANG module for transfering configuration information of deployments of network services towards leaf nodes (hosts) on the internet. Such configuration files are meant to be discovered, consumed and used by configuration agents on the host to achieve correct and secure setup of these services on the consuming device. This iteration of the I-D concentrates on Wi-Fi network setup and EAP credentials, but is extensible to cover a wide range including VPN, E-Mail and other services.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on September 19, 2016.

Copyright Notice

Table of Contents

1. Introduction

1.1. Problem Statement

The IETF produces many protocols which require configuration by the respective end users. Many of those protocols can be configured securely or not, depending on whether features are turned on or off.

One random example is E-Mail: "STARTTLS when available" allows MITM attacks towards plaintext; "STARTTLS always" prevents them. Another example is the Extensible Authentication Protocol (EAP, [RFC3748]) and its numerous EAP methods (for example EAP-TTLS [RFC5281], EAP-TLS [RFC5216] and EAP-pwd [RFC5931]); the methods have many properties which need to be setup on the EAP server and matched as configuration items on the EAP peer for a secure EAP deployment.

Setting up these protocols and services is comparatively easy if the end-user devices which are to be configured are under central administrative control, e.g. in closed enterprise environments. Group policies or device provisioning by the IT department can push the settings to user devices.

In other environments, for example "BYOD" scenarios where users bring their own devices which are not under enterprise control, service configuration is significantly harder as it has to be done by potentially very non-technical end users.

In the case of Wi-Fi and EAP, correct configuration of all EAP deployment parameters is required to make the resulting authentications

- functional (i.e. the end user can authenticate to an EAP server at all)
- secure (i.e. the end user device can unambiguously authenticate the EAP server prior to releasing any sensitive client-side credentials)
- privacy-preserving (i.e. the end user is able to conceal his username from the EAP authenticator)

It would be desirable to be able to convey the configuration information of a deployment in a machine parseable way to the end-user device, so that all the details need not be known/understood by the user. Instead, the configuration agent on the device could consume the configuration information and set up all details automatically.

However, there is currently no standard way of communicating configuration parameters to devices.

1.2. Approach

This specification defines such a file format for network service configuration metadata. The source definition is a YANG module which allows for automatic derivation of XML and JSON formats.

The specification contains several top-level elements which form the building blocks of service configuration:

- a "Certificates" section which can contain trust roots, intermediate CA certificates, and client certificates.
- a "ClientSideCredentials" section which contains a list of credentials which are valid for one or more services, possibly referencing a client certificate
- a "EAPIdentityProviderList" section with a collection of EAP method details, possibly referencing certificates and ClientSideCredentials as defined above
- a "IPSettingsList" with network configuration items needed to use a network after the authentication
- one or more "WiFiNetwork" blocks which specify layer 2 details of a Wi-Fi connection, referencing IPSettings and possibly EAPIdentityProvider if the network is secured with EAP
- exactly one "ProviderInfo" block with user-displayable information about the entity that provides this configuration file

The specification allows for unique identification of all elements by attaching a UUID to each setting. Using this unique identification, all parts of the configuration file can then refer to this particular piece of information. In particular, several different Wi-Fi networks can reference the same EAPIdentityProvider (and thus the same ClientSideCredential) to indicate that the same authentication settings are valid on all the networks. Configuration agents consuming the file can then ask for the corresponding client-side password once and apply it to all configuration blocks referencing that credential. When considering a hypothetical setup of three Wi-Fi networks, a VPN connection and an E-Mail account which all use the same username and password, all of those can be installed by asking the user for that username/password combination only once.

1.3. Other Approaches

Device manufacturers sometimes have developed their own proprietary configuration formats, examples include Apple’s "mobileconfig" (MIME type application/x-apple-aspen-config), Microsoft’s XML schemata for EAP methods for use with the command-line "netsh" tool, or Intel’s "PRO/Set Wireless" binary configuration files. The multitude of proprietary file formats and their different levels of richness in expression of EAP details create a very heterogenous and non-interoperable landscape.

All of the solutions have their own excentricities or drawbacks. The Microsoft and Intel file formats are limited to Wi-Fi purposes. The Apple mobileconfig approach treats each service as distinct; the same hypothetical situation from the previous paragraph would trigger a username/password prompt five times consecutively during the installation which is rather annoying for end users.

New devices which would like to benefit from machine-parseable configuration information currently either have to choose to follow a competitor’s approach and use that competitor’s file format or have to develop their own. This situation is very unsatisfactory.

1.4. Requirements Language

In this document, several words are used to signify the requirements of the specification. The 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. [RFC2119]

1.5. Terminology

2. YANG module for Network Service Configuration

2.1. Location of the YANG module and derived XML Schema

The schema files are currently hosted on this location:

- YANG module: https://www.supplicants.net/site/standardisation/ietf-winter-netconfig-metadata-00
- XML Schema: https://www.supplicants.net/site/standardisation/ietf-winter-netconfig-metadata-00.xsd

2.2. Description of YANG Module Elements

2.2.1. Overall structure

The root of the Yang module is the container ’NetworkConfiguration’, which contains any of the five elements Certificates, ClientSideCredentials, EAPIdentityProviderList, IPSettingsList, WiFiNetwork, ProviderInfo. These blocks carry the actual configuration information. Individual configuration elements are linked among each other to allow for re-use of the elements: a root CA certificate can be the trust root for multiple purposes; a client credential can be valid for multiple services.

Each linkable element is unique in the sense that it is identified by a UUID value which is required to be unique across the file. These linkable elements ("building blocks") are:

- zero or more Certificate. A Certificate contains the actual blob of the certificate, an indicator what the type of the certificate is, and a display name to present in end user UI.

- zero or more ClientSideCredential. Besides the core information - the ability to store username and password, or link to a client certificate, it contains some meta information such as:

  * a 'ValidUntil' date-and-time timestamp with an indication of possible expiry of the information in the configuration file. Configuration agents importing the configuration file can use this information for example to re-assess whether the account is still valid (e.g. if the ValidUntil timestamp has passed, and authentication attempts consistently fail, the supplicant should consider the information stale and ask the user to verify his access authorisation with the identity provider). Typical use case: student account which expires at end of semester and needs to be reinstated.

  * whether or not the system is allowed to save passwords when they are supplied by the user (allow-save)

  * the format requirements on the username, if there are any (e.g. to enforce that users enter their username as a full NAI realm in the form user@suffix or as an AD domain identifier DOMAIN\user)

  Consumers of configuration files MUST be able to fall back to user-interactive configuration for these parts if they are not specified (e.g. ask for the username and password during import of the configuration data). Configuration files which contain sensitive elements such as 'Password' MUST be handled with due care after the import on the device (e.g. ensure minimal file permissions, or delete the source file after installing).

- zero or more EAPIdentityProvider containers with a list of EAP authentication details for services requiring EAP Authentication. The contents of this container are described in more detail in section Section 2.2.2

- exactly one 'ProviderInfo' container can provide additional information about the supplier of the configuration file, e.g. a logo to allow visual identification of the provider to the user in a user interface, or Acceptable Use Policies pertaining to the use

of the configured services. This element is described in more detail in section Section 2.2.3

- zero or more IPSettings. This block describes details on how to access the wider internet after authentication has completed. It includes ways to get an IP address (DHCP, SLAAC, manual) and proxy settings.

- zero or more WiFiNetwork. This block describes Wi-Fi specific properties of a network and references IPSettings for layer 3, and possibly an EAPIdentityProvider for EAP authentication.

2.2.2. The 'AuthenticationMethods' container

'AuthenticationMethods' contains a sequence of 'AuthenticationMethod' groupings. Each such grouping specifies the properties of one supported authentication method of an EAPIdentityProvider. The content of this grouping is enumerated in section Section 2.2.2.1 The set of configuration parameters specified in the grouping depends on the particular EAP method to be configured.

For instance, EAP-PWD [RFC5931] does not require any server certificate parameters; EAP-FAST and TEAP are the only ones making use of Protected Access Credential (PAC) provisioning. On the other hand, properties such as outer ("anonymous") identity or the need for a trusted root Certification Authority are common to several EAP methods. The server- and client-side credential types of EAP methods are defined as a flat list of elements to choose from (see 'ServerSideCredential' and 'EAPClientSideCredential' below); see section Section 7.2 for a rationale.

Where the sequence of 'AuthenticationMethod' groupings contains more than one element, the order of appearance in the file indicates the server operator's preference for the supported EAP types; occurences earlier in the file indicate a more preferred authentication method.

When a consuming device receives multiple 'AuthenticationMethod' groupings inside 'AuthenticationMethods', it should attempt to install more preferred methods first. During interactive provisioning of EAP properties, if the configuration information for a preferred method is insufficient (e.g. the 'AuthenticationMethod' is EAP-TLS, but the configuration file does not contain the client certificate/private key and the device's credential store is not pre-loaded with the client's certificate), the device should query whether this more preferred method should be used (requiring the user to supplement the missing data) or whether a less-preferred method should be configured instead. In non-interactive provisioning scenarios, all methods should be tried non-interactively in order

until one method can be installed; if no method can be installed in a fully automated way, provisioning is aborted.

2.2.2.1. Authentication Method Properties

The 'AuthenticationMethod' grouping contains

- exactly one 'EAPMethod' leaf, which is an enumerated integer of the EAP method identifier as assigned by IANA (typedef eap-method)
- zero or one container 'ServerSideCredential' which defines means to authenticate the EAP server to the EAP peer (for a list of the elements comprising this container, see section Section 2.2.2.2)
- zero or one container 'EAPClientSideCredential' which defines means to authenticate the EAP peer to the EAP server (for a list of the elements comprising this container, see section Section 2.2.2.3)
- zero or more 'InnerAuthenticationMethod' lists. Occurence of this list indicates that a tunneled EAP method is in use, and that further server-side and/or client-side credentials are defined inside the tunnel. The presence of more than one 'InnerAuthenticationMethod' indicates that EAP Method Chaining is in use, i.e. that several inner EAP methods are to be executed in sequence inside the tunnel. The order of occurence of the inner EAP methods defines the chaining order of the methods.

The 'InnerAuthenticationMethod' list itself contains the same 'EAPMethod', 'ServerSideCredentials' and 'EAPClientSideCredentials' elements as described in the preceding list, but differs in two points:

- It can optionally contain the leaf 'NonEAPAuthMethod' (an enumerated integer of authentication methods not based on EAP) instead of 'EAPMethod' because some tunneled EAP types do not necessarily contain EAP inside the tunnel (e.g. TTLS-PAP, TEAP). The YANG definition ensures that EAPMethod and NonEAPAuthMethod are mutually exclusive in instantiations of the YANG module.
- It can NOT contain a further 'InnerAuthenticationMethod' because establishing a secure tunnel inside an already established secure tunnel is considered a pathological case which needs not be considered. See section Section 7.3 for a rationale.

2.2.2.2. The 'ServerSideCredential' container

The server-side authentication of a mutually authenticating EAP method is typically based on X.509 certificates, which requires the EAP peer to be pre-provisioned with one or more trusted root Certification Authority (CA) prior to authenticating. A server is uniquely identified by presenting a certificate which is signed by these trusted CAs, and by the EAP peer verifying that the name of the server matches the expected one. Consequently, a (set of) CAs and a (set of) server names make up the ServerSideCredentials block.

Note that different EAP methods use different terminology when referring to trusted CA roots, server certificates, and server name identification. They also differ or have inherent ambiguity in their interpretation on where to extract the server name from (e.g. is the server name the CN part of the DistinguishedName, or is the server name one of the subjectAltName:DNS entries; what to do if there is a mismatch?). This specification introduces one single element for CA trust roots and naming; these notions map into the naming of the particular EAP methods very naturally. This specification can not remove the CN vs. sAN:DNS ambiguity in many EAP methods.

- zero or more 'CA' lists: a Certification Authority which is trusted to sign the expected server certificate. The set of 'CA' occurences SHOULD contain self-signed root certificates to establish trust, and MAY contain additional intermediate CA certificates which ultimately root in these self-signed root CAs. A configuration file can, but SHOULD NOT include only an intermediate CA certificate (i.e. without also including the corresponding self-signed root) because trusting only an intermediate CA without being able to verify to a self-signed root is an unsupported notion in many EAP peers.

- zero or more 'ServerID' leafs: these leafs contain the expected server names in incoming X.509 EAP server certificates. For EAP methods not using X.509 certificates for their mutual authentication, these elements contain other string-based handles which identify the server (Example: EAP-pwd).

2.2.2.3. The 'EAPClientSideCredential' container

The actual ClientSideCredential is defined on the top-level and referenced here only by its UUID.

Besides the credentials themselves, there are a variety of EAP-specific properties pertaining to the credential. EAP methods make use of a subset of these properties only. One such property is the "Outer Identity" that some EAP methods support; another one the

ClientSideMTU which is relevant when the client has to send large amounts of client credential data (e.g. a large client certificate). As with server-side credentials, the terminology for the properties may differ slightly between EAP types. The naming convention in this specification maps nicely into the method-specific terminology. Not all the criteria make sense in all contexts; for EAP methods which do not support a criterion, configuration files SHOULD NOT contain the corresponding elements, and consumers of the file MUST ignore these elements.

Specifying any one of these elements except the UUID of the ClientSideCredential is optional.

Leaf 'AnonymousIdentity' is typically used on the outside of a tunneled EAP method and allows to specify which user identity should be used outside the tunnel. This string is not used for actual user authentication, but may contain routing hints to send the request to the right EAP server.

'PAC' contains the Protected Access Credential, typically used in EAP-FAST and TEAP.

'ProvisionPAC' is a boolean which indicates whether a PAC should be provisioned on the first connection. Note that this specification allows to use 'ProvisionPAC' without a CA nor ServerID in 'ServerSideCredential'. While this allows the operation mode of "Anonymous PAC Provisioning" as used in many field deployments of EAP-FAST (and is thus supported here), due to the known security vulnerabilities of anonymous PAC provisioning, this combination SHOULD NOT be used.

2.2.3. The 'ProviderInfo' container

This specification needs to consider that user interaction during the installation time may be required; the user at the very least must be empowered to decide whether the configuration file was issued by a provider he has an account with; the provider may have hints for the user (e.g. which password to use for the login), or may want to display links to helpdesk pages in case the user has problems with the setup or use of his identity.

The 'ProviderInfo' container allows to specify a range of potentially useful information for display to the user (some of which is relevant only during installation time, other pieces of information could be retained by the configuration agent and displayed e.g. in case of failed authentication):

- ’DisplayName’ specifies a user-friendly name for the configuration data provider. Consumers of this specification should be aware that this is simple text, and self-asserted by the producer of the configuration file. If more authoritative information about the issuer is available (e.g. if the file is signed with S/MIME and carries an Organisation name (O attribute) in the signing certificate) then the more authoritative information should be displayed with more prominence than the self-asserted one.

- ’Description’ specifies a generic descriptive text which should be displayed to the user prior to the installation of the configuration data.

- ’ProviderLocation’ specifies the approximate geographic location(s) of the configuration data provider and/or his Points of Presence. This can be useful if a configuration agent has stored or access to many configuration files and tries to suggest probable matching providers based on the device location.

- ’ProviderLogo’ specifies the logo of the configuration data provider. The same self-assertion considerations as for ’DisplayName’ above apply.

- ’TermsOfUse’ contains terms of use to be displayed to and acknowledged by the user prior to the installation of the configuration on the user’s system

- ’Helpdesk’ is a container with three possible sub-elements: ’EmailAddress’, ’WebAddress’ and ’Phone’, all of which can be displayed to the user and possibly retained for future debugging hints.

2.3. Internationalisation / Multi-language support

Some elements in this specification contain text to be displayed in User Interfaces; depending on the user’s language preferences, it would be desirable to present the information in a local language. Other elements contain contact information, and those contact points may only be able to handle requests in a number of languages; it may be desirable to present only contact points to the user which are compatible with his language capabilities.

All elements which either contain localisable text, or which point to external resources in localised languages, use the grouping ’localized-non-interactive’ or ’localized-interactive’. These groupings can occur more than once in the specification, which enables an iteration of all applicable languages. If the grouping is omitted or its ’lang’ leaf is set to "C", the instance of the element

is considered a default choice which is to be displayed if no other language is a better match.

If the entire file content consistently uses only one language set, e.g. all the elements are to be treated as "default" choices, the language can also be set for the entire 'EAPIdentityProvider' element in its own 'lang-tag' leaf.

3. Derivation of formats from YANG source

The utility 'pyang' is used to derive XML Schema (XSD) from the YANG source. The Schema for this Internet-Draft was generated with pyang 1.4.1.

4. Issuer Authentication, Integrity Protection and Encryption of EAP Metadata configuration files

S/MIME, XMLDSIG, JOSE or underlying transport security. Decisions TBD. Nuff said :-)

5. XML Farget Format: File Discovery

5.1. By MIME-Type: application/netconfig-metadata-xml

For transports where the categorisation of file types via MIME types is possible (e.g. HTTP, E-Mail), this document assigns the MIME type

application/netconfig-metadata-xml

Edge devices can associate this MIME type to incoming files on such transports, and register the configuration agent which can consume the data in XML format as the default handler for this file type. By doing so, for example a single click or tap on a link to the file in the device's browser will invoke the configuration process.

This method of discovery is analogous to the Apple "mobileconfig" discovery on recent versions of Mac OS and iOS.

5.2. By filename extension: .netconfig-metadata-xml

In situations where file types can not be determined by MIME type meta-information (e.g. when the file gets stored on a local filesystem), this document RECOMMENDs that configuration data in XML format files be stored with the extension

.netconfig-metadata-xml

to identify the file as containing configuration information in XML format. Edge devices can register the configuration agent which can consume the data with this file extension. By doing so, for example a single click or tap on the filename in the device's User Interface will invoke the configuration process.

5.3. By network location: SCAD

6. JSON Farget Format: File Discovery

6.1. By MIME-Type: application/netconfig-metadata-json

For transports where the categorisation of file types via MIME types is possible (e.g. HTTP, E-Mail), this document assigns the MIME type

application/netconfig-metadata-json

Edge devices can associate this MIME type to incoming files on such transports, and register the configuration agent which can consume the data in JSON format as the default handler for this file type. By doing so, for example a single click or tap on a link to the file in the device's browser will invoke the configuration process.

6.2. By filename extension: .netconfig-metadata-json

In situations where file types can not be determined by MIME type meta-information (e.g. when the file gets stored on a local filesystem), this document RECOMMENDs that configuration data in JSON format files be stored with the extension

.netconfig-metadata-json

to identify the file as containing configuration information in JSON format. Edge devices can register the configuration agent which can consume the data with this file extension. By doing so, for example a single click or tap on the filename in the device's User Interface will invoke the configuration process.

6.3. By network location: SCAD

7. Design Decisions

7.1. Why YANG and not directly XML, JSON or $FOO?

XML is a popular choice for EAP configurations: Microsoft's "netsh" files, Apple's "mobileconfig" files, the Wi-Fi Alliance's "PerProviderSubscription Managed Object", and other vendor/SDO definitions are all using XML.

JSON file formats for EAP configuration exist as well; most notable are Google's most recent efforts for their Chromebook Operating system.

YANG has a very rich feature set, and can codify restrictions on which element is allowed when in a much more fine-grained way than XML Schema could. Since YANG modules can be converted to XML Schema and be instantiated as XML or JSON, they can serve as an abstract notion of EAP configuration which can be deployed on consumer devices in either of those two more popular formats as needed by the device in question.

7.2. Shallow vs. Deep definition of EAP method properties

7.3. EAP tunneling inside EAP tunnels

7.4. Placement of 'OuterIdentity' inside 'AuthenticationMethod'

8. Implementation Status

RFC Editor Note: Please remove this section and the reference to [RFC6982] prior to publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

All of the implementations listed below interoperate from producer- to consumer-side of the EAP metadata specification.

Producers of the configuration files

- eduroam Configuration Assistant Tool

  Organisation: Nicolaus Copernicus University, Torun, Poland

  Implementation Name: eduroam Configuration Assistant Tool

  This existing tool already produces EAP configuration files in various proprietary formats for hundreds of EAP Identity Providers. A module which produces configuration files in the XML variant as specified in an eralier revision of this draft (-00) is in production deployment.

  Link to production version: https://cat.eduroam.org

  Maturity: production

  Coverage: entire specification; XML structure aligns with version -00 of this draft

  Licensing: freely distributable with acknowledgement (BSD style)

  Implementation experience: given that the specification is XML, it is easy to produce a configuration file with common XML libraries. The CAT Framework is written in PHP, which provides ample procedures to produce well-formed XML.

  Contact Information: Tomasz Wolniewicz (see Section 11); the CAT software homepage at http://forge.geant.net/CAT/

Consumers of the configuration files

- Android

  Organisation: Swansea University, Swansea, Wales, U.K.

  Implementation Name: eduroam CAT app

  An Android app, compatible with API level 18 of Android (i.e. version 4.3 and above); the app consumes the -00 revision of this specification. The information in the config files is used to push settings to the SSID 'eduroam' (hard-coded) via the WifiEnterpriseConfig API. The app is in production deployment, with a 4-four digit amount of downloads one month after launch.

  Link to production version: https://play.google.com/store/apps/details?id=uk.ac.swansea.eduroamcat

Maturity: production

Coverage: entire specification; XML structure aligns with version -00 of this draft

Licensing: Apache 2.0

Implementation experience: parsing XML is rather straightfoward. The ability to verify signatures on XML files (S/MIME vs. XMLDSIG as discussed in Section 4) remains unclear at this point.

Contact Information: eduroam CAT Play Store app contact address (playstore@eduroam.org)

- Windows

Organisation: Amebis, d.o.o.i, Kamnik, Slovenia

Implementation Name: ArnesLink

A Windows supplicant/Enterprise WiFi installer/debugging assistant. The application consumes the -02 revision of this specification. The information from the XML variant of this specification is embedded in a larger XML file. The additional parts of the overall configuration file include information regarding the SSID to configure and other useful, but not EAP-specific information. The complete set of information is used to push settings into the Windows Wi-Fi configuration via the 'netsh' tool. The app is in production deployment.

Link to production version: http://ftp.arnes.si/software/eduroam/ArnesLink/

Maturity: production

Coverage: entire specification; XML structure aligns with version -02 of this draft

Licensing: GPL

Implementation experience: parsing XML is rather straightfoward. For Wi-Fi configuration use, the lack of 802.11 specific details in the config file is an issue.

Contact Information: info@amebis.si

- Linux: the authors of this specification are currently developing an application for UNIX-like operating systems which configure enterprise networks via the NetworkManager daemon; the application can consume the file format as defined in this draft specification (XML format) and configure the settings via Networkmanager's D-BUS interface.

9. Security Considerations

10. IANA Considerations

IANA is requested to allocate the MIME type "application/netconfig-metadata-xml" in the MIME Media Types / application registry (see section Section 5.1). The allocation should contain the following values:

- Name: netconfig-metadata-xml
- Template: see Appendix A (RFC editor note: remove this appendix prior to publication; replace this line with the URL to the application as posted online)
- Reference: RFCabcd (RFC editor note: replace with the RFC number of this document)

IANA is requested to allocate the MIME type "application/netconfig-metadata-json" in the MIME Media Types / application registry (see section Section 5.1). The allocation should contain the following values:

- Name: netconfig-metadata-json
- Template: see Appendix A (RFC editor note: remove this appendix prior to publication; replace this line with the URL to the application as posted online)
- Reference: RFCabcd (RFC editor note: replace with the RFC number of this document)

IANA is requested to allocate the location "TBD" in the "well-known URIs" registry. The allocation should contain the following values:

- URI Suffix: TBD
- Change Controller: IETF
- Reference: RFCabcd (RFC editor note: replace with the RFC number of this document)

- Related Information: none

IANA is requested to register the XML namespace "urn:ietf:params:xml:ns:netconfig-metadata-xml" in the "IETF XML Registry / ns". The allocation should contain the following values:

- ID: netconfig-metadata-xml
- URI: urn:ietf:params:xml:ns:netconfig-metadata-xml
- Filename: https://www.iana.org/assignments/xml-registry/ns/netconfig-metadata-xml.txt (to be created by IANA)
- Reference: RFCabcd (RFC editor note: replace with the RFC number of this document)

IANA is requested to register the XML schema "urn:ietf:params:xml:schema:netconfig-metadata-xml" in the "IETF XML Registry / schema". The allocation should contain the following values:

- ID: netconfig-metadata-xml
- URI: urn:ietf:params:xml:schema:netconfig-metadata-xml
- Filename: https://www.iana.org/assignments/xml-registry/schema/netconfig-metadata-xml.xsd (to be created by IANA; current XSD file is linked to in section Section 2.1)
- Reference: RFCabcd (RFC editor note: replace with the RFC number of this document)

11. Contributors

Tomasz Wolniewicz of Nicolaus Copernicus University in Torun, Poland, and Gareth J. Ayres of Swansea University in Swansea, United Kingdom, provided significant input into this specification.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

12.2. Informative References

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008.

[RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, August 2008.

[RFC5931] Harkins, D. and G. Zorn, "Extensible Authentication Protocol (EAP) Authentication Using Only a Password", RFC 5931, August 2010.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

[RFC7593] Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam Architecture for Network Roaming", RFC 7593, DOI 10.17487/RFC7593, September 2015, <http://www.rfc-editor.org/info/rfc7593>.

[HS20] Wi-Fi Alliance, "Hotspot 2.0 Technical Specification", 2012, <https://www.wi-fi.org/hotspot-20-technical-specification-v100>.

Appendix A. Appendix A: MIME Type Registration Template

The following values will be used for the online MIME type registration at https://www.iana.org/form/media-types

Your Name: Stefan Winter

Your Email Address: stefan.winter@restena.lu

Media Type Name: Application

Subtype name: 1) (Standards tree) netconfig-metadata-xml

Subtype name: 2) (Standards tree) netconfig-metadata-json

Required parameters: (none)

Optional parameters: (none)

Encoding Considerations: 8-Bit text

Security Considerations: This file type carries configuration information for consumer devices. It has the potential to substantially alter the consumer's device; particularly to install a new trusted Certification Authority. Applications consuming files of this type need to be cautious to explain to the end user what is being altered, so that they understand the consequences. For further explanations, see Section 9 of this draft. (Note to RFC Editor: replace with the number of this RFC once known)

Interoperability Considerations: The file content is in 1) XML version 1.0 or later; 2) JSON. The encoding SHOULD be UTF-8, but implementations consuming the file SHOULD be prepared to encounter different encodings.

Published Specification: draft-winter-opsec-netconfig-metadata (Note to RFC Editor: replace this reference with the RFC number of this document once known)

Applications which use this media type: files of this type are intended for consumption by sortware on edge devices; they consume the information therein to configure authentication parameters of various network services which are then applied to network or application authentication scenarios.

Fragment Identifier Considerations: files of this type are expected to be transmitted in their entirety. If a reference to a specific part of the content is to be made, XML XPath expressions

are to be used. I.e. fragment identifier formats are not expected to be used.

Restrictions on Usage: none

Provisional registration: initial submission of this form will be executed after adoption in the IETF; it will be a provisional registration. Final registration will be done after IESG review.

Additional information:

Deprecated alias types for this name: none

Magic numbers: none

File extensions: 1) netconfig-metadata-xml

File extensions: 2) netconfig-metadata-json

Macintosh File Type Codes: TBD

Object Identifiers or OIDs: none

Intended Usage: Common (no further provisions)

Other Information/General Comment: none

Person to contact for further information:

Name: Stefan Winter

E-Mail: stefan.winter@restena.lu

Author/Change controller: IETF

DATA

Author’s Address

Stefan Winter
Fondation RESTENA
6, rue Richard Coudenhove-Kalergi
Luxembourg 1359
LUXEMBOURG

Phone: +352 424409 1
Fax: +352 422473
EMail: stefan.winter@restena.lu
URI: http://www.restena.lu.