calext
Internet-Draft
Intended status: Standards Track
Expires: 21 June 2026

R. Stepanek
Fastmail
M. Loffredo
IIT-CNR

18 December 2025

Protocol-Specific Profiles for JSContact
draft-ietf-calext-jscontact-profiles-11

Abstract

This document defines JSContact profiles, an IANA registry for named subsets of JSContact elements. It aims to facilitate using JSContact in context of contact data exchange protocols or other use cases, in which supporting all of JSContact semantics might be inappropriate.

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 21 June 2026.

Table of Contents

1. Notational Conventions 2
2. Introduction 2
3. JSContact Profiles 3
3.1. Profile Name 4
3.2. Profile Version 4
3.3. Profile Properties 5
3.4. Supported Properties 6
4. Example Profile 7
5. IANA Considerations 10
5.1. Creation of the JSContact Profile Registry 10
5.2. Initial Contents of the JSContact Profile Registry . . . 11
6. Security Considerations 11
7. References 11
7.1. Normative References 11
7.2. Informative References 11
Appendix A. Supported Properties Example 12
Authors' Addresses 15

1. Notational Conventions

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.

The ABNF definitions in this document use the notations of [RFC5234]. ABNF rules not defined in this document are defined in [RFC5234] (such as the ABNF for DIGIT).

2. Introduction

The JSContact [RFC9553] contact card data model and format is designed for use in address book applications and directory services. Intended as an alternative to the prevalent vCard [RFC6350] data format, it covers vCard core semantics and extensions, and provides a rich model for personal names, postal addresses and localization. All JSContact elements are relevant for some contact card use case and, similar to vCard, implementations are expected to support these elements when exchanging contact card information using protocols such as CardDAV [RFC6352] and JMAP for Contacts [RFC9610].

In contrast, other protocols and internet standards might require exchanging _some_ contact card information, but not all of what JSContact provides. Section 1.7.4 of [RFC9553] outlines how JSContact implementations may ignore unknown JSContact elements, but

this only applies to future extensions of [RFC9553]; they are still expected to implement all elements of the core specification. Also, the extensibility of JSContact and the requirement to preserve arbitrary contact elements might not be adequate for some protocols.

To make use of JSContact under these circumstances, this document defines a new IANA registry for JSContact that allows for registering named subsets of JSContact elements. These subsets are referred to as "JSContact profiles" and are meant to bring the following benefits:

* Protocol designers might be encouraged to use JSContact, rather than coming up with their own contacts format. This facilitates cross-protocol data exchange and migration.

* Different protocols use the same IANA registry to express which JSContact elements they support. This facilitates understanding their commonalities and reusing existing profiles.

* A central registry provides implementors of JSContact libraries with a consistent format documenting which profile supports what elements, rather than having to look up that information from possibly distinctly organized internet drafts.

This document is organized as follows: Section 3 defines JSContact profiles, Section 4 illustrates JSContact profiles by example, Section 5 summarizes the relevant information for IANA to establish the JSContact Profiles registry.

3. JSContact Profiles

A JSContact profile is a named and versioned set of JSContact properties, value types and values. These JSContact elements MAY be further restricted by the profile, but a profile can not loosen restrictions. For example, a profile can define an originally optional property to become mandatory, but it can not make a mandatory property become optional.

The JSContact elements MUST be registered in the IANA JSContact registry. A JSContact extension MAY define both a new profile and new properties or other elements, as long as they are registered at the same time. A JSContact profile MUST be registered at IANA (see Section 5). Section 3.1 defines how to name a JSContact profile, Section 3.2 defines how to version it, Section 3.3 defines how to specify the properties supported by that profile.

A JSContact object complies with a profile if all its properties are in the set of properties defined by that profile and the property values comply with the profile restrictions for that property. A JSContact object can comply with multiple profiles, even with profiles that weren't defined at the time when the JSContact data was created. Accordingly, this document does not specify any means for JSContact data to communicate which profiles it complies with, e.g. it does not define a "profile" property for the Card object.

All properties and values of a JSContact object that complies with a profile MUST also be valid (Section 1.7 of [RFC9553]). Handling JSContact data that is valid but that does not comply with the expected profile is protocol-specific. This document deliberately does not define such non-compliant data as _invalid_. Profile designers decide on their own strategies for handling non-compliant data, one of which may be to reject it as invalid. JSContact data that complies with a profile may still not be valid in the context of that protocol; the protocol specification MAY define additional restrictions that a profile cannot express.

3.1. Profile Name

A JSContact profile has a unique name. The name MUST only contain ASCII lowercase alphabetic and numeric characters, optionally separated by hyphens. It MUST start with an alphabetic character and it MUST be of at least 1 character and at most 255 characters in size. Formally, it MUST be a valid "profile-name" defined in Figure 1.

```
profile-name = LALPHA *( ["-"] LALPHA / DIGIT )
                ; at most 255 octets in size

LALPHA       = %x61-7A ; a-z
```

Figure 1: ABNF Rule for JSContact Profile Name

3.2. Profile Version

A JSContact profile has a current version, each profile is versioned independently. The version MUST be a positive integer and it MUST increase whenever the profile properties (Section 3.3) change. The inital version value is 1.

If this versioning scheme is not adequate for protocol designers making use of JSContact profiles, then an alternative approach is to register a new JSContact profile for each new version.

3.3. Profile Properties

A profile defines a list of property entries that together determine the set of properties supported by that profile, as described in Section 3.4. The list MUST NOT be empty.

Each property entry consists of the following elements:

Property Name: This is the name of the JSContact property that this entry refers to. This MUST be a property name registered in the "JSContact Properties" registry. A property name MAY occur multiple times in the property entry list if the profile-specific restrictions for that property can not be expressed with a single entry, but such multiple entries MUST NOT result in conflicting restrictions for the same property. This field MUST NOT be empty.

Property Context: This is a comma-separated list of JSContact object types that support this property in this profile. Each of these types MUST also be listed in the property contexts for this property in the "JSContact Properties" registry, but a profile MAY only support the property in a subset of these contexts. This field MUST NOT be empty.

Restricted Attributes: This restricts the attributes of the property. This specification only defines how to restrict the attributes such that a property becomes mandatory in the listed property contexts, despite the property originally being defined to be optional in the same contexts. The verbatim value "mandatory" (without quotes) indicates that it is mandatory in this profile, the absence of any value indicates that this profile does not restrict the property attributes of the original definition.

Restricted Property Type: This restricts the property value type in the listed property contexts, even though the property was originally defined with a less restrictive type definition. If set, the original value type MUST contain some type signature in form "A|B" and the restricted value type MUST resemble the original except that some of the original "A|B" forms now only allow a subset of the original choices. Restricted property types can not redefine the "defaultType" attribute [RFC9553] (Section 1.3.3), the rules when to set the "@type" property [RFC9553] (Section 1.3.4) of the original type definition still apply.

For example, one might want to restrict the "date" property of the Anniversary object to only allow partial dates as values. To do so, the original value type "PartialDate|Timestamp" can be

restricted to "PartialDate". Note that the original value type need not be exactly in form "A|B". For example, the type signature "Id[A|B|C]" could be restricted to any of "Id[A|B]", "Id[A|C]", "Id[B|C]", "Id[A]", "Id[B]", "Id[C]", and the type signature "(A|B)[]" to one of "A[]" or "B[]".

Restricted Enum Values: This restricts the enumerated values defined for this property to a subset of those values. The values MUST be listed in the "JSContact Enum Values" registry for this property and context. Allowed values are separated by comma; the absence of any value indicates that all enumerated values are allowed. A profile MAY exclude the default enumerated value of a property, in which case all instances of the object MUST have this property set to one of the allowed values.

Restricted PatchObject: This restricts the PatchObject value of this property such that each JSON Pointer key in the PatchObject MUST consist of exactly one JSON Pointer reference token [RFC6901] (Section 3). For example, with this restriction the "localizations" property of the Card object can only patch properties of the Card object by replacing their values entirely. The verbatim value "yes" (without quotes) indicates that this profile restricts PatchObject keys to a single token; the absence of any value indicates that it does not restrict them. This MUST NOT be set to "yes" if the property value type is not a PatchObject.

All profiles MUST support "@type" and "version", and therefore profiles MUST NOT include entries for these properties.

3.4. Supported Properties

The supported properties of a JSContact profile are determined by the profile’s property entries and the contents of the IANA "JSContact Properties" registry, which is here referred to as "IANA-registered properties" for short. The "version" property of the Card object and the "@type" property of any object type are always supported.

A Card object complies with the profile if all its properties are part of the supported properties and all property values are valid according to the restrictions defined in the applicable property entries. A PatchObject MUST NOT patch properties that are not supported in that profile.

The following describes the steps to determine the supported properties:

1. Initialize the set with all properties of the Card object for which a property entry contains "Card" in the Property Context of the profile. If no such entry exists then initialize the set with all IANA-registered properties of the Card object.

2. For every property in the set having either an object type as value type, or a list or map or union of object types, add all properties for which a property entry contains those object types in the Property Context of the profile. If no such property entry exists then add all IANA-registered properties for the object types.

3. Repeat the previous step until all object types that are value types of properties in the set have been considered.

Appendix A describes how to determine the supported properties of the example profile in Section 4.

4. Example Profile

This section provides an example for a JSContact profile and illustrates how a JSContact Card complies with that profile. Appendix A describes how to determine the supported properties for that profile. The example profile is defined the same as if it would be registered in the IANA JSContact Profiles registry. See Section 5 for the exact meaning of each registry item. This profile is just for illustration, it is not registered at IANA.

The following assignments would be registered at IANA if this were a real profile:

Name:
   jscontact-simple

Version:
   1

Reference:
   This document, Table 1

The following table defines the properties of that profile. This profile does not specify any "Restricted Property Type" and the according column is omitted.

Property Name	Property Context	Restricted Attributes	Restricted Enum Values	Restricted PatchObject
addresses	Card			
emails	Card			
kind	Card		individual,org	
localizations	Card			yes
name	Card			
full	Address	mandatory		
components	Name			
full	Name			
kind	NameComponent			
value	NameComponent			

Table 1: Properties of the "jscontact-simple" profile

This profile describes contact cards that can only contain:

* Contact cards for individuals and organizations, but no other kinds such as groups or devices.

* Full postal address lines, but no address components or any other property of the Address object type.

* Full names and name components, but no other properties of the Name object type.

* Email addresses, where all properties of the EmailAddress object are supported.

* Localizations, with the restriction that the PatchObject must only patch properties of the Card object.

The following Card object complies with the example profile:

```
{
  "@type": "Card",
  "version": "1.0",
  "name": {
     "components": [
      { "kind": "given", "value": "Hayao" },
      { "kind": "surname", "value": "Miyazaki" }
    ]
  },
  "addresses": {
    "a1": {
      "full": "71 Cherry Court, Somewhere, 123SO, UK"
    }
  },
  "emails": {
    "e1": {
      "address": "hayao@example.com"
    }
  },
  "localizations": {
    "jp": {
      "name": {
        "components": [
         { "kind": "surname", "value": "宮崎" },
         { "kind": "given", "value": "駿" }
        ]
      }
    }
  }
}
```

Note that:

* The Address, Card, Name, and NameComponent object values only contain properties for which a property entry exists in the profile.

* The EmailAddress object contains the "address" property. This is allowed because the profile contains a property entry for the "emails" property of the Card object, but it does not define entries for the EmailAddress object properties. Consequently, all properties of the EmailAddress object can be set.

* The "full" property of the Address object is set. It is the only property allowed to be set in that profile for Address, and the "full" property is mandatory for this profile.

* The "kind" property of the the NameComponent objects is set to "surname" and "given", respectively. Since the profile does not restrict the enumerated values of this property, all valid NameComponent "kind" property values are supported.

* The "kind" property of the Card object is not set. The profile restricts the enumerated values of this property to "individual" and "org". Since "individual" is the default value for this property, there is no need to set it.

5. IANA Considerations

5.1. Creation of the JSContact Profile Registry

IANA will add the "JSContact Profile" registry to the "JSContact" registry group. The purpose of this new registry is to register profiles for JSContact data. The registry policy to add a new profile to this registry is "Specification Required", while the registry policy to update an existing profile is "Expert Review". The change controller is IETF.

Designated experts assert that all proposed assignments are valid according to the definitions in this document, e.g. they check that the profile name is unique, that the entries in the profile entries are syntactically valid, that only known property names are referred to, that type restrictions do not incorrectly alter the type of a property, and that the version number increases. On the other hand, designated experts do not decide the contents of the profile as long as the assignments are valid.

An entry in this registry consists of the following assignments, all of which MUST be set:

Name:
: This is the name of the profile. This field is immutable after creation. The name MUST be unique among all registered profiles and MUST comply with the definitions in Section 3.1.

Version:
: This is the version number of the profile. It MUST comply with the definitions in Section 3.2.

Reference:
: This refers to the specifications of the protocol or use case for which this profile applies. The reference MUST must include the section number or name that defines or updates the property entries for this profile, as defined in Section 3.3.

5.2. Initial Contents of the JSContact Profile Registry

This document does not define any initial contents for the newly created registries.

6. Security Considerations

This document does not provide new security considerations. The security considerations of Section 4 of [RFC9553] apply.

7. References

7.1. Normative References

[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/info/rfc2119>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <https://www.rfc-editor.org/info/rfc6901>.

[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/info/rfc8174>.

[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/info/rfc9553>.

7.2. Informative References

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <https://www.rfc-editor.org/info/rfc6350>.

[RFC6352] 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/info/rfc6352>.

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

Appendix A. Supported Properties Example

The following illustrates how to determine the supported properties of the example profile (Section 4), according to the steps defined in Section 3.4:

1. We initialize the set of supported properties with all profile properties for which the property context includes the "Card" object. The set now includes the properties:

   * Card.addresses
   * Card.emails
   * Card.kind
   * Card.localizations
   * Card.name

2. Next, we determine which of the properties in the current set have an object type as value type. These are the "Card.address", "Card.emails" and "Card.name" properties, so we need to inspect the object types "Address", "EmailAddress" and "Name".

3. For the "Address" object type, the profile has an entry for the "full" property which contains the "Address" object in the Property Context, so we add that and only that to the set of supported properties. It now contains:

   * Address.full
   * Card.addresses
   * Card.emails
   * Card.kind
   * Card.localizations
   * Card.name

The value type of the "Address.full" property is not an object type, so we need not consider a new object type to inspect. The remaining object types to inspect are the "EmailAddress" and "Name" objects.

4. For the "EmailAddress" object type, the profile does not contain any entry where the Property Context contains the "EmailAddress" object. Instead, we add properties of the "JSContact Properties" registry where the property context includes the "EmailAddress" object. The set of supported properties now contains:

   * Address.full
   * Card.addresses
   * Card.emails
   * Card.kind
   * Card.localizations
   * Card.name
   * EmailAddress.address
   * EmailAddress.contexts
   * EmailAddress.label
   * EmailAddress.pref

   None of the newly added properties have object value types. The remaining object type to inspect is the "Name" object.

5. For the "Name" object type, the profile explicitly lists the "components" and "full" properties, so we add them to the set of supported properties. It now contains:

   * Address.full
   * Card.addresses
   * Card.emails
   * Card.kind
   * Card.localizations

* Card.name
* EmailAddress.address
* EmailAddress.contexts
* EmailAddress.label
* EmailAddress.pref
* Name.full
* Name.components

The newly added Name.components property has object value type "NameComponent", so we add that to the list of object types to inspect.

6. For the "NameComponent" object type, the profile does not explicitly list any property. Instead, we add properties of the "JSContact Properties" registry where the property context includes the "NameComponent" object. The set of supported properties now contains:

* Address.full
* Card.addresses
* Card.emails
* Card.kind
* Card.localizations
* Card.name
* EmailAddress.address
* EmailAddress.contexts
* EmailAddress.label
* EmailAddress.pref
* NameComponent.kind
* NameComponent.phonetic

* NameComponent.value

None of the newly added properties have object value types and there are no remaining object types to inspect. This determines the set of supported properties by that profile, in addition to the "Card.version" and "@type" property that are always supported.

Authors' Addresses

Robert Stepanek
Fastmail
PO Box 234
Collins St. West
Melbourne VIC 8007
Australia
Email: rsto@fastmailteam.com

Mario Loffredo
IIT-CNR
Via Moruzzi, 1
56124 Pisa
Italy
Email: mario.loffredo@iit.cnr.it