JMAP
Internet-Draft
Intended status: Standards Track
Expires: December 17, 2020

R. Stepanek
FastMail
M. Loffredo
IIT-CNR

June 15, 2020

JSContact: A JSON representation of contact data
draft-ietf-jmap-jscontact-02

Abstract

This specification defines a data model and JSON representation of contact card information that can be used for data storage and exchange in address book or directory applications. It aims to be an alternative to the vCard data format and to be unambiguous, extendable and simple to process. In contrast to the JSON-based jCard format, it is not a direct mapping from the vCard data model and expands semantics where appropriate.

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 December 17, 2020.

Table of Contents

1. Introduction

This document defines a data model for contact card data normally used in address book or directory applications and services. It aims to be an alternative to the vCard data format [RFC6350] and to provide a JSON-based standard representation of contact card data.

The key design considerations for this data model are as follows:

- Most of the initial set of attributes should be taken from the vCard data format [RFC6350] and extensions ([RFC6473], [RFC6474], [RFC6715], [RFC6869], [RFC8605]). The specification should add new attributes or value types, or not support existing ones, where appropriate. Conversion between the data formats need not fully preserve semantic meaning.

- The attributes of the cards data represented must be described as a simple key-value pair, reducing complexity of its representation.

- The data model should avoid all ambiguities and make it difficult to make mistakes during implementation.

- Extensions, such as new properties and components, MUST NOT lead to requiring an update to this document.

The representation of this data model is defined in the I-JSON format [RFC7493], which is a strict subset of the JavaScript Object Notation (JSON) Data Interchange Format [RFC8259]. Using JSON is mostly a pragmatic choice: its widespread use makes JSCard easier to adopt, and the availability of production-ready JSON implementations eliminates a whole category of parser-related interoperability issues.

1.1. Relation to the xCard and jCard formats

The xCard [RFC6351] and jCard [RFC7095] specifications define alternative representations for vCard data, in XML and JSON format respectively. Both explicitly aim to not change the underlying data model. Accordingly, they are regarded as equal to vCard in the context of this document.

1.2. Terminology

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.

1.3. Vendor-specific Property Extensions and Values

Vendors MAY add additional properties to JSContact objects to support their custom features. The names of these properties MUST be prefixed with a domain name controlled by the vendor to avoid conflict, e.g. "example.com/customprop".

Some JSContact properties allow vendor-specific value extensions. If so, vendor-specific values MUST be prefixed with a domain name controlled by the vendor, e.g. "example.com/customrel".

Vendors are strongly encouraged to register any new property values or extensions that are useful to other systems as well, rather than using a vendor-specific prefix.

2. JSCard

MIME type: "application/jscontact+json;type=jscard"

A JSCard object stores information about a person, organization or company.

2.1. Metadata properties

2.1.1. uid

Type: "String" (mandatory).

An identifier, used to associate the object as the same across different systems, addressbooks and views. [RFC4122] describes a range of established algorithms to generate universally unique identifiers (UUID), and the random or pseudo-random version is recommended. For compatibility with [RFC6350] UIDs, implementations MUST accept both URI and free-form text.

2.1.2. prodId

Type: "String" (optional).

The identifier for the product that created the JSCard object.

2.1.3. updated

Type: "String" (optional).

The date and time when the data in this JSCard object was last modified. The timestamp MUST be formatted as specified in [RFC3339].

2.1.4. kind

Type: "String" (optional). The kind of the entity the Card represents.

The value MUST be either one of the following values, registered in a future RFC, or a vendor-specific value:

- "individual": a single person
- "org": an organization
- "location": a named location
- "device": a device, such as appliances, computers, or network elements
- "application": a software application

2.1.5. relatedTo

Type: "String[Relation]" (optional).

Relates the object to other JSCard objects. This is represented as a map of the URI (or single text value) of the related objects to a possibly empty set of relation types. The Relation object has the following properties:

- relation: "String[Boolean]" (optional, default: empty Object) Describes how the linked object is related to the linking object. The relation is defined as a set of relation types. If empty, the relationship between the two objects is unspecified. Keys in the set MUST be one of the RELATED property [RFC6350] type parameter values, or an IANA-registered value, or a vendor-specific value. The value for each key in the set MUST be true.

Note, the Relation object only has one property; it is specified as an object with a single property to allow for extension in the future.

2.2. Name and Organization properties

2.2.1. fullName

Type: "LocalizedString" (optional).

The full name (e.g. the personal name and surname of an individual, the name of an organization) of the entity represented by this card.

2.2.2. name

Type: "NameComponent[]" (optional).

The name components of the name of the entity represented by this JSCard. Name components SHOULD be ordered such that their values joined by whitespace produce a valid full name of this entity.

A NameComponent has the following properties:

- value: "String" (mandatory). The value of this name component.
- type: "String" (mandatory). The type of this name component. Valid name component types are:
  - "prefix". The value is a honorific title(s), e.g. "Mr", "Ms", "Dr".
  - "personal". The value is a personal name(s), also known as "first name", "given name".
  - "surname". The value is a surname, also known as "last name", "family name".
  - "additional". The value is an additional name, also known as "middle name".
  - "suffix". The value is a honorific suffix, e.g. "B.A.", "Esq.".
  - "nickname". The value is a nickname.

2.2.3. organization

Type: "LocalizedString[]" (optional).

The company or organization name and units associated with this card. The first entry in the list names the organization, and any following entries name organizational units.

2.2.4. jobTitle

Type : "LocalizedString[]" (optional).

The job title(s) or functional position(s) of the entity represented by this card.

2.2.5. role

Type : "LocalizedString[]" (optional).

The role(s), function(s) or part(s) played in a particular situation by the entity represented by this card. In contrast to a job title, the roles might differ for example in project contexts.

2.3. Contact and Resource properties

2.3.1. emails

Type: "Resource[]" (optional).

An array of Resource objects where the values are URLs in the "mailto" scheme [RFC6068] or free-text email addresses. The default value of the "type" property is "email". If set, the type MUST be "email" or "other".

2.3.2. phones

Type: "Resource[]" (optional).

An array of Resource objects where the values are URIs scheme or free-text phone numbers. Typical URI schemes are the [RFC3966] "tel" or [RFC3261] "sip" schemes, but any URI scheme is allowed. Types are:

- "voice" The number is for calling by voice.
- "fax" The number is for sending faxes.
- "pager" The number is for a pager or beeper.
- "other" The number is for some other purpose. A label property MAY be included to display next to the number to help the user identify its purpose.

2.3.3. online

Type: "Resource[]" (optional).

An array of Resource objects where the values are URIs or usernames associated with the card for online services. Types are:

- "uri" The value is a URI, e.g. a website link.
- "username" The value is a username associated with the entity represented by this card (e.g. for social media, or an IM client). A label property SHOULD be included to identify what service this is for. For compatibility between clients, this label SHOULD be the canonical service name, including capitalisation. e.g. "Twitter", "Facebook", "Skype", "GitHub", "XMPP".
- "other" The value is something else not covered by the above categories. A label property MAY be included to display next to the number to help the user identify its purpose.

2.3.4. preferredContactMethod

Type : "String" (optional)

Defines the preferred contact method or resource with additional information about this card. The value MUST be the property name of one of the Resource lists: "emails", "phones", "online", "other".

2.3.5. preferredContactLanguages

Type : "String[ContactLanguage[]]" (optional)

Defines the preferred languages for contacting the entity associated with this card. The keys in the object MUST be [RFC5646] language tags. The values are a (possibly empty) list of contact language preferences for this language. Also see the definition of the VCARD LANG property (Section 6.4.4., [RFC6350]).

A ContactLanguage object has the following properties:

- type: "String" (optional). Defines the context of this preference. This could be "work", "home" or another value.
- preference: "Number" (optional). Defines the preference order of this language for the context defined in the type property. If set, the property value MUST be between 1 and 100 (inclusive). Lower values correspond to a higher level of preference, with 1 being most preferred. If not set, the default MUST be to

interpret the language as the least preferred in its context. Preference orders SHOULD be unique across language for a specific type.

A valid ContactLanguage object MUST have at least one of its properties set.

2.4. Address and Location properties

2.4.1. addresses

Type: Address[] (optional).

An array of Address objects, containing physical locations. An Address object has the following properties:

- context: "String" (optional, default "other"). Specifies the context of the address information. The value MUST be either one of the following values, registered in a future RFC, or a vendor-specific value:
  - "private" An address of a residence.
  - "work" An address of a workplace.
  - "billing" An address to be used for billing.
  - "postal" An address to be used for delivering physical items.
  - "other" An address not covered by the above categories.
- label: "String" (optional). A label describing the value in more detail.
- fullAddress: "LocalizedString" (optional). The complete address, excluding type and label. This property is mainly useful to represent addresses of which the individual address components are unknown, or to provide localized representations.
- street: "String" (optional). The street address. This MAY be multiple lines; newlines MUST be preserved.
- extension: "String" (optional) The extended address, such as an apartment or suite number, or care-of address.
- locality: "String" (optional). The city, town, village, post town, or other locality within which the street address may be found.

- region: "String" (optional). The province, such as a state, county, or canton within which the locality may be found.

- country: "String" (optional). The country name.

- postOfficeBox: "String" (optional) The post office box.

- postcode: "String" (optional). The postal code, post code, ZIP code or other short code associated with the address by the relevant country's postal system.

- countryCode: "String" (optional). The ISO-3166-1 country code.

- coordinates: "String" (optional) A [RFC5870] "geo:" URI for the address.

- timeZone: "String" (optional) Identifies the time zone this address is located in. This SHOULD be a time zone name registered in the IANA Time Zone Database [1]. Unknown time zone identifiers MAY be ignored by implementations.

- isPreferred: Boolean (optional, default: false). Whether this Address is the preferred for its type. This SHOULD only be one per type.

2.5. Additional properties

2.5.1. anniversaries

Type : Anniversary[] (optional).

Memorable dates and events for the entity represented by this card. An Anniversary object has the following properties:

- type: "String" (mandatory). Specifies the type of the anniversary. This RFC predefines the following types, but implementations MAY use additional values:

  * "birth": a birth day anniversary

  * "death": a death day anniversary

  * "other": an anniversary not covered by any of the known types.

- label: "String" (optional). A label describing the value in more detail, especially if the type property has value "other" (but MAY be included with any type).

- date: "String" (mandatory). The date of this anniversary, in the form "YYYY-MM-DD" (any part may be all 0s for unknown) or a [RFC3339] timestamp.

- place: Address (optional). An address associated with this anniversary, e.g. the place of birth or death.

2.5.2. personalInfo

Type: PersonalInformation[] (optional).

A list of personal information about the entity represented by this card. A PersonalInformation object has the following properties:

- type: "String" (mandatory). Specifies the type for this personal information. Allowed values are:

  * "expertise": a field of expertise or credential

  * "hobby": a hobby

  * "interest": an interest

  * "other": an information not covered by the above categories

- value: "String" (mandatory). The actual information. This generally is free-text, but future specifications MAY restrict allowed values depending on the type of this PersonalInformation.

- level: "String" (optional) Indicates the level of expertise, or engagement in hobby or interest. Allowed values are: "high", "medium" and "low".

2.5.3. notes

Type: "LocalizedString[]" (optional).

Arbitrary notes about the entity represented by this card.

2.5.4. categories

Type: "String[]" (optional). A list of free-text or URI categories that relate to the card.

2.6. Common JSCard types

2.6.1. LocalizedString

A LocalizedString object has the following properties:

- value: "String" (mandatory). The property value.
- language: "String" (optional). The [RFC5646] language tag of this value, if any.
- localizations: "String[String]" (optional). A map from [RFC5646] language tags to the value localized in that language.

2.6.2. Resource

A Resource object has the following properties:

- context: "String" (optional) Specifies the context in which to use this resource. Pre-defined values are:
  - "private": The resource may be used to contact the card holder in a private context.
  - "work": The resource may be used to contact the card holder in a professional context.
  - "other": The resource may be used to contact the card holder in some other context. A label property MAY be help to identify its purpose.
- type: "String" (optional). Specifies the property-specific variant of the resource. This MUST be taken from the set of allowed types specified in the respective contact method property.
- labels: "String[Boolean]" (optional). A set of labels that describe the value in more detail, especially if the type property has value "other" (but MAY be included with any type). The keys in the map define the label, the values MUST be "true".
- value: "String" (mandatory). The actual resource value, e.g. an email address or phone number.
- mediaType: "String" (optional). Used for properties with URI values. Provides the media type [RFC2046] of the resource identified by the URI.

- isPreferred: Boolean (optional, default: false). Whether this resource is the preferred for its type. This SHOULD only be one per type.

3. JSCardGroup

MIME type: "application/jscontact+json;type=jscardgroup"

A JSCardGroup object represents a named set of JSCards.

3.1. Properties

3.1.1. uid

Type : "String" (mandatory).

A globally unique identifier. The same requirements as for the JSCard uid property apply.

3.1.2. name

Type: "String" (optional).

The user-visible name for the group, e.g. "Friends". This may be any UTF-8 string of at least 1 character in length and maximum 255 octets in size. The same name may be used by two different groups.

3.1.3. cards

Type : "JSCard[]" (mandatory). The cards in the group. Implementations MUST preserve the order of list entries.

4. Implementation Status

NOTE: Please remove this section and the reference to [RFC7942] prior to publication as an RFC. 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 [RFC7942]. 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 [RFC7942], "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".

4.1. IIT-CNR/Registro.it

- Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
- Location: https://rdap.pubtest.nic.it/ [2]
- Description: This implementation includes support for RDAP queries using data from the public test environment of .it ccTLD. The RDAP server does not implement any security policy because data returned by this server are only for experimental testing purposes. The RDAP server returns responses including JSCard in place of jCard when queries contain the parameter jscard=1.
- Level of Maturity: This is a "proof of concept" research implementation.
- Coverage: This implementation includes all of the features described in this specification.
- Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

5. IANA Considerations

TBD

6. Security Considerations

TBD

7. References

7.1. Normative References

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[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>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC5870] Mayrhofer, A. and C. Spanring, "A Uniform Resource Identifier for Geographic Locations ('geo' URI)", RFC 5870, DOI 10.17487/RFC5870, June 2010, <https://www.rfc-editor.org/info/rfc5870>.

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

[RFC6351] Perreault, S., "xCard: vCard XML Representation", RFC 6351, DOI 10.17487/RFC6351, August 2011, <https://www.rfc-editor.org/info/rfc6351>.

[RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <https://www.rfc-editor.org/info/rfc7095>.

[RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <https://www.rfc-editor.org/info/rfc7493>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

7.2. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3966] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966, DOI 10.17487/RFC3966, December 2004, <https://www.rfc-editor.org/info/rfc3966>.

[RFC6068] Duerst, M., Masinter, L., and J. Zawinski, "The 'mailto' URI Scheme", RFC 6068, DOI 10.17487/RFC6068, October 2010, <https://www.rfc-editor.org/info/rfc6068>.

[RFC6473] Saint-Andre, P., "vCard KIND:application", RFC 6473, DOI 10.17487/RFC6473, December 2011, <https://www.rfc-editor.org/info/rfc6473>.

[RFC6474] Li, K. and B. Leiba, "vCard Format Extensions: Place of Birth, Place and Date of Death", RFC 6474, DOI 10.17487/RFC6474, December 2011, <https://www.rfc-editor.org/info/rfc6474>.

[RFC6715] Cauchie, D., Leiba, B., and K. Li, "vCard Format Extensions: Representing vCard Extensions Defined by the Open Mobile Alliance (OMA) Converged Address Book (CAB) Group", RFC 6715, DOI 10.17487/RFC6715, August 2012, <https://www.rfc-editor.org/info/rfc6715>.

[RFC6869] Salgueiro, G., Clarke, J., and P. Saint-Andre, "vCard KIND:device", RFC 6869, DOI 10.17487/RFC6869, February 2013, <https://www.rfc-editor.org/info/rfc6869>.

[RFC8605] Hollenbeck, S. and R. Carney, "vCard Format Extensions: ICANN Extensions for the Registration Data Access Protocol (RDAP)", RFC 8605, DOI 10.17487/RFC8605, May 2019, <https://www.rfc-editor.org/info/rfc8605>.

7.3. URIs

[1] https://www.iana.org/time-zones

[2] https://rdap.pubtest.nic.it/

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
Pisa, 56124
Italy

Email: mario.loffredo@iit.cnr.it