JSContact: A JSON representation of addressbook data
draft-stepanek-jscontact-03

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 31, 2019.

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...
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] 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. It has the following properties:

- **uid**: 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.

- **prodId**: String (optional). The identifier for the product that created the JSCard object.

- **updated**: 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].

- **kind**: 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

fullName: FullName[] (mandatory). The full name(s) of the entity represented by this card. A FullName object has the following properties:

  * name: String (mandatory) The full name (e.g. the personal name and surname of an individual, the name of an organization).
  * language: String (optional) The [RFC5646] language tag of this name, if any.
  * isPreferred: Boolean (optional, default: "false"). Whether this FullName is the preferred name.

structuredName: StructuredName (optional). The name of the person represented by this card, structured by its constituents. A StructuredName object has the following properties:

  * prefix: String[] (optional). The honorific title(s), e.g. "Mr", "Ms", "Dr".
  * personalName: String[] (optional). The personal name(s), also known as "first name", "give name".
  * surname: String[] (optional). The surname(s) (also known as "last name", "family name").
  * additionalName: String[] (optional). The additional name(s), also known as "middle name".
  * suffix: String[] (optional). The honorific suffix(es), e.g. "B.A.", "Esq.".

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

anniversaries: 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.

* 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.

  o organization: String[] (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.

  o jobTitle[]: String (optional). The job title(s) or functional position(s) of the entity represented by this card.

  o role[]: String (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.

  o emails: ContactMethod[] (optional). An array of ContactMethod objects where the values are URLs in the [RFC2368] "mailto" scheme or free-text email addresses. Types are:
    * "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.

  o phones: ContactMethod[] (optional). An array of ContactMethod 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. Contact method 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 contact methods:

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

- online: ContactMethod[] (optional). An array of ContactMethod 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.

- preferredContactMethod: String (optional) Defines the preferred contact method. The value MUST be the property name of one of the ContactMethod lists: "emails", "phones", "online", "other".

- addresses: Address[] (optional). An array of Address objects, containing physical locations.

- personalInfo: 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".

notes: String (optional). Arbitrary notes about the entity represented by this card.

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

A ContactMethod object has the following properties:

* type: String (mandatory). Specifies the context of the contact method. This MUST be taken from the set of values allowed depending on whether this is part of the phones, emails or online property (see above).

* 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 contact method, e.g. the email address or phone number.

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

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.

  o label: String (optional). A label describing the value in more detail.
  o fullAddress: String (optional). The complete address, excluding type and label. This property is mainly useful to represent addresses of which the individual address components are unknown.
  o street: String (optional). The street address. This MAY be multiple lines; newlines MUST be preserved.
  o extension: String (optional) The extended address, such as an apartment or suite number, or care-of address.
  o postOfficeBox: String (optional) The post office box.
  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 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 country: String (optional). The country name.
  o countryCode: String (optional). The ISO-3166-1 country code.
  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.
3. JSCardGroup

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

A JSCardGroup object represents a named set of JSCards. It has the following properties:

- **uid**: String (mandatory). A globally unique identifier. The same requirements as for the JSCard uid property apply.
- **name**: 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.
- **cardIds**: String[] (mandatory). The ids of the cards in the group. Implementations MUST preserve the order of list entries.

4. IANA Considerations

TBD

5. Security Considerations

TBD

6. References

6.1. Normative References


6.2. Informative References


6.3. 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