CALSIFY
Internet-Draft
Intended status: Standards Track
Expires: 14 June 2026

P. M. Hallam-Baker
MPlace2.social

11 December 2025

JSContact Cryptographic Key Extensions
draft-ietf-calext-jscontact-cryptographic-key-00

Abstract

Extensions to the JSContact data model for contact card data are defined to provide improved support for describing cryptographic credentials to be used with applications and services and to provide support for authenticated updates to a contact card. These features in combination provide a basis for establishing and maintaining peer-to-peer trust.

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 https://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 14 June 2026.

Table of Contents

1. Introduction

This document defines extensions to the JSContact data model for contact card data [RFC9553] to provide improved support for describing cryptographic credentials to be used with applications and services and to provide support for authenticated updates to a contact card.

The key design considerations for these extensions are as follows:

* Maintain compatibility with the approach in the base specification. Avoiding unexpected behavior from legacy applications.

* Allow cryptographic credentials to be specified as JSON Web Key Sets [RFC7517].

* Provide more descriptive information for use of cryptographic credentials, in particular specifying which key is to be used with which information service.

* Allow specification of groups of related email addresses and information services.

* Enable automated updates to the contact card data with cryptographic authentication.

In addition, specific guidance is provided on specifying credentials for use with S/MIME, OpenPGP, SSH and Code Signing.

1.1. Linking Keys to Services

JSContact allows a card to specify online services and cryptographic keys but does not provide a means of specifying which key is to be used for which purpose. This is a particular problem with key formats used to support a wide range of applications, an OpenPGP key may be used to sign email or sign a repository commit.

The EmailAddress object and OnlineService object are extended to add a keys property which MAY be used to specify the key identifiers of the related keys.

For example, Alice has an email address entry that has keys for signing and encrypting emails using S/MIME and OpenPGP.

```
{
    "@type":"Card",
    ...
    "emails" : $$$$ Empty $$$$,
    ...
    }
```

1.2. Enhanced specification of cryptographic credentials

The JSContact cryptokeys property allows a card to specify cryptographic credentials as URIs but not their intended uses. A data URI containing an X.509v3 certificate might be intended for use with S/MIME, for code signing or some entirely unrelated purpose. Best design practice encourages the use of common cryptographic infrastructures to support a wide range of applications but best use practices encourage limiting the use of particular cryptographic keys to a single application.

The use of the JSON Web Key (JWK) format provides a much richer format for describing cryptographic keys and their properties than a URI and a media type.

For example, Bob is trying to send an encrypted email to Alice, her contact card lists two keys but only one is an encryption key with the JWK use parameter enc:

[NB: This example is showing presentation as a JWK raw key alone, we would probably want to present the certificate as well, this has been left open so we can have a full discussion before fixing a scheme.]

```
{
    "@type":"Card",
    ...
    "cryptoKeys" : {
...
}
```

1.3. Groups

It is often convenient to organize related email addresses and online services used for a common purpose into groups.

For example, Alice might create separate sets of SSH, repository commit signing and code signing keys for code development:

```
{
    "@type":"Card",
    ...
    "groups" : $$$$ Empty $$$$,
    ...
    }
```

Each group identifier is specified as an online service:

```
{
    "@type":"Card",
    ...
    "onlineServices" : {
...
}
```

1.4. Authenticated Locators

The features described in this document are designed to support but not require the exchange of JSContact data by means of an Encrypted Authenticated Resource Locator (EARL) [draft-hallambaker-earl]. An EARL is a URI form that contains a multi-purpose key that MAY be used to locate, decrypt and authenticate an associated ciphertext package.

For example, Alice’s JSContact information might be retrievable from the EARL:

jscontact://example.com/eio5-53ip-ct6i-x7c7-mm6a-65jg-34pw

Alice might publish her EARL on her business card either as text or as a machine readable code such as a QR code. Alternatively, Alice might publish the information as a prefixed DNS TXT record in the domain she uses as her DNS handle:

```
_jscontact.alice.example.com TXT "jscontact=jscontact://example.com/e
    io5-53ip-ct6i-x7c7-mm6a-65jg-34pw"
```

1.5. Authenticated Updates

The authenticated locator mechanisms described above are intended to be used to establish a ’first contact’ between the parties preserving the maximum possible degree of trust from the context.

Once the initial contact exchange has been achieved, the credentials exchanged in that first contact SHOULD be used to obtain and authenticate future updates.

Contact cards that support updates MUST include a UID property. Updates to contact cards MUST specify the same UID value.

The updates property provides an open framework for describing the update mechanisms supported. The mechanisms provided to update the contact MAY be different from the mechanism originally used to distribute it.

For example, Alice publishes her current contact card by means of a DNS TXT record containing an EARL and a QR code encoding the same EARL on the business card she presents when meeting people in person. Applications MAY update the card by polling the URI specified in the updates entry and verifying the signature on the plaintext enveloped data returned.

```
{
    "@type":"Card",
    ...
    "updates" : {
"update1" : {
  "@type": "Update",
  "uri": "https://contacts.example.com/NCZD-QRTK-J44L-DB4T-4GY6-B35
B-3OWI",
  "cryptoKeyIds": {
    "MDJ2-HOZA-LLSB-ETKS-4HTO-T2DG-6XOP": "sign"}}}
    ...
    "cryptoKeys" : {
"MDJ2-HOZA-LLSB-ETKS-4HTO-T2DG-6XOP" : {
  "@type": "JsonWebKeySet",
  "jsonWebKeys": [{
      "kty": "OKP",
      "kid": "MDJ2-HOZA-LLSB-ETKS-4HTO-T2DG-6XOP",
      "crv": "Ed448",
      "x": "MP4ugyKPz-2lfMCVDqRHPryM5KLQ5MR85lgzQQS-BlmWmB2Oxau8Mj7
SBg3gb1VIJdmxVE8vrOoA"}
    ]}}
    ...
    }
```

Since it is easier to update information on a Web site than in DNS or on a business card, it is likely that some users will prefer to use these mechanisms to distribute pro-forma contact information consisting of basic contact information and update information alone. Therefore, contact applications SHOULD attempt to update contact cards providing update information on receipt.

Retrieving a plaintext signed contact assertion via HTTPS provides a simple but limited update mechanism providing end-to-end integrity but not confidentiality. While the contact information is delivered over an encrypted transport, the contact card is stored unencrypted on the server which may not be acceptable in certain applications. Another limitation is that the relying party is required to poll the contact service for updates. A more sophisticated updates protocol might provide update notifications. Such considerations are outside the scope of this document and left for future work.

[Remark: Addition of a JOSE based encryption scheme would be straightforward. This would require extension of the envelope specification and to pass a decryption key to the parties the contact is passed to.]

A contact card MAY specify multiple update mechanisms providing a degree of resilience in the case that a publication service stops providing service.

For example, Alice might choose three independent contact directory services publishing her contact information on all three ensuring that the people she has shared her contact information with can remain in contact years or even decades after the initial contact exchange.

2. Definitions

This section presents the related specifications and standards, the terms that are used as terms of art within the documents and the terms used as requirements language.

2.1. Requirements Language

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].

2.2. Defined Terms

TBS

2.3. Related Specifications

JSContact: A JSON Representation of Contact Data [RFC9553]. Describe s the format used for contact data interchange.

JSON Web Key (JWK) [RFC7517]. Describes publishing public keys in JSON format.

Encrypted Authenticated Resource Locator [draft-hallambaker-earl]. D escribes a URI form that provides means of retrieving, decrypting and authenticating an associated ciphertext.

2.4. Implementation Status

Reference code under the MIT Open-Source license has been developed to demonstrate all the features described in this document.

3. Card Extensions

The JSContact objects specified in [RFC9553] is extended as described in the following sections.

3.1. Additional Card Properties

The following properties are added to the Card object specified in [RFC9553].

3.1.1. updates

"updates": String[Update] (optional) Specifies mechanisms for obtaining updates to the card.

An Update object has all the properties of the Resource data type, with the following additional definitions:

"protocol": String (optional) The IANA update protocol identifier

"cryptoKeyIds": String[String] (optional) The identifiers of the set of cryptographic keys to be used to authenticate the updated contact information and their use.

```
"updates": {
  "update1": {
    "@type": "Update",
    "uri": "https://contacts.example.com/NCZD-QRTK-J44L-DB4T-4GY6-B35
        B-3OWI",
    "cryptoKeyIds": {
      "MDJ2-HOZA-LLSB-ETKS-4HTO-T2DG-6XOP": "sign"}}}
```

3.1.2. serviceGroups

"serviceGroups": String[ServiceGroup] (optional) Specifies groups of related email addresses and online services.

A ServiceGroup object has all the properties of the Resource data type, with the following additional definition:

"members": String[Boolean] (optional) The identifiers of the service group members. The boolean value corresponding to each MUST be true.

"serviceGroups":

3.2. Extended Objects

The following objects specified in [RFC9553] are extended to add the specified properties.

3.2.1. EmailAddress

An EmailAddress object has the following properties.

3.2.2. cryptoKeyIds

"cryptoKeyIds": String[String] (optional) The identifiers of the set of cryptographic keys to be used to authenticate the updated contact information and their use.

3.2.3. OnlineService

An OnlineService object has the following properties.

3.2.4. cryptoKeyIds

"cryptoKeyIds": String[String] (optional) The identifiers of the set of cryptographic keys to be used to authenticate the updated contact information and their use.

3.3. New Objects

The following object is defined:

3.3.1. JsonWebKeySet

A JsonWebKeySet object has all the properties of the CryptoKey data type, with the following additional definition:

3.3.2. jsonWebKeys

"jsonWebKeys": JWK[] (optional) A Json Web Key Set.

The JWK object is defined in [TBS].

4. Application Profiles

This section _will_ provide guidance on how to encode cryptographic keys for specific applications. It is currently a placeholder so the relevant expert groups have an opportunity to advise the precise means by which the relevant information is presented.

4.1. S/MIME

S/MIME (Secure/Multipurpose Internet Mail Extensions) provides a consistent way to send and receive secure MIME data over SMTP and other messaging transports.

Certificates issued for use with S/MIME are commonly used for other purposes, for example TLS client authentication. For the purposes of this discussion, our focus is strictly limited to the use of the certificate for messaging.

If the messaging protocol is SMTP, the service is specified as an EmailAddress, otherwise the OnlineService structure is used with the appropriate service specifier.

Strawman proposal:

* Use a single JWK entry with an x5c to specify the encryption key cert, signature key cert and cert path.

* Specify the JWK in the keys section of the corresponding EmailAddress or OnlineService with the key identifier of the corresponding JWK and the ’smime’ use.

```
{
    "@type":"Card",
    ...
    "cryptoKeys" : {
...
}
```

4.2. OpenPGP

OpenPGP provides encryption with public key or symmetric cryptographic algorithms, digital signatures, compression, and key management.

OpenPGP key management is commonly used for many purposes including signing repository commits. For the purposes of this section, our focus is strictly limited to the use of keys for messaging.

If the messaging protocol is SMTP, the service is specified as an EmailAddress, otherwise the OnlineService structure is used with the appropriate service specifier.

Strawman proposal:

* User specifies their OpenPGP primary key as an OnlineService with service type 'openpgp'. This is also the place any key server information would be placed.

* The first key in the keys entry of the service is the key identifier of the primary key

* The sub keys are the following keys

* The corresponding key entries are of type JsonWebKey

* The key data is specified as a binary data property

* Other services specify the subkeys that they use.

```
{
    "@type":"Card",
    ...
    "cryptoKeys" : {
...
}
```

4.3. SSH

Although the SSH protocol does support certificates these are not currently widely used on the client side and it is not clear that the use case of a single user specifying multiple SSH client keys for use on multiple devices has been fully distinguished from the use case of a corporation certifying keys for multiple employees.

If the user only has a client single key for a given use, this can be specified as JWK public key parameters.

If the user is making use of client side SSH certificates to provision each device with separate SSH credentials, the contact card should specify the SSH certificate.

```
{
    "@type":"Card",
    ...
    "onlineServices" : {}

{
    "@type":"Card",
    ...
    "cryptoKeys" : {}
```

4.4. Code Signing

Code signing typically makes use of PKIX certificates. While a single certificate could in practice be used for multiple platforms, each code signing program tends to have their own requirements and set of recognized CAs.

It is however useful for a user to specify their own personal hierarchy with a personal root for testing and development purposes.

A developer may have multiple code signing keys for the same platform. For example, separate signing keys for each development machine.

Some mechanism is required to allow development and production keys to be distinguished.

```
{
    "@type":"Card",
    ...
    "onlineServices" : {}

{
    "@type":"Card",
    ...
    "cryptoKeys" : {}
```

4.5. Commit Signing

OpenPGP key management is commonly used for signing repository commits.

This use is specified by means of an OnlineService with the service type ’commit’ with key entries for the set of authorized signing keys.

```
{
    "@type":"Card",
    ...
    "onlineServices" : {}

{
    "@type":"Card",
    ...
    "cryptoKeys" : {}
```

5. IANA Considerations

This document does not specify any actions for IANA yet but it will...[TBS]

6. Acknowledgements

Many thanks to Robert Stepanek who helped refine the approach to extending the schema.

7. Normative References

[draft-hallambaker-earl] Hallam-Baker, P., "Encrypted Authenticated Resource Locator", Work in Progress, Internet-Draft, draft-hallambaker-earl-01, 6 October 2025, <https://datatracker.ietf.org/doc/html/draft-hallambaker-earl-01>.

[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>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/rfc/rfc7517>.

[RFC9553] Stepanek, R. and M. Loffredo, "JSContact: A JSON Representation of Contact Data", RFC 9553, DOI 10.17487/RFC9553, May 2024, <https://www.rfc-editor.org/rfc/rfc9553>.

Author’s Address

Phillip Hallam-Baker
MPlace2.social
Email: phill@hallambaker.com