JSContact: A JSON representation of contact data
draft-stepanek-jscontact-05

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 May 1, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

Stepanek & Loffredo Expires May 1, 2020 [Page 1]
include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ......................................................... 3
   1.1. Relation to the xCard and jCard formats ................... 3
   1.2. Terminology .................................................. 4
2. JSCard ................................................................. 4
   2.1. Metadata properties ........................................... 4
       2.1.1. uid ....................................................... 4
       2.1.2. prodId ................................................... 4
       2.1.3. updated .................................................. 4
       2.1.4. kind ..................................................... 4
       2.1.5. relatedTo ................................................ 5
   2.2. Name and Organization properties ............................ 5
       2.2.1. name ...................................................... 5
       2.2.2. organization ............................................ 6
       2.2.3. jobTitle ................................................ 6
       2.2.4. role ....................................................... 6
   2.3. Contact and Resource properties .............................. 6
       2.3.1. emails ................................................... 6
       2.3.2. phones ................................................. 7
       2.3.3. online .................................................. 7
       2.3.4. preferredContactMethod ................................ 8
   2.4. Address and Location properties .............................. 8
       2.4.1. addresses ................................................ 8
   2.5. Additional properties ........................................... 9
       2.5.1. anniversaries ........................................... 9
       2.5.2. personalInfo ........................................... 10
       2.5.3. notes .................................................... 10
       2.5.4. categories .............................................. 10
   2.6. Common JSCard types .......................................... 11
       2.6.1.LocalizedString ........................................... 11
       2.6.2. Resource ................................................ 11
3. JSCardGroup .......................................................... 11
   3.1. Properties ...................................................... 11
       3.1.1. uid ....................................................... 12
       3.1.2. name ...................................................... 12
       3.1.3. cardIds .................................................. 12
4. Implementation Status .............................................. 12
   4.1. IIT-CNR/Registro.it ........................................... 12
5. IANA Considerations ................................................. 13
6. Security Considerations ............................................ 13
7. References ........................................................... 13
   7.1. Normative References .......................................... 13
   7.2. Informative References ...................................... 14
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.

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" (mandatory).

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:
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. name

Type: Name (mandatory).

The name of the entity represented by this JSCard. A Name object has the following properties:

- fullName: "LocalizedString" (mandatory). 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.

- prefix: "String[]" (optional). The honorific title(s), e.g. "Mr", "Ms", "Dr".
o personalName: "String[]" (optional). The personal name(s), also known as "first name", "give name".

o surname: "String[]" (optional). The surname(s) (also known as "last name", "family name").

o additionalName: "String[]" (optional). The additional name(s), also known as "middle name".

o suffix: "String[]" (optional). The honorific suffix(es), e.g. "B.A.", "Esq.".

o nickname: "String[]" (optional). The nickname(s) of the person represented by this card.

2.2.2. 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.

2.2.3. jobTitle

Type: "LocalizedString[]" (optional).

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

2.2.4. 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 [RFC6068] "mailto" scheme or free-text email addresses. Types are:

o "personal" The address is for emailing in a personal context.
"work" The address is for emailing in a professional context.

"other" The address is for some other purpose. A label property MAY be included to display next to the address to help the user identify its purpose.

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. Resource 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.

The following labels are pre-defined for phone resources:

"private" The phone number should be used in a private context.

"work" The phone number should be used in a professional context

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".
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".

2.4. 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:

- type: "String" (mandatory). 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:
  - "home" 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.
o locality: "String" (optional). The city, town, village, post
town, or other locality within which the street address may be
found.

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

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

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

o 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.

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

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

o 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.

o 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:

o 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.
o  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.

o  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.

2.6.2. Resource

A Resource object has the following properties:

- type: "String" (mandatory). Specifies the context of the resource. This MUST be taken from the set of values specified for the respective property.
- 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).
- 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. cardIds

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


7.2. Informative References


7.3. URIs

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


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