| | | |
|---|---|---|
| Stream: | Internet Engineering Task Force (IETF) | |
| RFC: | 9555 | |
| Updates: | 6350 | |
| Category: | Standards Track | |
| Published: | May 2024 | |
| ISSN: | 2070-1721 | |
| Authors: | M. Loffredo | R. Stepanek |
| | *IIT-CNR/Registro.it* | *Fastmail* |

# RFC 9555
# JSContact: Converting from and to vCard

## Abstract

This document defines how to convert contact information between the JSContact and vCard data formats. It defines conversion rules for every JSContact and vCard element registered at IANA at the time of publication. It also defines new JSContact properties as well as vCard properties and parameters, to support converting arbitrary or unknown JSContact and vCard elements.

## Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9555.

## Table of Contents

# 1. Introduction

## 1.1. Motivation

The JSContact data model and format [RFC9553] aims to be an alternative to the widely used vCard standard [RFC6350] and jCard format [RFC7095].

While applications might prefer JSContact to exchange contact card data with other systems, they are likely to interoperate with services and clients that only support vCard or jCard. Similarly, existing contact data providers and consumers already using vCard or jCard might also want to represent their contact data in JSContact.

To achieve this, this document defines standard rules to convert contact data between JSContact and vCard (and consequently jCard).

## 1.2. 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.

### 1.3. ABNF Notations

The ABNF definitions in this document use the notations of [RFC5234]. ABNF rules not defined in this document are defined in either [RFC5234] (such as the ABNF for CRLF, WSP, DQUOTE, VCHAR, ALPHA, and DIGIT) or [RFC6350].

## 2. Converting vCard to JSContact

This section contains the conversion rules from the vCard to the JSContact Card. It follows the same structure as vCard v4 [RFC6350]. Properties and parameters of vCard extension RFCs, including those described in "vCard Format Extension for JSContact" [RFC9554], have been added to the appropriate subsections.

### 2.1. General Rules

#### 2.1.1. The Card uid Property

The UID property (Section 6.7.6 of [RFC6350]) in vCard is optional, but the Card object's uid property (Section 2.1.9 of [RFC9553]) is mandatory. Implementations that convert a vCard without a UID property **MUST** generate a unique identifier as value for the uid property. This value **SHOULD** be the same when converting the same vCard multiple times, but how to achieve this is implementation-specific.

#### 2.1.2. Choosing Identifiers

Multivalued properties in JSContact are typically represented as a JSON object where the object keys are of the Id type (Section 1.4.1 of [RFC9553]) and the object values are the converted vCard property. In the absence of the PROP-ID parameter (see Section 2.3.18), implementations are free to choose any identifier as key for such entries. Whatever identifier generation scheme implementations use, they **MUST** generate values that are valid according to the definition of the Id type in [RFC9553]. For example, this could be an incrementing number across all identifier keys in the Card object or only unique within one JSON object.

### 2.2. vCard Value Data Types

#### 2.2.1. BOOLEAN

The BOOLEAN type (Section 4.4 of [RFC6350]) converts to the JSContact Boolean type (Section 1.3.2 of [RFC9553]).

#### 2.2.2. DATE, TIME, DATE-TIME, DATE-AND-OR-TIME, and TIMESTAMP

The TIMESTAMP type (Section 4.3.5 of [RFC6350]) converts to the UTCDateTime type (Section 1.4.5 of [RFC9553]), except for anniversaries. For anniversaries, it converts to the Timestamp type (Section 2.8.1 of [RFC9553]).

The DATE type (Section 4.3.1 of [RFC6350]) converts to a PartialDate object (Section 2.8.1 of [RFC9553]) when used for an anniversary, unless the DATE value only contains a month or a day (but not both).

The following temporal types do not convert to a JSContact datetime type. Instead, vCard properties or parameters having such value types convert as defined in Section 2.15.

- TIME (Section 4.3.2 of [RFC6350])
- DATE-TIME (Section 4.3.3 of [RFC6350])
- DATE-AND-OR-TIME (Section 4.3.4 of [RFC6350])
- DATE type values that only define a month or day (but not both)

#### 2.2.3. INTEGER

The INTEGER type (Section 4.5 of [RFC6350]) converts to the JSContact Int and UnsignedInt types (Section 1.4.2 of [RFC9553]).

#### 2.2.4. FLOAT

The FLOAT type (Section 4.6 of [RFC6350]) converts to the JSContact Number type (Section 1.3.2 of [RFC9553]).

#### 2.2.5. LANGUAGE-TAG

The LANGUAGE-TAG type (Section 4.8 of [RFC6350]) converts to the JSContact String type (Section 1.3.2 of [RFC9553]). The value **MUST** be a language tag as defined in [RFC5646].

#### 2.2.6. TEXT

The TEXT type (Section 4.1 of [RFC6350]) converts to the JSContact String type (Section 1.3.2 of [RFC9553]).

#### 2.2.7. URI

The URI type (Section 4.2 of [RFC6350]) converts to the JSContact String type (Section 1.3.2 of [RFC9553]). The value **MUST** be a URI as defined in Section 3 of [RFC3986]

#### 2.2.8. UTC-OFFSET

The UTC-OFFSET type (Section 4.7 of [RFC6350]) either converts to a String value containing an IANA Time Zone Database entry name (see Section 2.8.2) or does not convert to any JSContact type. For the latter, vCard properties or parameters having such values convert as defined in Section 2.15.

### 2.3. vCard Parameters

This section contains the conversion rules for vCard parameters. A rule typically applies only for specific vCard properties. To convert a vCard parameter on an arbitrary vCard property, see Section 2.15.2.

#### 2.3.1. ALTID

The ALTID parameter (Section 5.4 of [RFC6350]) does not convert to an IANA-registered property in JSContact, but several conversion rules make use of this parameter to combine multiple vCard properties into a single JSContact object instance. For an example of this, see Section 2.6.1. To preserve the verbatim value of the ALTID parameter, set the JSContact properties defined in Section 2.15.

#### 2.3.2. AUTHOR

The AUTHOR parameter (Section 4.1 of [RFC9554]) on a NOTE property converts to the Author object's uri property (Section 2.8.3 of [RFC9553]). That Author object is set as the value of the Note object's author property (Section 2.8.3 of [RFC9553]).

#### 2.3.3. AUTHOR-NAME

The AUTHOR-NAME parameter (Section 4.2 of [RFC9554]) on a NOTE property converts to the Author object's name property (Section 2.8.3 of [RFC9553]). That Author object is set as the value of the Note object's author property.

#### 2.3.4. CALSCALE

The CALSCALE parameter (Section 5.8 of [RFC6350]) set on a BDAY, DEATHDATE, or ANNIVERSARY property converts to the PartialDate object's calendarScale property (Section 2.8.1 of [RFC9553]).

#### 2.3.5. CC

The CC parameter (Section 3.1 of [RFC8605]) on an ADR property converts to the Address object's countryCode property (Section 2.5.1.1 of [RFC9553]).

#### 2.3.6. CREATED

The CREATED parameter (Section 4.3 of [RFC9554]) on a NOTE property converts to the Note object's created property (Section 2.8.3 of [RFC9553]).

#### 2.3.7. DERIVED

The DERIVED parameter (Section 4.4 of [RFC9554]) does not convert to JSContact. If the DERIVED parameter is set to "true" on a vCard property, then implementations **MAY** choose not to convert that property.

#### 2.3.8. GEO

The GEO parameter (Section 5.10 of [RFC6350]) set on an ADR property converts to the Address object's coordinates property (Section 2.5.1.1 of [RFC9553]).

#### 2.3.9. GROUP

The GROUP parameter (Section 7.1 of [RFC7095]) does not convert to JSContact. It exclusively is for use in jCard and **MUST NOT** be set in a vCard.

Preserving the exact group name when converting from vCard to JSContact and back to vCard is not necessary. Any group identifiers will do, as long as the resulting vCard groups its properties equally to the original vCard. Implementations that still wish to preserve the exact property group name of a vCard property **MAY** set the jCard "group" parameter in the JSContact properties vCardProps or vCardParams as defined in Section 2.15.

```
item1.TEL;VALUE=uri:tel:+1-555-555-5555
```

```
"phones": {
  "p1": {
    "number": "tel:+1-555-555-5555",
    "vCardParams" : {
     "group" : "item1"
    }
  }
}
```

*Figure 1: Example of How to Preserve the Group Name in vCardParams during Conversion*

```
item2.X-FOO:bar
```

```
"vCardProps": [
  ["x-foo", {
    "group": "item2"
  }, "unknown", "bar"]
]
```

*Figure 2: Example of How to Preserve the Group Name in vCardProps during Conversion*

#### 2.3.10. INDEX

The INDEX parameter (Section 3.1 of [RFC6715]) set on the EXPERTISE, HOBBY, INTEREST, and ORG-DIRECTORY properties converts to the PersonalInfo (Section 2.8.4 of [RFC9553]) and Directory (Section 2.6.2 of [RFC9553]) objects' listAs property.

#### 2.3.11. LANGUAGE

The LANGUAGE parameter (Section 5.1 of [RFC6350]) converts to an entry in the Card object's localizations property (Section 2.7.1 of [RFC9553]) for that vCard property on which this parameter is set on. The value of the LANGUAGE parameter defines the language tag key in the localizations property.

This specification does not define a single standard conversion rule for how to convert the property values. Instead, building the localizations value is implementation-specific.

Two options to populate the localizations property are:

- One Patch per Property: For each vCard property with a LANGUAGE parameter, set the complete path in the PatchObject to the JSContact property that the vCard property converts to. The value of the patch is the converted property value. This is simple to process and adequate if the vCard only contains a few properties with the LANGUAGE parameter.
- Bundle Patches by Parent: If a PatchObject contains multiple paths that have the same parent paths, then it might be possible to combine these patches into one patch that patches the parent property. This is possible if the property in the Card is patched in its entirety.

Generally, localizations only localize properties that are present in the non-localized version of this Card. Figure 3 illustrates this.

```
FN;LANGUAGE=EN:John Doe
TITLE;ALTID=1;LANGUAGE=EN:Boss
TITLE;ALTID=1;LANGUAGE=fr:Patron
```

```
"language": "en",
"name": {
  "full": "John Doe"
},
"titles": {
  "t1": {
    "name": "Boss"
  }
},
"localizations": {
  "fr": {
    "titles/t1/name": "Patron"
  }
}
```

*Figure 3: LANGUAGE Conversion Example: One Dominant Language*

As a special case, if one or more vCard properties of the same type do not have the LANGUAGE parameter set, add them to the non-localized Card. Convert any with LANGUAGE parameters to the localizations property. Figure 4 illustrates this.

```
FN:John Doe
TITLE;ALTID=1:Boss
TITLE;ALTID=1;LANGUAGE=fr:Patron
```

```
"name": {
  "full": "John Doe"
},
"titles": {
  "t1": {
    "name": "Boss"
  }
},
"localizations": {
  "fr": {
    "titles/t1/name": "Patron"
  }
}
```

*Figure 4: LANGUAGE Conversion Example: Property without Language*

#### 2.3.12. LABEL

The LABEL parameter (Section 6.3.1 of [RFC6350]) on an ADR property converts to the Address object's full property (Section 2.5.1.1 of [RFC9553]).

#### 2.3.13. LEVEL

The LEVEL parameter (Section 3.2 of [RFC6715]) converts to the PersonalInfo object's level property (Section 2.8.4 of [RFC9553]). If this parameter is set on the EXPERTISE property, then its values convert as follows:

- "beginner" converts to "low";
- "average" converts to "medium"; and
- "expert" converts to "high".

In all other cases, the values convert verbatim, but lowercase **MUST** be used for the JSContact value.

#### 2.3.14. MEDIATYPE

The MEDIATYPE parameter (Section 5.7 of [RFC6350]) converts to the Resource object's mediaType property (Section 1.4.4 of [RFC9553]).

#### 2.3.15. PHONETIC

The PHONETIC parameter (Section 4.6 of [RFC9554]) converts to the Name (Section 2.2.1 of [RFC9553]) and Address (Section 2.5.1 of [RFC9553]) objects' phoneticSystem property unless the parameter value is "script", in which case the phoneticSystem property is not set.

The value of the SCRIPT parameter converts to the phoneticScript property (see Section 2.3.19).

The related N or ADR property is defined by the vCard ALTID parameter. The conversion rules for the N (Section 2.5.5) and ADR (Section 2.6.1) properties define how the vCard components convert to JSContact.

The component values of the property on which the PHONETIC parameter is set convert to the respective NameComponent or AddressComponent objects' phonetic properties.

If more than one property has the PHONETIC parameter set and relates to the same property, then they convert to the Card object's localizations property according to their LANGUAGE parameter values as outlined in Section 2.3.11.

```
LANGUAGE=zh-Hant
N;ALTID=1;LANGUAGE=zh-Hant:孫;中山;文,逸仙;;
N;ALTID=1;PHONETIC=jyut;
  SCRIPT=Latn;LANGUAGE=yue:syun1;zung1saan1;man4,jat6sin1;;
```

```
"language": "zh-Hant",
"name": {
  "components": [
    { "kind": "surname", "value": "孫" },
    { "kind": "given", "value": "中山" },
    { "kind": "given2", "value": "文" },
    { "kind": "given2", "value": "逸仙" }
  ]
},
"localizations": {
  "yue": {
    "name/phoneticSystem": "jyut",
    "name/phoneticScript": "Latn",
    "name/components/0/phonetic": "syun1",
    "name/components/1/phonetic": "zung1saan1",
    "name/components/2/phonetic": "man4",
    "name/components/3/phonetic": "jat6sin1"
  }
}
```

*Figure 5: PHONETIC Conversion Example*

#### 2.3.16. PID

The PID parameter (Section 5.5 of [RFC6350]) converts to the vCardParams property; see Section 2.15.2.

#### 2.3.17. PREF

The PREF parameter (Section 5.3 of [RFC6350]) converts to the pref property of the derived JSContact object.

#### 2.3.18. PROP-ID

The PROP-ID parameter (Section 4.7 of [RFC9554]) converts to the Id-typed key of the derived JSContact object.

```
TEL;PROP-ID=PHONE-A;VALUE=uri;PREF=1;TYPE="voice,home"
  :tel:+1-555-555-5555;ext=5555
TEL;PROP-ID=PHONE-B;VALUE=uri;TYPE=home
  :tel:+33-01-23-45-67
```

```
"phones": {
  "PHONE-A": {
    "contexts": { "private": true },
    "features": { "voice": true },
    "number": "tel:+1-555-555-5555;ext=5555",
    "pref": 1
  },
  "PHONE-B": {
    "contexts": { "private": true },
    "number": "tel:+33-01-23-45-67"
  }
}
```

*Figure 6: PROP-ID Conversion Example*

#### 2.3.19. SCRIPT

The SCRIPT parameter (Section 4.8 of [RFC9554]) converts to the Name (Section 2.2.1 of [RFC9553]) or Address (Section 2.5.1 of [RFC9553]) objects' phoneticScript property.

Also see Section 2.3.15.

#### 2.3.20. SERVICE-TYPE

The SERVICE-TYPE parameter (Section 4.9 of [RFC9554]) converts to the OnlineService object's service property (Section 2.3.2 of [RFC9553]).

#### 2.3.21. SORT-AS

The SORT-AS parameter (Section 5.9 of [RFC6350]) converts to the Name, Organization, and OrgUnit objects' sortAs properties.

#### 2.3.22. TYPE

The TYPE parameter (Section 5.6 of [RFC6350]) converts to either the contexts property or the kind property, as defined in later sections. If not otherwise specified, the vCard "home" and "work" parameter values convert to the JSContact "private" and "work" contexts, respectively.

#### 2.3.23. TZ

The TZ parameter (Section 5.11 of [RFC6350]) on an ADR property converts to the Address object's timeZone property (Section 2.5.1.1 of [RFC9553]). Also see the conversion of the TZ property in Section 2.8.2.

#### 2.3.24. USERNAME

The USERNAME parameter (Section 4.10 of [RFC9554]) converts to the OnlineService object's user property (Section 2.3.2 of [RFC9553]).

#### 2.3.25. VALUE

The VALUE parameter (Section 5.2 of [RFC6350]) does not convert to an IANA-registered property in JSContact. To preserve properties with experimental values, see Sections 2.15.1 and 2.15.2.

### 2.4. General Properties

#### 2.4.1. BEGIN and END

The BEGIN and END properties do not convert to IANA-registered properties in JSContact.

#### 2.4.2. KIND

The KIND property (Section 6.1.4 of [RFC6350]) converts to the kind property (Figure 7). Allowed values are those described in Section 6.1.4 of [RFC6350] and extended with the values declared in [RFC6473] and [RFC6869].

```
KIND:individual
```

```
"kind": "individual"
```

*Figure 7: KIND Conversion Example*

#### 2.4.3. SOURCE

The SOURCE property (Section 6.1.3 of [RFC6350]) converts to a Directory object (Section 2.6.2 of [RFC9553]) in the Card object's directories property (Figure 8). The Directory object's kind property is set to "entry". The uri property is set to the SOURCE property value.

The PREF and MEDIATYPE parameters convert according to the rules defined in Section 2.3.

```
SOURCE:https://dir.example.com/addrbook/jdoe/Jean%20Dupont.vcf
```

```
"directories": {
   "ENTRY-1": {
     "kind": "entry",
     "uri": "https://dir.example.com/addrbook/jdoe/Jean%20Dupont.vcf"
   }
}
```

*Figure 8: SOURCE Conversion Example*

#### 2.4.4. XML

The XML property (Section 6.1.5 of [RFC6350]) converts to the vCardProps property; see Section 2.15.1.

### 2.5. Identification Properties

#### 2.5.1. ANNIVERSARY, BDAY, BIRTHPLACE, DEATHDATE, and DEATHPLACE

The following properties all convert to Anniversary objects in the Card object's anniversaries property (Figure 9):

- ANNIVERSARY (Section 6.2.6 of [RFC6350])
- BDAY (Section 6.2.5 of [RFC6350])
- BIRTHPLACE (Section 2.1 of [RFC6474])
- DEATHDATE (Section 2.3 of [RFC6474])
- DEATHPLACE (Section 2.2 of [RFC6474])

BDAY and BIRTHPLACE convert to an Anniversary object (Section 2.8.1 of [RFC9553]) having the date and place properties set. The kind property is set to "birth".

DEATHDATE and DEATHPLACE convert to an Anniversary object having the date and place properties set. The Anniversary object's kind property is set to "death".

ANNIVERSARY converts to the Anniversary object's date property. The Anniversary object's kind property is set to "wedding".

If the BIRTHPLACE or DEATHPLACE property value is of type URI using the "geo:" URI scheme, then it converts to the Address object's coordinates property. If the value type is TEXT, then it converts to the Address object's full property. Otherwise, it converts to the vCardProps property; see Section 2.15.1.

The ALTID and LANGUAGE parameters of both the BIRTHPLACE and DEATHPLACE properties convert according to the rules defined in Section 2.3.

```
BDAY:19531015T231000Z
BIRTHPLACE:
 123 Main Street\nAny Town, CA 91921-1234\nU.S.A.
DEATHDATE:19960415
DEATHPLACE:
 5 Court Street\nNew England, ND 58647\nU.S.A.
ANNIVERSARY:19860201
```

```
"anniversaries": {
  "ANNIVERSARY-1" : {
    "kind": "birth",
    "date": {
      "@type": "Timestamp",
      "utc": "1953-10-15T23:10:00Z"
    },
    "place": {
      "full":
         "123 Main Street\nAny Town, CA 91921-1234\nU.S.A."
    }
  },
  "ANNIVERSARY-2" : {
    "kind": "death",
    "date": {
      "year": 1996,
      "month": 4,
      "year": 15
    },
    "place": {
      "full": "5 Court Street\nNew England, ND 58647\nU.S.A."
    }
  },
  "ANNIVERSARY-3" : {
    "kind": "wedding",
    "date": {
      "year": 1986,
      "month": 2,
      "day": 1
    }
  }
}
```

*Figure 9: ANNIVERSARY, BDAY, BIRTHPLACE, DEATHDATE, and DEATHPLACE Conversion Example*

#### 2.5.2. FN

The FN property (Section 6.2.1 of [RFC6350]) converts to the Name object's full property (Figure 10). If the LANGUAGE parameter is set, then the FN property converts as outlined in Section 2.3.11. In the unexpected case where the vCard contains more than one FN property without the LANGUAGE parameter, convert the FN property that has the least parameters. If multiple such FN properties are present, choose any of them. All other FN properties convert to the vCardProps (Section 2.15.1) property.

```
FN:John Q. Public, Esq.
```

```
"name": {
  "full": "John Q. Public, Esq."
}
```

*Figure 10: FN Conversion Example*

#### 2.5.3. GENDER

The GENDER property (Section 6.2.7 of [RFC6350]) does not convert to an IANA-registered property in JSContact. To convert this property, see Section 2.15.1. Alternatively, the Card object's speakToAs property defines how to address and refer to an individual represented by the Card, as do the newly defined vCard GRAMGENDER and PRONOUNS properties of [RFC9554].

#### 2.5.4. GRAMGENDER and PRONOUNS

The GRAMGENDER property (Section 3.2 of [RFC9554]) converts to the SpeakToAs object's grammaticalGender property (Figure 11).

The PRONOUNS property (Section 3.4 of [RFC9554]) converts to the SpeakToAs object's pronouns property (Figure 11).

```
GRAMGENDER:NEUTER
PRONOUNS;PREF=2:they/them
PRONOUNS;PREF=1:xe/xir
```

```
"speakToAs": {
  "grammaticalGender": "neuter",
  "pronouns": {
    "PRONOUNS-1": {
      "pronouns": "they/them",
      "pref": 2
    },
    "PRONOUNS-2": {
      "pronouns": "xe/xir",
      "pref": 1
    }
  }
}
```

*Figure 11: GRAMGENDER and PRONOUNS Conversion Example*

#### 2.5.5. N

The N property (Section 6.2.2 of [RFC6350]) converts to a Name object (Section 2.2.1 of [RFC9553]) in the Card object's name property. Each component in the N property structured value converts to a NameComponent in the Name object's components property. The following table shows this relation:

| N component | NameComponent kind | Remarks |
|---|---|---|
| Family name | surname | To vCard: add any "surname2" NameComponent to the Family name component, after all "surname" values.<br>From vCard: ignore any value that also occurs in the Secondary surname component. |
| Given name | given | |
| Additional name | given2 | |
| Honorific prefix | title | |
| Honorific suffix | credential | To vCard: add any "generation" NameComponent to the Honorific suffix component.<br>From vCard: ignore any value that also occurs in the Generation component. |
| Secondary surname | surname2 | |
| Generation | generation | |

*Table 1: N Components Conversion*

If the JSCOMPS (Section 3.3.1) parameter is set, then the Name object's isOrdered property value is "true", and the defaultSeparator property and any "separator" NameComponent objects are set according to the parameter value. The order in the components property **MUST** adhere to the order of the JSCOMPS parameter value.

If the JSCOMPS parameter is not set, then the Name object's isOrdered property value is "false", and the defaultSeparator property **MUST NOT** be set. The order in the components property **MUST** follow the order of values in the N structured value when read from left to right.

If the SORT-AS parameter is set, then its structured value converts to the Name object's sortAs property according to Table 1. An empty or non-existent component value indicates that no sort is defined for this kind.

```
N;SORT-AS="Stevenson,John Philip":
 Stevenson;John;Philip,Paul;Dr.;Jr.,M.D.,A.C.P.;;Jr.
```

```
"name": {
  "components":[
    { "kind": "surname", "value": "Stevenson" },
    { "kind": "given", "value": "John" },
    { "kind": "given2", "value": "Philip" },
    { "kind": "given2", "value": "Paul" },
    { "kind": "title", "value": "Dr." },
    { "kind": "credential", "value": "M.D." },
    { "kind": "credential", "value": "A.C.P." },
    { "kind": "generation", "value": "Jr." }
  ],
  "sortAs": {
    "surname": "Stevenson",
    "given": "John Philip"
  }
}
```

*Figure 12: N Conversion Example*

See Section 3.3.1 for examples of using the JSCOMPS parameter for vCard-structured property values.

#### 2.5.6. NICKNAME

The NICKNAME property (Section 6.2.3 of [RFC6350]) converts to a Nickname object (Section 2.2.2 of [RFC9553]) in the Card object's nicknames property (Figure 13). The name property is set to the NICKNAME property value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
NICKNAME:Johnny
```

```
"nicknames": {
   "NICK-1": {
    "name": "Johnny"
   }
}
```

*Figure 13: NICKNAME Conversion Example*

#### 2.5.7. PHOTO

The PHOTO property (Section 6.2.4 of [RFC6350]) converts to a Media object (Section 2.6.4 of [RFC9553]) in the Card object's media property (Figure 14). The Media object's kind property is set to "photo" and the uri property is set to the PHOTO value.

The PREF and MEDIATYPE parameters convert according to the rules defined in Section 2.3.

```
PHOTO:https://www.example.com/pub/photos/jqpublic.gif
```

```
"media": {
   "PHOTO-1": {
     "kind": "photo",
     "uri": "https://www.example.com/pub/photos/jqpublic.gif"
   }
}
```

*Figure 14: PHOTO Conversion Example*

## 2.6. Delivery Addressing Properties

### 2.6.1. ADR

The ADR property (Section 6.3.1 of [RFC6350]) converts to an Address object (Section 2.5.1.1 of [RFC9553]) in the Card object's addresses property. Each component in the ADR-structured property value converts to an AddressComponent in the Address object's components property.

[RFC9554] defines new components for the ADR property. Implementations **SHOULD** set these new components, even if all their values are the empty string.

The following table shows how the ADR component and AddressComponent kind relate:

| ADR component | AddressComponent kind | Remarks |
|---|---|---|
| post office box | postOfficeBox | [RFC6350] recommends that this component not be set, but this is now disputable given the new components. Instead, set this component and use the new ADR value format defined in [RFC9554]. |
| extended address | apartment | To vCard: set the values of the following components:<br>• room<br>• floor<br>• apartment<br>• building<br><br>From vCard: ignore if the ADR structured value is of the format defined in [RFC9554]. Otherwise, convert to "apartment". |

| ADR component | AddressComponent kind | Remarks |
|---|---|---|
| street address | name | To vCard: set the values of the following components:<br>• number<br>• name<br>• block<br>• direction<br>• landmark<br>• subdistrict<br>• district<br><br>From vCard: ignore if the ADR structured value is of the format defined in [RFC9554]. Otherwise, convert to "name". |
| locality | locality | |
| region | region | |
| postal code | postcode | |
| apartment | apartment | Defined in [RFC9554]. |
| block | block | Defined in [RFC9554]. |
| building | building | Defined in [RFC9554]. |
| direction | direction | Defined in [RFC9554]. |
| district | district | Defined in [RFC9554]. |
| floor | floor | Defined in [RFC9554]. |
| landmark | landmark | Defined in [RFC9554]. |
| room | room | Defined in [RFC9554]. |
| street number | number | Defined in [RFC9554]. |
| subdistrict | subdistrict | Defined in [RFC9554]. |

*Table 2: ADR Components Conversion*

If the JSCOMPS (Section 3.3.1) parameter is set, then the Address object's isOrdered property value is "true", and the defaultSeparator property and any separator name components are set according to the parameter value. The order in the components property **MUST** adhere to the order of the JSCOMPS parameter value.

If the JSCOMPS parameter is not set, then the Address object's isOrdered property value is "false", and the defaultSeparator property **MUST NOT** be set. The order in the components property **MUST** follow the order of values in the ADR structured value when read from left to right.

The LABEL parameter converts to the Address object's full property.

The GEO parameter converts to the Address object's coordinates property.

The TZ parameter converts to the Address object's timeZone property.

The CC parameter converts to the Address object's countryCode property.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3. The ADR-specific values of the TYPE parameter defined in Sections 5.1 and 5.2 of [RFC9554] convert to the corresponding entries of the contexts property as defined in Section 2.5.1 of [RFC9553].

The ALTID and LANGUAGE parameters convert according to the rules defined in Section 2.3. Each possible language-dependent alternative converts to an entry of the PatchObject where the key references the full property.

```
ADR;TYPE=work;CC=US:
 ;;54321 Oak St;Reston;VA;20190;USA;;;;54321;Oak St;;;;;;
```

```
"addresses": {
  "ADDR-1" : {
    "contexts": { "work": true },
    "components": [
      { "kind": "number", "value": "54321" },
      { "kind": "name", "value": "Oak St" },
      { "kind": "locality", "value": "Reston" },
      { "kind": "region", "value": "VA" },
      { "kind": "postcode", "value": "20190" },
      { "kind": "country", "value": "USA" }
    ],
    "countryCode": "US"
  }
}
```

*Figure 15: ADR Conversion Example*

See Section 3.3.1 for examples of using the JSCOMPS parameter for vCard-structured property values.

## 2.7. Communications Properties

### 2.7.1. EMAIL

The EMAIL property (Section 6.4.2 of [RFC6350]) converts to an EmailAddress object (Section 2.3.1 of [RFC9553]) in the Card object's emails property (Figure 16). The EmailAddress object's address property is set to the EMAIL value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
EMAIL;TYPE=work:jqpublic@xyz.example.com
EMAIL;PREF=1:jane_doe@example.com
```

```
"emails": {
  "EMAIL-1": {
    "contexts": { "work": true },
    "address": "jqpublic@xyz.example.com"
  },
  "EMAIL-2": {
    "address": "jane_doe@example.com",
    "pref": 1
  }
}
```

*Figure 16: EMAIL Conversion Example*

### 2.7.2. IMPP

The IMPP property (Section 6.4.3 of [RFC6350]) converts to an OnlineService object (Section 2.3.2 of [RFC9553]) in the Card object's onlineServices property (Figure 17). The vCardName property is set to "impp", and the uri property is set to the IMPP value.

The SERVICE-TYPE, USERNAME, PREF, and TYPE parameters convert according to the rules defined in Section 2.3.

```
IMPP;PREF=1:xmpp:alice@example.com
```

```
"onlineServices": {
  "OS-1": {
    "uri": "xmpp:alice@example.com",
    "pref": 1,
    "vCardName": "impp"
  }
}
```

*Figure 17: IMPP Conversion Example*

### 2.7.3. LANG

The LANG property (Section 6.4.4 of [RFC6350]) converts to a LanguagePref object (Section 2.3.4 of [RFC9553]) in the Card object's preferredLanguages property (Figure 18). The LANG property value converts to the LanguagePref object's language property value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
LANG;TYPE=work;PREF=1:en
LANG;TYPE=work;PREF=2:fr
LANG;TYPE=home:fr
```

```
"preferredLanguages": {
  "LANG-1": {
    "language": "en",
    "contexts": { "work": true },
    "pref": 1
  },
  "LANG-2": {
    "language": "fr",
    "contexts": { "work": true },
    "pref": 2
  },
  "LANG-3": {
    "language": "fr",
    "contexts": { "private": true }
  }
}
```

*Figure 18: LANG Conversion Example*

### 2.7.4. LANGUAGE

The LANGUAGE property (Section 3.3 of [RFC9554]) converts to the Card object's language property (Figure 19).

```
LANGUAGE:de-AT
```

```
"language": "de-AT"
```

*Figure 19: LANGUAGE Conversion Example*

### 2.7.5. SOCIALPROFILE

The SOCIALPROFILE property (Section 3.5 of [RFC9554]) converts to an OnlineService object (Section 2.3.2 of [RFC9553]) in the Card object's onlineServices property (Figure 20). The vCardName property is set to "socialprofile", or it can be omitted. If the SOCIALPROFILE property value is of type URI, then the OnlineService object's uri property is set; otherwise, the user property is set.

The SERVICE-TYPE, USERNAME, PREF, and TYPE parameters convert according to the rules defined in Section 2.3.

```
SOCIALPROFILE;SERVICE-TYPE=Mastodon:https://example.com/@foo
```

```
"onlineServices": {
  ...
  "OS-1": {
    "service": "Mastodon",
    "uri": "https://example.com/@foo"
  }
}
```

*Figure 20: SOCIALPROFILE Conversion Example*

### 2.7.6. TEL

The TEL property (Section 6.4.1 of [RFC6350]) converts to a Phone object (Section 2.3.3 of [RFC9553]) in the Card object's phones property (Figure 21).

The TEL-specific values of the TYPE parameter convert to the features property keys as outlined in Table 3. Note that Section 6.4.1 of [RFC6350] defines the default type to be "voice", but the default Phone features property is absent by default. Accordingly, an implementation **SHOULD** only set the Phone object's features property if the TEL property actually has a TEL-specific TYPE parameter set.

| TYPE value | Phone feature |
|---|---|
| cell | mobile |
| fax | fax |
| main-number | main-number |
| pager | pager |
| text | text |
| textphone | textphone |

| TYPE value | Phone feature |
|---|---|
| video | video |
| voice | voice |

*Table 3: TEL TYPE Conversion*

The value of the TEL property converts to the Phone object's number property.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
TEL;VALUE=uri;PREF=1;TYPE="voice,home":tel:+1-555-555-5555;ext=5555
TEL;VALUE=uri;TYPE=home:tel:+33-01-23-45-67
```

```
"phones": {
  "PHONE-1": {
    "contexts": { "private": true },
    "features": { "voice": true },
    "number": "tel:+1-555-555-5555;ext=5555",
    "pref": 1
  },
  "PHONE-2": {
    "contexts": { "private": true },
    "number": "tel:+33-01-23-45-67"
  }
}
```

*Figure 21: TEL Conversion Example*

## 2.8. Geographical Properties

### 2.8.1. GEO

The GEO property (Section 6.5.2 of [RFC6350]) converts to the Address object's coordinates property (Section 2.5.1 of [RFC9553]). Also see Section 2.8.3 to determine which Address object instance to convert to.

### 2.8.2. TZ

The TZ property (Section 6.5.1 of [RFC6350]) converts an Address object (Section 2.5.1 of [RFC9553]) in the Card object's addresses property.

A value of type TEXT converts to the Address object's timeZone property.

A value of type UTC-OFFSET converts to the Address object's timeZone property if the offset has zero minutes and the hour offset is between -12 and +14, both inclusively. Note that:

- If the hour offset is zero, use the time zone name "Etc/UTC".

- Otherwise, construct the time zone name with "Etc/GMT" suffixed with the string representation of the reversed sign hour offset, including the sign but excluding leading zeros and minutes. For example, the UTC offset value "-0500" converts to "Etc/GMT+5".

For such property values, also see Section 2.8.3 to determine which Address object instance to convert to.

Any other value of type UTC-OFFSET or URI does not convert to an IANA-registered property in JSContact. To convert such property, see Section 2.15.1.

#### 2.8.3. Combining Geographical Properties

In vCard, the properties ADR, GEO, and TZ occur independently of each other. In JSContact, they all convert to properties of an Address object. It is implementation-specific if these vCard properties convert to *separate* address instances in JSContact or if some or all of them convert to the *same* address. That being said, implementations **MUST** convert the properties to the *same* address for the following cases:

- The GROUP parameter values of the properties match.
- The GROUP parameters are not set, but they are set on any other ADR, GEO, and TZ properties.

### 2.9. Organizational Properties

#### 2.9.1. CONTACT-URI

The CONTACT-URI property (Section 2.1 of [RFC8605]) converts to a Link object (Section 2.6.3 of [RFC9553]) in the Card object's links property (Figure 22). The Link object's kind property is set to "contact" and the uri property is set to the CONTACT-URI property value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
CONTACT-URI;PREF=1:mailto:contact@example.com
```

```
"links": {
  "CONTACT-1": {
    "kind": "contact",
    "uri": "mailto:contact@example.com",
    "pref": 1
  }
}
```

*Figure 22: CONTACT-URI Conversion Example*

### 2.9.2. LOGO

The LOGO property (Section 6.6.3 of [RFC6350]) converts to a Media object (Section 2.6.4 of [RFC9553]) in the Card object's media property (Figure 23). The Media object's kind property is set to "logo" and the uri property is set to the LOGO property value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
LOGO:https://www.example.com/pub/logos/abccorp.jpg
```

```
"media": {
  "LOGO-1": {
    "kind": "logo",
    "uri": "https://www.example.com/pub/logos/abccorp.jpg"
  }
}
```

*Figure 23: LOGO Conversion Example*

### 2.9.3. MEMBER

The MEMBER property (Section 6.6.5 of [RFC6350]) converts to the Card object's members property (Figure 24). Each MEMBER property value is a key in the members property. The PREF parameter (Section 5.3 of [RFC6350]) does not convert to JSContact.

```
KIND:group
FN:The Doe family
MEMBER:urn:uuid:03a0e51f-d1aa-4385-8a53-e29025acd8af
MEMBER:urn:uuid:b8767877-b4a1-4c70-9acc-505d3819e519
```

```
"kind": "group",
"name": {
  "full": "The Doe family"
},
"uid": "urn:uuid:ab4310aa-fa43-11e9-8f0b-362b9e155667",
"members": {
  "urn:uuid:03a0e51f-d1aa-4385-8a53-e29025acd8af": true,
  "urn:uuid:b8767877-b4a1-4c70-9acc-505d3819e519": true
}
```

*Figure 24: Group Example*

#### 2.9.4. ORG

The ORG property (Section 6.6.4 of [RFC6350]) converts to an Organization object (Section 2.2.3 of [RFC9553]) in the Card object's organizations property (Figure 25). The Organization object's name property is set to the ORG property organizational name component. The Organization object's units property is an array of OrgUnit objects that each contain an organizational unit name component value of the ORG property value.

Implementations **MAY** allow representation of organizational units without the organizational name. In this case, the first component of the ORG value **MUST** be an empty string (e.g., ORG:;DepartmentA).

The ALTID and LANGUAGE parameters convert according to the rules defined in Section 2.3.

The first item of the comma-separated SORT-AS parameter value converts to the sortAs property of the Organization object. The subsequent items convert to the sortAs property of the corresponding OrgUnit object.

The TYPE parameter converts according to the rules defined in Section 2.3.

```
ORG;SORT-AS="ABC":ABC\, Inc.;North American Division;Marketing
```

```
"organizations": {
  "ORG-1": {
    "name": "ABC, Inc.",
    "units":[
      { "name": "North American Division" },
      { "name": "Marketing" }
    ],
    "sortAs": "ABC"
  }
}
```

*Figure 25: ORG Conversion Example*

#### 2.9.5. RELATED

The RELATED property (Section 6.6.6 of [RFC6350]) converts to the Card object's relatedTo property (Figure 26). The property value converts to the key in the relatedTo property. The TYPE parameters convert to the Relation object's relation property (Section 2.1.8 of [RFC9553]). Any other parameters convert as defined in Section 2.15.2.

```
RELATED;TYPE=friend:urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
RELATED;TYPE=contact:https://example.com/directory/john.vcf
RELATED;VALUE=text:Please contact my deputy John for any inquiries.
```

```
"relatedTo" : {
  "urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6" : {
    "relation" : {
      "friend" : true
    }
  },
  "https://example.com/directory/john.vcf" : {
    "relation" : {
      "contact" : true
    }
  },
  "Please contact my deputy John for any inquiries." : {
    "relation" : { }
  }
}
```

*Figure 26: RELATED Conversion Example*

#### 2.9.6. TITLE and ROLE

The TITLE (Section 6.6.1 of [RFC6350]) and ROLE (Section 6.6.2 of [RFC6350]) properties convert to a Title object (Section 2.2.5 of [RFC9553]) in the Card object's titles property (Figure 27). The Title object's kind property is set to "title" or "role" for the TITLE and ROLE vCard properties, respectively. The name property is set to the vCard property value.

The value of the organizationId property can be derived if the TITLE or ROLE property is a member of a vCard property group and if exactly one other ORG property is also a part of that group.

The ALTID and LANGUAGE parameters convert according to the rules defined in Section 2.3.

```
TITLE:Research Scientist
group1.ROLE:Project Leader
group1.ORG:ABC, Inc.
```

```
"titles": {
  "TITLE-1": {
    "kind": "title",
    "name": "Research Scientist"
  },
  "TITLE-2": {
    "kind": "role",
    "name": "Project Leader",
    "organizationId": "ORG-1"
  }
},
"organizations": {
  "ORG-1": {
    "name": "ABC, Inc."
  }
}
```

*Figure 27: TITLE and ROLE Conversion Example*

## 2.10. Personal Information Properties

### 2.10.1. EXPERTISE

The EXPERTISE property (Section 2.1 of [RFC6715]) converts to a PersonalInfo object (Section 2.8.4 of [RFC9553]) in the Card object's personalInfo property (Figure 28). The PersonalInfo object's kind property is set to "expertise".

The INDEX and LEVEL parameters convert according to the rules defined in Section 2.3.

```
EXPERTISE;LEVEL=beginner;INDEX=2:Chinese literature
EXPERTISE;INDEX=1;LEVEL=expert:chemistry
```

```
"personalInfo": {
  "PERSINFO-1" : {
    "kind": "expertise",
    "value": "Chinese literature",
     "level": "low",
     "listAs": 2
  },
  "PERSINFO-2" : {
    "kind": "expertise",
    "value": "chemistry",
    "level": "high",
    "listAs": 1
  }
}
```

*Figure 28: EXPERTISE Conversion Example*

#### 2.10.2. HOBBY

The HOBBY property (Section 2.2 of [RFC6715]) converts to a PersonalInfo object (Section 2.8.4 of [RFC9553]) in the Card object's personalInfo property (Figure 29). The PersonalInfo object's kind property is set to "hobby".

The INDEX and LEVEL parameters convert according to the rules defined in Section 2.3.

```
HOBBY;INDEX=1;LEVEL=high:reading
HOBBY;INDEX=2;LEVEL=high:sewing
```

```
"personalInfo": {
  "PERSINFO-1" : {
    "kind": "hobby",
    "value": "reading",
    "level": "high",
    "listAs": 1
  },
  "PERSINFO-2" : {
    "kind": "hobby",
    "value": "sewing",
    "level": "high",
    "listAs": 2
  }
}
```

*Figure 29: HOBBY Conversion Example*

### 2.10.3. INTEREST

The INTEREST property (Section 2.3 of [RFC6715]) converts to a PersonalInfo object (Section 2.8.4 of [RFC9553]) in the Card object's personalInfo property (Figure 30). The PersonalInfo object's kind property is set to "interest".

The INDEX and LEVEL parameters convert according to the rules defined in Section 2.3.

```
INTEREST;INDEX=1;LEVEL=medium:r&b music
INTEREST;INDEX=2;LEVEL=high:rock&roll music
```

```
"personalInfo": {
  "PERSINFO-1" : {
    "kind": "interest",
    "value": "r&b music",
    "level": "medium",
    "listAs": 1
  },
  "PERSINFO-2" : {
    "kind": "interest",
    "value": "rock&roll music",
    "level": "high",
    "listAs": 2
  }
}
```

*Figure 30: INTEREST Conversion Example*

### 2.10.4. ORG-DIRECTORY

The ORG-DIRECTORY property (Section 2.4 of [RFC6715]) [RFC6715] converts to a Directory object (Section 2.6.2 of [RFC9553]) in the Card object's directories property (Figure 31). The Directory object's kind property is set to "directory". The uri property is set to the ORG-DIRECTORY property value.

The INDEX, PREF, and TYPE parameters convert according to the rules defined in Section 2.3.

```
ORG-DIRECTORY;INDEX=1:https://directory.mycompany.example.com
ORG-DIRECTORY;PREF=1:ldap://ldap.tech.example/o=Tech,ou=Engineering
```

```
"directories": {
  "DIRECTORY-1": {
    "kind": "directory",
    "uri": "https://directory.mycompany.example.com",
    "listAs": 1
  },
  "DIRECTORY-2": {
    "kind": "directory",
    "uri": "ldap://ldap.tech.example/o=Tech,ou=Engineering",
    "pref": 1
  }
}
```

*Figure 31: ORG-DIRECTORY Conversion Example*

## 2.11. Explanatory Properties

### 2.11.1. CATEGORIES

The CATEGORIES property (Section 6.7.1 of [RFC6350]) converts to a set of entries of the Card object's keywords property (Figure 32). The keys are the comma-separated text values of the CATEGORIES property.

In this case, the PREF parameter does not have a JSContact counterpart; however, the implementors **MAY** insert the entries by order of preference.

```
CATEGORIES:internet,IETF,Industry,Information Technology
```

```
"keywords": {
  "internet": true,
  "IETF": true,
  "Industry": true,
  "Information Technology": true
}
```

*Figure 32: CATEGORIES Conversion Example*

### 2.11.2. CLIENTPIDMAP

The CLIENTPIDMAP property (Section 6.7.7 of [RFC6350]) converts to the vCardProps (Section 2.15.1) property.

#### 2.11.3. CREATED

The CREATED property (Section 3.1 of [RFC9554]) converts to the Card object's created property (Figure 33).

```
CREATED:19940930T143510Z
```

```
"created": "1994-09-30T14:35:10Z"
```

*Figure 33: CREATED Conversion Example*

#### 2.11.4. NOTE

The NOTE property (Section 6.7.2 of [RFC6350]) converts to a Note object (Section 2.8.3 of [RFC9553]) in the Card object's notes property (Figure 34).

The ALTID and LANGUAGE parameters convert according to the rules defined in Section 2.3.

```
NOTE;CREATED=20221123T150132Z;AUTHOR-NAME="John":
  Office hours are from 0800 to 1715 EST\, Mon-Fri.
```

```
"notes": {
  "NOTE-1" : {
     "note": "Office hours are from 0800 to 1715 EST, Mon-Fri.",
     "created": "2022-11-23T15:01:32Z",
     "author": {
       "name": "John"
     }
  }
}
```

*Figure 34: NOTE Conversion Example*

#### 2.11.5. PRODID

The PRODID property (Section 6.7.3 of [RFC6350]) converts to the Card object's prodId property (Figure 35).

```
PRODID:ACME Contacts App version 1.23.5
```

```
"prodId": "ACME Contacts App version 1.23.5"
```

*Figure 35: PRODID Conversion Example*

#### 2.11.6. REV

The REV property (Section 6.7.4 of [RFC6350]) converts to the Card object's updated property (Figure 36).

```
REV:19951031T222710Z
```

```
"updated": "1995-10-31T22:27:10Z"
```

*Figure 36: REV Conversion Example*

#### 2.11.7. SOUND

The SOUND property (Section 6.7.5 of [RFC6350]) converts to a Media object (Section 2.6.4 of [RFC9553]) in the Card object's media property (Figure 37). The Media object's kind property is set to "sound" and the uri property is set to the SOUND value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
SOUND:CID:JOHNQPUBLIC.19960229T080000.xyzMail@example.com
```

```
"media": {
  ...
  "SOUND-1": {
    "kind": "sound",
    "uri": "CID:JOHNQPUBLIC.19960229T080000.xyzMail@example.com"
  }
}
```

*Figure 37: SOUND Conversion Example*

#### 2.11.8. UID

The UID property (Section 6.7.6 of [RFC6350]) converts to the Card object's uid property (Figure 38).

```
UID:urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

```
"uid": "urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
```

*Figure 38: UID Conversion Example*

#### 2.11.9. URL

The URL property (Section 6.7.8 of [RFC6350]) converts to a Link object (Section 2.6.3 of [RFC9553]) in the Card object's links property (Figure 39). The Link object's uri property is set to the URL value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
URL:https://example.org/restaurant.french/~chezchic.html
```

```
"links": {
  "LINK-1": {
    "uri": "https://example.org/restaurant.french/~chezchic.html"
  }
}
```

*Figure 39: URL Conversion Example*

#### 2.11.10. VERSION

The VERSION property (Section 6.7.9 of [RFC6350]) converts to the vCardProps (Section 2.15.1) property.

#### 2.11.11. X-ABLabel

The X-ABLabel property is experimental but widely in use in existing vCard data. It converts to the label property of a JSContact object. The X-ABLabel property is preceded by a vCard property group name, and the label converts to the JSContact object, which was converted from a vCard property of the same group.

The group name is not preserved; implementations are free to choose any unique group name when converting back to vCard. For an example on how to preserve the group name, see Section 2.3.9.

```
item1.TEL;VALUE=uri:tel:+1-555-555-5555
item1.X-ABLabel:foo
```

```
"phones": {
  "p1": {
    "number": "tel:+1-555-555-5555",
    "label": "foo"
  }
}
```

*Figure 40: X-ABLabel Conversion Example*

## 2.12. Security Properties

### 2.12.1. KEY

The KEY property (Section 6.8.1 of [RFC6350]) converts to a CryptoKey object (Section 2.6.1 of [RFC9553]) in the Card object's cryptoKeys property (Figure 41). The CryptoKey object's uri property is set to the KEY property value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
KEY:https://www.example.com/keys/jdoe.cer
```

```
"cryptoKeys": {
  "KEY-1": {
    "uri": "https://www.example.com/keys/jdoe.cer"
  }
}
```

*Figure 41: KEY Conversion Example*

## 2.13. Calendar Properties

### 2.13.1. CALADRURI

The CALADRURI property (Section 6.9.2 of [RFC6350]) converts to a SchedulingAddress object (Section 2.4.2 of [RFC9553]) in the Card object's schedulingAddresses property (Figure 42). The SchedulingAddress object's uri property is set to the CALADRURI value.

The PREF parameter (Section 5.3 of [RFC6350]) converts according to the rules defined in Section 2.3.

```
CALADRURI;PREF=1:mailto:janedoe@example.com
CALADRURI:https://example.com/calendar/jdoe
```

```
"schedulingAddresses": {
  "SCHEDULING-1": {
    "uri": "mailto:janedoe@example.com",
    "pref": 1
  },
  "SCHEDULING-2": {
    "uri": "https://example.com/calendar/jdoe"
  }
}
```

*Figure 42: CALADRURI Conversion Example*

### 2.13.2. CALURI

The CALURI property (Section 6.9.3 of [RFC6350]) converts to a Calendar object (Section 2.4.1 of [RFC9553]) in the Card object's calendars property (Figure 43). The Calendar object's kind property is set to "calendar" and the uri property is set to the CALURI value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
CALURI;PREF=1:https://cal.example.com/calA
CALURI;MEDIATYPE=text/calendar:https://ftp.example.com/calA.ics
```

```
"calendars": {
  "CAL-1": {
    "kind": "calendar",
    "uri": "https://cal.example.com/calA",
    "pref": 1
  },
  "CAL-2": {
    "kind": "calendar",
    "uri": "https://ftp.example.com/calA.ics",
    "mediaType": "text/calendar"
  }
}
```

*Figure 43: CALURI Conversion Example*

### 2.13.3. FBURL

The FBURL property (Section 6.9.1 of [RFC6350]) converts to a Calendar object (Section 2.4.1 of [RFC9553]) in the Card object's calendars property (Figure 44). The Calendar object's kind property is set to "freeBusy" and the uri property is set to the FBURL value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.3.

```
FBURL;PREF=1:https://www.example.com/busy/janedoe
FBURL;MEDIATYPE=text/calendar:https://example.com/busy/project-a.ifb
```

```
"calendars": {
  "FBURL-1": {
    "kind": "freeBusy",
    "uri": "https://www.example.com/busy/janedoe",
    "pref": 1
  },
  "FBURL-2": {
    "kind": "freeBusy",
    "uri": "https://example.com/busy/project-a.ifb",
    "mediaType": "text/calendar"
  }
}
```

*Figure 44: FBURL Conversion Example*

## 2.14. Extended Properties and Parameters

Extended properties and parameters convert as specified in Section 2.15.

## 2.15. New JSContact Properties

vCards may contain properties or parameters for which no IANA-registered JSContact property is defined. For example, a vCard may contain properties and parameters of which the semantics or purposes are unknown to the implementation; see Section 6.10 of [RFC6350].

This section defines JSContact properties by which such vCard properties and parameters **MAY** be represented in JSContact. Implementations **MAY** choose to convert differently if they deem that more appropriate.

### 2.15.1. vCardProps

vCardProps: JCardProp[] (optional). Contains vCard properties that are set in the vCard represented by this JSContact object. The JCardProp type denotes a jCard-encoded vCard property as defined in Section 3.3 of [RFC7095].

Example: This illustrates how to convert a vCard extension property:

```
item1.X-FOO;X-BAR=Hello:World!
```

```
"vCardProps": [
  ["x-foo", {
    "x-bar": "Hello",
    "group": "item1"
  }, "unknown", "World!"]
]
```

*Figure 45: JSContact vCardProps Example*

### 2.15.2. vCardParams

vCardParams: String[String|String[]] (optional). Contains vCard parameters that are set on the vCard property represented by this JSContact object. The value **MUST** be a JSON object containing vCard property parameters as defined in Section 3.3 of [RFC7095]. Each entry represents a parameter of the vCard property that converts to the JSContact object.

Example: This illustrates how to convert a vCard extension parameter:

```
EMAIL;X-FOO=Bar:jane_doe@example.com
```

```
"emails": {
  "email1": {
    "address": "jane_doe@example.com",
    "vCardParams": {
      "x-foo": "Bar"
    }
  }
}
```

*Figure 46: JSContact vCardParams Example*

### 2.15.3. vCardName

vCardName: String (optional). Contains the name of the vCard element that is represented by this JSContact object. For example, this allows to preserve the name of a vCard property when multiple vCard properties convert the same JSContact type. The case-insensitive value **MUST** be valid according to the "name" ABNF defined in Section 3.3 of [RFC6350].

Example: Both vCard IMPP and SOCIALPROFILE convert to an OnlineService object (Section 2.3.2 of [RFC9553]) in JSContact. The vCardName property value indicates that the vCard source element was IMPP as follows:

```
IMPP:xmpp:alice@example.com
```

```
"onlineServices": {
  "os1": {
    "uri": "xmpp:alice@example.com",
    "vCardName": "impp"
  },
}
```

*Figure 47: JSContact vCardName Example*

# 3. Converting JSContact to vCard

## 3.1. Conversion Rules

A Card object converts to vCard by applying the reverse rules of converting vCard to JSContact. In addition to those listed in Appendix A, the following rules apply:

- Multivalued JSContact properties convert to separate instances of their equivalent vCard property, and each of the PROP-ID parameters **MUST** be set to the Id-typed key of the converted value (see Section 2.3.18).
- The full property of the name property in JSContact is optional, but the FN property is mandatory in vCard. The following rules apply:
  - If the Name object's full property is set, then implementations **MUST** use its value for the vCard FN property.
  - If the Name object's full property is not set, then implementations **SHOULD** derive the full name from the Name object's components property values. If the isOrdered property is "true", then this can be done by concatenating the name component values. Otherwise, or alternatively, an implementation can choose any other heuristic to generate the full name from its components such as [CLDRPersonName]. Implementations **MUST** set the DERIVED parameter on the FN property.
  - Otherwise, the FN property **MUST** be set to the empty value.
- Vendor-specific and unknown properties convert to vCard as outlined in Section 3.1.1.

### 3.1.1. Converting Unknown Properties

JSContact objects may contain properties for which no IANA-registered vCard property is defined. For example, a JSContact object may contain vendor-specific properties of which the semantics or purpose are unknown.

This specification defines the new JSPROP (Section 3.2.1) vCard property and JSPTR (Section 3.3.2) vCard parameter by which such JSContact properties **MAY** be represented in vCard. Implementations **MAY** choose to convert differently if they deem that more appropriate.

## 3.2. New vCard Properties

### 3.2.1. JSPROP

Property name: JSPROP

Purpose: Represents a JSContact property in vCard.

Value type: TEXT; also see "Format definition" below for value restrictions.

Conformance: Can be specified multiple times in a vCard.

Property parameters: The JSPTR parameter **MUST** be set for this property. Other IANA-registered and experimental property parameters can be specified on this property.

Description: This property converts an arbitrary JSContact property from and to vCard. The vCard property value is the JSON-encoded value of the JSContact property, represented as a TEXT value. The format of the JSON value **MUST** be compact, e.g., without insignificant whitespace as defined in Section 2 of [RFC8259]. The value of the JSPTR parameter points to the JSContact property within the Card.

The root of the JSON pointer is always the Card object that this vCard converts to, irrespective if the JSON pointer starts with the SOLIDUS (U+002F) character. The pointer **MUST NOT** reference into an array.

All JSPROP properties in a vCard together form a PatchObject as defined in [RFC9553]. The value of its JSPTR parameter corresponds to a key in the PatchObject; the value of the JSPROP property corresponds to the value for that key. When converting from vCard to JSContact, the PatchObject **MUST** only be applied after all other vCard properties have already been converted. The PatchObject **MUST** be valid, including the restriction that an invalid PatchObject **MUST NOT** be applied.

Format definition: This property is defined by the following notation:

```
jsprop = "JSPROP" jsprop-param ":" TEXT

jsprop-param  = *(
            ; The following are REQUIRED and MUST NOT
            ; occur more than once
            ( ";" jsptr-param ) /    ; see next section
            ( ";" "VALUE" "=" "TEXT")
            ;
            ; The following is OPTIONAL
            ; and MAY occur more than once.
            ;
            (";" other-param)
            ;
            )
```

Example(s): This illustrates how to convert a property at the top level in a Card object that is unknown to the implementation.

```
"someUnknownProperty": true
```

```
JSPROP;JSPTR="someUnknownProperty":true
```

*Figure 48: Unknown Property Example*

This illustrates how to convert a vendor-specific property at the top level of a Card object. Note the required use of quoted string for the JSPTR value, which allows the path to include the COLON (U+003A) character.

```
"example.com:foo": {
  "bar": 1234
}
```

```
JSPROP;JSPTR="example.com:foo":{"bar":1234}
```

*Figure 49: Vendor-Specific Property Conversion Example*

This illustrates how to convert a vendor-specific property at a nested level in a Card object using a path relative to the Card object. Although not recommended, the property name includes the SOLIDUS (U+002F) character, which requires escaping in the JSON pointer.

```
"phones": {
  "phone1": {
    "number": "tel:+33-01-23-45-67",
    "example.com:foo/bar": "tux hux"
  }
}
```

```
TEL:tel:+33-01-23-45-67
JSPROP;JSPTR="phones/phone1/example.com:foo~1bar":
 "tux hux"
```

*Figure 50: Nested Vendor-Specific Property Example with a Path Relative to Card*

## 3.3. New vCard Parameters

### 3.3.1. JSCOMPS

Parameter name: JSCOMPS

Purpose: Defines the order and separators for the elements of a structured property value.

Description: The JSCOMPS parameter value facilitates converting name and address components between JSContact and vCard. It preserves the order of the components of the JSContact property and contains the verbatim values of separator components.

If this parameter is set and its value is valid (see later), then implementations **MUST** set the isOrdered property of the Name or Address object to "true". Otherwise, they **MUST** set the isOrdered property value to "false".

The JSCOMPS parameter value is a structured type value. Its value **MUST** be quoted. The parameter value consists of a sequence of entries, separated by the SEMICOLON character (U+003B). The first entry defines the value of the defaultSeparator property. If it is the empty string, then no default separator is defined. Otherwise, the first entry **MUST** be a separator entry. All following entries processed in order result in an ordered list of JSContact components and **MUST** be one of the following two kinds:

1. A positional. This refers to a component value in the vCard structured value. A position consists of the numeric index of a component in the structured value, optionally followed by a COMMA (U+002C) character and the non-zero index of a value within that component. The zero index selects the first component or value, respectively. The second index is zero by default, in which case it **MUST** be omitted (as well as the leading COMMA).

   The resulting JSContact component is formed by determining its kind by the position in the vCard structured value. The component value is the verbatim value of the vCard component. Figures 51 and 52 illustrate this by example.

2. A separator. This contains the verbatim value of a separator component. It starts with the LATIN SMALL LETTER S (U+0073) character, followed by the COMMA (U+002C) character, followed by zero or more "param-value" characters (see Section 3.3 of [RFC6350]), where the COMMA (U+002C) and SEMICOLON (U+003B) characters **MUST** be escaped according to the rules defined in Section 3.4 of [RFC6350]. Figure 53 illustrates this by example.

   The resulting JSContact component is formed by setting its kind to "separator" and its value to the verbatim value of the entry.

A JSCOMPS parameter value is valid if and only if:

- All indexes in the positional entries refer to an existing component value in the vCard property value.
- The count of positional entries equals the count of deduplicated component values. Deduplication is required because some values may occur in both their designated and backwards-compatible components in the vCard property value:
  - A value that occurs in both the N property secondary surname component and the family name component only counts once.
  - A value that occurs in both the N property generation component and the honorific suffix component only counts once.

- A value in the ADR property street address component does not count if the ADR property value contains a value in one of the new components defined in [RFC9554].
- All other values count once each.

Format definition:

```
jscomps-param      = "JSCOMPS" "=" DQUOTE [jscomps-entry-sep ] ";"
                                   jscomps-entrylist DQUOTE

jscomps-entrylist  = jscomps-entry *(";" jscomps-entry)
jscomps-entry      = jscomps-entry-pos / jscomps-entry-sep
jscomps-entry-pos  = 1*DIGIT [ "," 1*DIGIT ]
jscomps-entry-sep  = "s" "," jscomps-entry-verb
jscomps-entry-verb = *QSAFE-CHAR ; encode according to RFC 6868
```

Example(s): The following example demonstrates the use of positional entries for the name "Jane Doe". The given name is ordered before the surname. No secondary index is required for either positional because both are zero.

```
"name": {
  "components": [
    { "kind": "given", "value": "Jane" },
    { "kind": "surname", "value": "Doe" }
  ],
  "isOrdered": true
}
```

```
N;JSCOMPS=";1;0":Doe;Jane;;;;;;
FN;DERIVED=TRUE:Jane Doe
```

*Figure 51: Example of a Secondary Positional Index*

The following example demonstrates a secondary positional index. The "Jr." generation marker only counts once because it occurs in both the designated generation component and the backwards-compatible honorific suffixes component.

```
"name": {
  "components": [
    { "kind": "given", "value": "John" },
    { "kind": "given2", "value": "Philip" },
    { "kind": "given2", "value": "Paul" },
    { "kind": "surname", "value": "Stevenson" },
    { "kind": "generation", "value": "Jr." },
    { "kind": "credential", "value": "M.D." }
  ],
  "isOrdered": true
}
```

```
N;JSCOMPS=";1;2;2,1;0;6;4,1":
 Stevenson;John;Philip,Paul;;Jr.,M.D.;;Jr.
```

*Figure 52: Example of Positional Entries*

The following example demonstrates the use of separator entries for the (shortened for brevity) address "54321 Oak St, Reston". The first entry defines the default separator to be ", ". The second and fourth positional entries are separated with the separator value " ". For backwards compatibility, the street address component of the ADR property contains both the street number and name, but it is not referred to in the JSCOMPS parameter and does not contribute to the count of values.

```
"addresses": {
  "a1": {
    "components": [
      { "kind": "number", "value": "54321" },
      { "kind": "separator", "value": " " },
      { "kind": "name", "value": "Oak St" },
      { "kind": "locality", "value": "Reston" }
    ],
    "defaultSeparator": ", ",
    "isOrdered": true
  }
}
```

```
ADR;JSCOMPS="s,\, ;11;s, ;10;3":
 ;;54321 Oak St;Reston;;;;;;;Oak St;54321;;;;;;
```

*Figure 53: Example of Separator Entries*

### 3.3.2. JSPTR

Parameter name: JSPTR

Purpose: This parameter is set on a JSPROP (Section 3.2.1) property. Its value is a JSON pointer [RFC6901] that points to the JSContact property that has the value of the JSPROP property.

Description: This parameter has a single value that **MUST** be a valid JSON pointer as defined in [RFC6901]. Note that the value **MUST** be quoted according to the "param-value" ABNF in [RFC6350].

Format definition:

```
jsptr-param  = "JSPTR" "=" param-value
             ; also see param-value in RFC 6350, Section 3.3
```

Example(s): This illustrates a simple example. For further examples, see Section 3.2.1.

```
JSPROP;JSPTR="example.com:foo":"bar"
```

# 4. Security Considerations

This specification defines how to convert between the JSContact and vCard formats. The security considerations for parsing and formatting such data apply and are outlined in Section 4 of [RFC9553] and Section 9 of [RFC6350].

# 5. IANA Considerations

## 5.1. New vCard Property

IANA has added the following entry to the "vCard Properties" registry, as defined in Section 10.3.1 of [RFC6350].

| Namespace | Property | Reference |
|---|---|---|
| | JSPROP | RFC 9555, Section 3.2.1 |

*Table 4: New vCard Property*

## 5.2. New vCard Parameter

IANA has added the following entry to the "vCard Parameters" registry, as defined in Section 10.3.2 of [RFC6350].

| Namespace | Parameter | Reference |
|---|---|---|
| | JSPTR | RFC 9555, Section 3.3.2 |

*Table 5: New vCard Parameter*

### 5.3. New JSContact Properties

IANA has added the following entries to the "JSContact Properties" registry. Note that the Since Version is 1.0, the Until Version is not set, and the Change Controller is IETF for all of these properties.

| Property Name | Property Type | Property Context | Intended Usage | Reference/ Description |
|---|---|---|---|---|
| vCardName | String | Any JSContact object | common | RFC 9555, Section 2.15.3 |
| vCardParams | String[String\|String[]] | Any JSContact object | common | RFC 9555, Section 2.15.2 |
| vCardProps | JCardProp[] | Card | common | RFC 9555, Section 2.15.1 |

*Table 6: JSContact Properties Registry*

### 5.4. New JSContact Type

IANA has added the following entry to the "JSContact Types" registry. Note that the Since Version is 1.0, the Until Version is not set, and the Change Controller is IETF for this type.

| Type Name | Intended Usage | Reference/Description |
|---|---|---|
| JCardProp | common | RFC 9555, Section 2.15.1 |

*Table 7: JSContact Types Registry*

## 6. References

### 6.1. 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>.

**[RFC3986]** Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

**[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>.

**[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>.

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

**[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>.

**[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>.

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

**[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>.

**[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>.

**[RFC9554]** Stepanek, R. and M. Loffredo, "vCard Format Extensions for JSContact", RFC 9554, DOI 10.17487/RFC9554, May 2024, <https://www.rfc-editor.org/info/rfc9554>.

## 6.2. Informative References

**[CLDRPersonName]**

Davis, M., Edberg, P., Gillam, R., Kolisnychenko, A., McKenna, M., and other CLDR committee members, "Unicode Locale Data Markup Language (LDML) Part 8: Person Names", Unicode Technical Standard #35, Version 44.1, July 2023, <https://www.unicode.org/reports/tr35/tr35-personNames.html>.

**[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>.

**[vOBJECT]** Tse, R., Tam, P., and M. Douglass, "vObject Internationalization", Work in Progress, Internet-Draft, draft-calconnect-vobject-i18n-00, 8 June 2018, <https://datatracker.ietf.org/doc/html/draft-calconnect-vobject-i18n-00>.

# Appendix A. Reverse Rules of Converting a vCard to a JSContact Card

Table 8 lists the relevant document sections for each JSContact type and property.

| JSContact Type | Property Name | Relevant Section(s) |
|---|---|---|
| Address | @type | not applicable |
| Address | components | Sections 2.6.1 and 3.3.1 |
| Address | contexts | Section 2.3.22 |
| Address | coordinates | Sections 2.3.8 and 2.8.1 |
| Address | country | Section 2.6.1 |
| Address | countryCode | Section 2.6.1 |
| Address | defaultSeparator | Sections 2.6.1 and 3.3.1 |
| Address | full | Section 2.6.1 |
| Address | isOrdered | Sections 2.6.1 and 3.3.1 |
| Address | locality | Section 2.6.1 |
| Address | phoneticScript | Sections 2.3.15 and 2.3.19 |
| Address | phoneticSystem | Section 2.3.15 |
| Address | postcode | Section 2.6.1 |
| Address | pref | Section 2.3.17 |

| JSContact Type | Property Name | Relevant Section(s) |
|---|---|---|
| Address | region | Section 2.6.1 |
| Address | timeZone | Sections 2.3.23 and 2.8.2 |
| AddressComponent | phonetic | Section 2.3.15 |
| Anniversary | @type | not applicable |
| Anniversary | date | Section 2.5.1 |
| Anniversary | kind | Section 2.5.1 |
| Anniversary | place | Section 2.5.1 |
| Author | @type | not applicable |
| Author | name | Section 2.3.3 |
| Author | uri | Section 2.3.2 |
| Calendar | @type | not applicable |
| Calendar | contexts | Section 2.3.22 |
| Calendar | kind | Sections 2.13.2 and 2.13.3 |
| Calendar | label | Section 2.11.11 |
| Calendar | mediaType | Section 2.3.14 |
| Calendar | pref | Section 2.3.17 |
| Calendar | uri | Sections 2.13.2 and 2.13.3 |
| Card | @type | not applicable |
| Card | @version | not applicable |
| Card | addresses | Section 2.6.1 |
| Card | anniversaries | Section 2.5.1 |
| Card | calendars | Sections 2.13.2 and 2.13.3 |
| Card | created | Section 2.11.3 |
| Card | directories | Sections 2.4.3 and 2.10.4 |

| JSContact Type | Property Name | Relevant Section(s) |
|---|---|---|
| Card | emails | Section 2.7.1 |
| Card | keywords | Section 2.11.1 |
| Card | kind | Section 2.4.2 |
| Card | language | Section 2.7.4 |
| Card | links | Sections 2.9.1 and 2.11.9 |
| Card | localizations | Section 2.3.11 |
| Card | media | Sections 2.5.7, 2.9.2, and 2.11.7 |
| Card | members | Section 2.9.3 |
| Card | name | Section 2.5.5 |
| Card | nicknames | Section 2.5.6 |
| Card | notes | Section 2.11.4 |
| Card | onlineServices | Section 2.7.2 |
| Card | organizations | Section 2.9.4 |
| Card | personalInfo | Sections 2.10.1, 2.10.2, and 2.10.3 |
| Card | phones | Section 2.7.6 |
| Card | preferredLanguages | Section 2.7.3 |
| Card | prodId | Section 2.11.5 |
| Card | relatedTo | Section 2.9.5 |
| Card | schedulingAddresses | Section 2.13.1 |
| Card | speakToAs | Section 2.5.4 |
| Card | titles | Section 2.9.6 |
| Card | uid | Section 2.11.8 |
| Card | updated | Section 2.11.6 |
| CryptoKey | @type | not applicable |

| JSContact Type | Property Name | Relevant Section(s) |
|---|---|---|
| CryptoKey | contexts | Section 2.3.22 |
| CryptoKey | kind | not applicable |
| CryptoKey | label | Section 2.11.11 |
| CryptoKey | mediaType | Section 2.3.14 |
| CryptoKey | pref | Section 2.3.17 |
| CryptoKey | uri | Section 2.12.1 |
| Directory | @type | not applicable |
| Directory | contexts | Section 2.3.22 |
| Directory | kind | Sections 2.4.3 and 2.10.4 |
| Directory | label | Section 2.11.11 |
| Directory | listAs | Section 2.3.10 |
| Directory | mediaType | Section 2.3.14 |
| Directory | pref | Section 2.3.17 |
| Directory | uri | Sections 2.4.3 and 2.10.4 |
| EmailAddress | @type | not applicable |
| EmailAddress | address | Section 2.7.1 |
| EmailAddress | contexts | Section 2.3.22 |
| EmailAddress | label | Section 2.11.11 |
| EmailAddress | pref | Section 2.3.17 |
| LanguagePref | @type | not applicable |
| LanguagePref | contexts | Section 2.3.22 |
| LanguagePref | pref | Section 2.3.17 |
| Link | @type | not applicable |
| Link | contexts | Section 2.3.22 |

| JSContact Type | Property Name | Relevant Section(s) |
|---|---|---|
| Link | kind | Section 2.9.1 |
| Link | label | Section 2.11.11 |
| Link | mediaType | Section 2.3.14 |
| Link | pref | Section 2.3.17 |
| Link | uri | Sections 2.9.1 and 2.11.9 |
| Media | @type | not applicable |
| Media | contexts | Section 2.3.22 |
| Media | kind | Sections 2.5.7, 2.9.2, and 2.11.7 |
| Media | label | Section 2.11.11 |
| Media | mediaType | Section 2.3.14 |
| Media | pref | Section 2.3.17 |
| Media | uri | Sections 2.5.7, 2.9.2, and 2.11.7 |
| Name | @type | not applicable |
| Name | components | Sections 2.5.5 and 3.3.1 |
| Name | defaultSeparator | Sections 2.5.5 and 3.3.1 |
| Name | full | Section 2.5.2 |
| Name | phoneticScript | Sections 2.3.15 and 2.3.19 |
| Name | phoneticSystem | Section 2.3.15 |
| Name | isOrdered | Sections 2.5.5 and 3.3.1 |
| Name | sortAs | Section 2.3.21 |
| NameComponent | @type | not applicable |
| NameComponent | kind | Section 2.5.5 |
| NameComponent | phonetic | Section 2.3.15 |
| NameComponent | value | Section 2.5.5 |

| JSContact Type | Property Name | Relevant Section(s) |
|---|---|---|
| Nickname | @type | not applicable |
| Nickname | contexts | Section 2.3.22 |
| Nickname | name | Section 2.5.5 |
| Nickname | pref | Section 2.3.17 |
| Note | @type | not applicable |
| Note | author | Sections 2.3.2 and 2.3.3 |
| Note | created | Section 2.3.6 |
| Note | note | Section 2.11.4 |
| OnlineService | @type | not applicable |
| OnlineService | contexts | Section 2.3.22 |
| OnlineService | label | Section 2.11.11 |
| OnlineService | pref | Section 2.3.17 |
| OnlineService | service | Section 2.3.20 |
| OnlineService | uri | Sections 2.7.2 and 2.7.5 |
| OnlineService | user | Section 2.3.24 |
| OrgUnit | @type | not applicable |
| OrgUnit | name | Section 2.9.4 |
| OrgUnit | sortAs | Section 2.3.21 |
| Organization | @type | not applicable |
| Organization | contexts | Section 2.3.22 |
| Organization | name | Section 2.9.4 |
| Organization | sortAs | Section 2.3.21 |
| Organization | units | Section 2.9.4 |
| PartialDate | @type | not applicable |

| JSContact Type | Property Name | Relevant Section(s) |
|---|---|---|
| PartialDate | calendarScale | Section 2.3.4 |
| PartialDate | day | Section 2.2.2 |
| PartialDate | month | Section 2.2.2 |
| PartialDate | year | Section 2.2.2 |
| PatchObject | @type | not applicable |
| PersonalInfo | @type | not applicable |
| PersonalInfo | kind | Sections 2.10.1, 2.10.2, and 2.10.3 |
| PersonalInfo | listAs | Section 2.3.10 |
| PersonalInfo | level | Section 2.3.13 |
| PersonalInfo | value | Sections 2.10.1, 2.10.2, and 2.10.3 |
| Phone | @type | not applicable |
| Phone | contexts | Section 2.3.22 |
| Phone | features | Section 2.7.6 |
| Phone | label | Section 2.11.11 |
| Phone | number | Section 2.7.6 |
| Phone | pref | Section 2.3.17 |
| Pronouns | @type | not applicable |
| Pronouns | contexts | Section 2.3.22 |
| Pronouns | pref | Section 2.3.17 |
| Pronouns | pronouns | Section 2.5.4 |
| Relation | @type | not applicable |
| Relation | relation | Section 2.9.5 |
| Resource | @type | not applicable |
| SchedulingAddress | @type | not applicable |

| JSContact Type | Property Name | Relevant Section(s) |
|---|---|---|
| SchedulingAddress | contexts | Section 2.3.22 |
| SchedulingAddress | label | Section 2.11.11 |
| SchedulingAddress | pref | Section 2.3.17 |
| SchedulingAddress | uri | Section 2.13.1 |
| SpeakToAs | @type | not applicable |
| SpeakToAs | grammaticalGender | Section 2.5.4 |
| SpeakToAs | pronouns | Section 2.5.4 |
| AddressComponent | @type | not applicable |
| AddressComponent | kind | Section 2.6.1 |
| AddressComponent | value | Section 2.6.1 |
| Timestamp | @type | not applicable |
| Timestamp | utc | Section 2.2.2 |
| Title | @type | not applicable |
| Title | kind | Section 2.9.6 |
| Title | name | Section 2.9.6 |
| Title | organizationId | Section 2.9.6 |

*Table 8: Conversion Rules for JSContact Types and Properties*

## Acknowledgements

The definition and examples of the PHONETIC (Section 2.3.15) and SCRIPT (Section 2.3.19) parameters are based on the initial draft version of [vOBJECT].

## Authors' Addresses

**Mario Loffredo**
IIT-CNR/Registro.it
Via Moruzzi, 1
56124 Pisa
Italy
Email: mario.loffredo@iit.cnr.it
URI: https://www.iit.cnr.it

**Robert Stepanek**
Fastmail
PO Box 234
Collins St. West
Melbourne VIC 8007
Australia
Email: rsto@fastmailteam.com
URI: https://www.fastmail.com