

Workgroup: calext
Internet-Draft:
draft-ietf-calext-jscontact-vcard-14
Updates: [6350](#) (if approved)
Published: 9 November 2023
Intended Status: Standards Track
Expires: 12 May 2024
Authors: M. Loffredo R. Stepanek
 IIT-CNR/Registro.it Fastmail

JSContact: Converting from and to vCard

Abstract

This document defines how to convert contact information between the JSContact and vCard data formats. To achieve this, it updates RFC I-D.ietf-calext-jscontact (JSContact) by registering new JSContact properties. Similarly, it updates RFC 6350 (vCard) by registering new vCard properties and parameters.

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 12 May 2024.

Copyright Notice

Copyright (c) 2023 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 include Revised BSD License text as described in Section 4.e of the

Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. [Introduction](#)
 - 1.1. [Motivation](#)
 - 1.2. [Notational Conventions](#)
 - 1.3. [ABNF Notations](#)
2. [Converting vCard to JSContact](#)
 - 2.1. [General rules](#)
 - 2.1.1. [The uid property](#)
 - 2.1.2. [Choosing identifiers](#)
 - 2.2. [vCard Value Data Types](#)
 - 2.2.1. [BOOLEAN](#)
 - 2.2.2. [DATE, TIME, DATE-TIME, DATE-AND-OR-TIME, and TIMESTAMP](#)
 - 2.2.3. [INTEGER](#)
 - 2.2.4. [FLOAT](#)
 - 2.2.5. [LANGUAGE-TAG](#)
 - 2.2.6. [TEXT](#)
 - 2.2.7. [URI](#)
 - 2.2.8. [UTC-OFFSET](#)
 - 2.3. [vCard Parameters](#)
 - 2.3.1. [ALTID](#)
 - 2.3.2. [AUTHOR](#)
 - 2.3.3. [AUTHOR-NAME](#)
 - 2.3.4. [CALSCALE](#)
 - 2.3.5. [CREATED](#)
 - 2.3.6. [DERIVED](#)
 - 2.3.7. [GEO](#)
 - 2.3.8. [GROUP](#)
 - 2.3.9. [INDEX](#)
 - 2.3.10. [LANGUAGE](#)
 - 2.3.11. [LEVEL](#)
 - 2.3.12. [MEDIATYPE](#)
 - 2.3.13. [PHONETIC](#)
 - 2.3.14. [PID](#)

- [2.3.15. PREF](#)
- [2.3.16. PROP-ID](#)
- [2.3.17. SCRIPT](#)
- [2.3.18. SERVICE-TYPE](#)
- [2.3.19. SORT-AS](#)
- [2.3.20. TYPE](#)
- [2.3.21. TZ](#)
- [2.3.22. USERNAME](#)
- [2.3.23. VALUE](#)
- [2.4. General Properties](#)
 - [2.4.1. BEGIN and END](#)
 - [2.4.2. KIND](#)
 - [2.4.3. SOURCE](#)
 - [2.4.4. XML](#)
- [2.5. Identification Properties](#)
 - [2.5.1. BDAY, BIRTHPLACE, DEATHDATE, DEATHPLACE, ANNIVERSARY](#)
 - [2.5.2. FN](#)
 - [2.5.3. GENDER](#)
 - [2.5.4. GRAMGENDER and PRONOUNS](#)
 - [2.5.5. N](#)
 - [2.5.6. NICKNAME](#)
 - [2.5.7. PHOTO](#)
- [2.6. Delivery Addressing Properties](#)
 - [2.6.1. ADR](#)
- [2.7. Communications Properties](#)
 - [2.7.1. EMAIL](#)
 - [2.7.2. IMPP](#)
 - [2.7.3. LANG](#)
 - [2.7.4. LANGUAGE](#)
 - [2.7.5. SOCIALPROFILE](#)
 - [2.7.6. TEL](#)
- [2.8. Geographical Properties](#)
 - [2.8.1. GEO](#)
 - [2.8.2. TZ](#)
 - [2.8.3. Combining geographical properties](#)
- [2.9. Organizational Properties](#)
 - [2.9.1. CONTACT-URI](#)
 - [2.9.2. LOGO](#)
 - [2.9.3. MEMBER](#)
 - [2.9.4. ORG](#)
 - [2.9.5. RELATED](#)
 - [2.9.6. TITLE and ROLE](#)
- [2.10. Personal Information Properties](#)
 - [2.10.1. EXPERTISE](#)
 - [2.10.2. HOBBY](#)
 - [2.10.3. INTEREST](#)
 - [2.10.4. ORG-DIRECTORY](#)
- [2.11. Explanatory Properties](#)
 - [2.11.1. CATEGORIES](#)

- [2.11.2. CLIENTPIDMAP and PID Parameter](#)
 - [2.11.3. CREATED](#)
 - [2.11.4. NOTE](#)
 - [2.11.5. PRODIG](#)
 - [2.11.6. REV](#)
 - [2.11.7. SOUND](#)
 - [2.11.8. UID](#)
 - [2.11.9. URL](#)
 - [2.11.10. VERSION](#)
 - [2.11.11. X-ABLabel](#)
 - [2.12. Security Properties](#)
 - [2.12.1. KEY](#)
 - [2.13. Calendar Properties](#)
 - [2.13.1. CALADRURI](#)
 - [2.13.2. CALURI](#)
 - [2.13.3. FBURL](#)
 - [2.14. Extended Properties and Parameters](#)
 - [2.15. New JSContact properties](#)
 - [2.15.1. Property vCardProps](#)
 - [2.15.2. Property vCardParams](#)
 - [2.15.3. Property vCardName](#)
- [3. Converting JSContact to vCard](#)
 - [3.1. Conversion Rules](#)
 - [3.1.1. Converting unknown properties](#)
 - [3.2. New vCard Properties](#)
 - [3.2.1. JSPROP](#)
 - [3.3. New vCard Parameter](#)
 - [3.3.1. JSCOMPS](#)
 - [3.3.2. JSPTR](#)
- [4. Security Considerations](#)
- [5. Acknowledgements](#)
- [6. IANA Considerations](#)
 - [6.1. New vCard Properties](#)
 - [6.2. New vCard Parameters](#)
 - [6.3. New JSContact Properties](#)
 - [6.4. New JSContact Types](#)
- [7. Implementation Status](#)
 - [7.1. CNR](#)
- [8. References](#)
 - [8.1. Normative References](#)
 - [8.2. Informative References](#)
- [Appendix A. Reverse Rules of Converting vCard to JSContact Card Authors' Addresses](#)

1. Introduction

1.1. Motivation

The JSContact data model and format [[I-D.ietf-calexext-jscontact](#)] aims to be an alternative to the widely used vCard [[RFC6350](#)] standard and jCard [[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 want to represent their contact data also 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 either are defined in [[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 vCard to JSContact Card. It follows the same structure as the [vCard v4](#) [[RFC6350](#)]. Properties and parameters of vCard extension RFCs, including the [vCard JSContact Extension](#) [[I-D.ietf-calexext-vcards-jscontact-extensions](#)] are added to appropriate subsections.

2.1. General rules

2.1.1. The uid property

The UID property in vCard is optional, but the uid property in JSContact is mandatory. Implementations that convert a vCard without UID property **MUST** generate a unique identifier as a 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 typically are represented as a JSON object where the object keys are of the Id type and the object values are the converted vCard property. In absence of the PROP-ID parameter (see [Section 2.3.16](#)), implementations are free to choose any identifier 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 [\[I-D.ietf-calext-jscontact\]](#). For example, this could be an incrementing number across all Ids or only unique within one JSON object.

2.2. vCard Value Data Types

2.2.1. BOOLEAN

This converts to the JSContact Boolean type.

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

The TIMESTAMP type generally converts to the UTCDateTime. It converts to the Timestamp type for anniversaries.

The DATE type converts to the PartialDate type, which only is relevant for anniversaries. This does not apply to DATE values that only define a month or a day.

The TIME, DATE-TIME, DATE-AND-OR-TIME types and DATE type values that only define a month or day do not convert to a JSContact datetime type. Instead, vCard properties or parameters having such values convert to the properties defined in [Section 2.15](#).

2.2.3. INTEGER

This converts to the JSContact Int and UnsignedInt types.

2.2.4. FLOAT

This converts to the JSContact Number type.

2.2.5. LANGUAGE-TAG

This converts to the JSContact String type.

2.2.6. TEXT

This converts to the JSContact String type.

2.2.7. URI

This converts to the JSContact String type.

2.2.8. UTC-OFFSET

This either converts to a String containing an IANA TimeZone Database entry name (see [Section 2.8.2](#)), or it does not convert to any JSContact type. For the latter, vCard properties or parameters having such values **MAY** convert to JSContact as defined in [Section 2.15](#).

2.3. vCard Parameters

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

2.3.1. ALTID

This 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 value of a vCard NOTE property converts to the uri property of the Author object for this note.

Note: This parameter is defined in [\[I-D.ietf-calext-vcard-jscontact-extensions\]](#).

2.3.3. AUTHOR-NAME

The AUTHOR-NAME parameter value of a vCard NOTE property converts to the name property of the Author object for this note.

Note: This parameter is defined in [\[I-D.ietf-calext-vcard-jscontact-extensions\]](#).

2.3.4. CALSCALE

This parameter set on a BDAY, DEATHDATE or ANNIVERSARY property converts to the calendarScale property of the PartialDate object type.

2.3.5. CREATED

The CREATED parameter value of a vCard NOTE property converts to the created property of the Note object.

Note: This parameter is defined in [\[I-D.ietf-calext-vcard-jscontact-extensions\]](#).

2.3.6. DERIVED

If this parameter is set to true on a vCard property, then implementations **MAY** choose to not convert that property.

Note: This parameter is defined in [\[I-D.ietf-calext-vcard-jscontact-extensions\]](#).

2.3.7. GEO

This parameter set on an ADR property converts to the JSContact coordinates property of the Address object that represents the vCard ADR.

2.3.8. GROUP

This parameter exclusively is for use in jCard (see [Section 7.1](#) of [\[RFC7095\]](#)). It **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 group parameter in the JSContact properties vCardProps or vCardParams 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: An example how to preserve the group name in vCardParams during conversion.

```
item2.X-FOO:bar

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

Figure 2: An example how to preserve the group name in vCardProps during conversion.

2.3.9. INDEX

This parameter set on the EXPERTISE, HOBBY, INTEREST and ORG-DIRECTORY properties defined in [[RFC6715](#)] converts to the JSContact listAs property of the PersonalInfo and Directory objects.

2.3.10. LANGUAGE

This converts to an entry in the localizations property for the vCard property that 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 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;LANGUAGE=EN:Boss
TITLE;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, then choose them to the non-localized Card. Convert any with LANGUAGE parameters to the localizations property. [Figure 4](#) illustrates this.

```
FN:John Doe
TITLE:Boss
TITLE;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

As the least preferred option, [Figure 5](#) illustrates how all vCard properties of the same type have the LANGUAGE parameter set, but none of their language tags match the language of the main Card. In this

case, implementations **MAY** choose to add the localized vCard properties only to the localizations object.

The following example uses non-ASCII characters to demonstrate multilingual content. The ASCII-formatted version of this specification might not show the correct characters. Please refer to the HTML or PDF versions instead.

```
LANGUAGE:es
FN:Gabriel García Márquez
TITLE;LANGUAGE=en:Boss
TITLE;LANGUAGE=fr:Patron

"language": "es",
"name" {
  "full": "Gabriel García Márquez"
},
"localizations": {
  "en": {
    "titles": {
      "t1": {
        "name": "Novelist"
      }
    },
  },
  "fr": {
    "titles": {
      "t1": {
        "name": "Écrivain"
      }
    }
  }
}
```

Figure 5: LANGUAGE conversion example: conflicting LANGUAGE property and parameter values

2.3.11. LEVEL

The LEVEL parameter [[RFC6715](#)] converts to the level property of the PersonalInfo type. If this parameter is set on the EXPERTISE property, then its values convert as:

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

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

2.3.12. MEDIATYPE

This converts to the `mediaType` property of the Resource object type.

2.3.13. PHONETIC

This parameter as well as the [SCRIPT \(Section 2.3.17\)](#) parameter set on a N or ADR property convert to JSContact as follows:

the values of the `phonetic`, `phoneticScript` and `phoneticSystem` properties of the `NameComponent` and `Name` or `AddressComponent` and `Address` object types, respectively.

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 value of the PHONETIC parameter converts to the `phoneticSystem` property unless it is `script`, in which case the `phoneticSystem` property is not set. The value of the SCRIPT parameter converts to the `phoneticScript` property.

The values of the components in the property value convert to values the `phonetic` properties for the respective `NameComponent` or `AddressComponent`.

If more than one property having the PHONETIC parameter set relates to the same property, then they convert to the `Card localizations` property according their LANGUAGE parameter values as outlined in [Section 2.3.10](#).

```

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

"language": "zh-Hant",
"name": {
  "components": [
    { "kind": "surname", "value": " " },
    { "kind": "given", "value": " " },
    { "kind": "given2", "value": " " },
    { "kind": "given2", "value": " " }
  ]
},
"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 6: PHONETIC conversion example

2.3.14. PID

This converts to an entry in the vCardParams property, see [Section 2.15.2](#).

2.3.15. PREF

This converts to the pref property.

2.3.16. PROP-ID

The PROP-ID parameter value of a vCard property converts to the Id of the JSContact property to which the vCard property converts.

```

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 7: PROP-ID conversion example

Note: This parameter is defined in [\[I-D.ietf-calex-vcards-jscontact-extensions\]](#).

2.3.17. SCRIPT

See [Section 2.3.13](#).

2.3.18. SERVICE-TYPE

This converts to the service property of the OnlineService object type.

Note: This parameter is defined in [\[I-D.ietf-calex-vcards-jscontact-extensions\]](#).

2.3.19. SORT-AS

This converts to the sortAs properties defined for the Name, Organization and OrgUnit object types.

2.3.20. TYPE

This converts to the contexts property as well as property-specific kind property values defined in later sections. If not specified otherwise for a specific JSContact property, the vCard home and work parameter values convert to the JSContact private and work contexts, respectively.

2.3.21. TZ

This parameter set on an ADR property converts to the JSContact `timeZone` property of the Address object that represents the vCard ADR. Also see the conversion of the TZ property in [Section 2.8.2](#).

2.3.22. USERNAME

This converts to the user property of the OnlineService object type.

Note: This parameter is defined in [\[I-D.ietf-calext-vcard-jscontact-extensions\]](#).

2.3.23. VALUE

This does not convert to an IANA-registered property in JSContact. To preserve properties with experimental values, see [Section 2.15.2](#) and [Section 2.15.1](#).

2.4. General Properties

2.4.1. BEGIN and END

These do not convert to IANA-registered properties in JSContact.

2.4.2. KIND

The KIND property converts to the kind property ([Figure 8](#)). 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 8: KIND conversion example

2.4.3. SOURCE

The SOURCE property converts to a Directory object in the `directories` property ([Figure 9](#)). The 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 as 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 9: SOURCE conversion example

2.4.4. XML

This converts to an entry in the vCardProps property, see [Section 2.15.1](#).

2.5. Identification Properties

2.5.1. BDAY, BIRTHPLACE, DEATHDATE, DEATHPLACE, ANNIVERSARY

The BDAY and ANNIVERSARY properties and the extensions BIRTHPLACE, DEATHDATE, DEATHPLACE described in [[RFC6474](#)] are represented as Anniversary objects included in the anniversaries property ([Figure 10](#)):

*BDAY and BIRTHPLACE convert to date and place where kind is set to "birth";

*DEATHDATE and DEATHPLACE convert to date and place where kind is set to "death";

*ANNIVERSARY converts to date where kind is "wedding".

Both birth and death places are represented as instances of the Address object.

The BIRTHPLACE and DEATHPLACE properties that are represented as geo URIs convert to Address instances including only the coordinates property. If the URI value is not a geo URI, the place is ignored.

The ALTID and LANGUAGE parameters of both BIRTHPLACE and DEATHPLACE properties convert according to the rules as 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 10: BDAY, BIRTHPLACE, DEATHDATE, DEATHPLACE, ANNIVERSARY conversion example

2.5.2. FN

The FN property converts to the Name object in the name property. Its value converts to the Name object full property ([Figure 11](#)). If the LANGUAGE parameter is set, then the FN property converts as outlined

in [Section 2.3.10](#). In the unexpected case that the vCard contains more than one FN property without LANGUAGE parameter, then convert the FN property that has the least parameters set to the full property. 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 11: FN conversion example

2.5.3. GENDER

This does not map to an IANA-registered property in JSContact. To convert this property, see [Section 2.15.1](#). Note the alternative JSContact speakToAs property which defines how to address and refer to an individual represented by the card, as do the newly defined vCard GRAMGENDER and PRONOUNS properties of [\[I-D.ietf-calext-vcard-jscontact-extensions\]](#).

2.5.4. GRAMGENDER and PRONOUNS

The GRAMGENDER property converts to the grammaticalGender property of the SpeakToAs object ([Figure 12](#)).

The PRONOUNS property converts to an entry in the pronouns property of the SpeakToAs object ([Figure 12](#)).

```

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 12: GRAMGENDER and PRONOUNS conversion example

2.5.5. N

The N property converts to the Name object in the name property. Each component in the N property structured value converts to a NameComponent in the Name components property. The following table shows this relation:

N component	NameComponent kind	Remarks
Family name	surname	To vCard: add any surname2 NameComponent value in here, after all surname values. 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 value also in here. 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 isOrdered property value is "true", and the defaultSeparator and any separator name components are set according to the parameter value. The components list **MUST** adhere to the order of the JSCOMPS parameter value.

If the JSCOMPS parameter is not set, then the Name isOrdered property value is "false" and the defaultSeparator property **MUST NOT** be set. The components list **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 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

```
"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 13: N conversion example

See [Section 3.3.1](#) for examples of using the JSCOMPS parameter for vCard structured property values.

2.5.6. NICKNAME

A NICKNAME property converts to Nickname object in the nicknames property ([Figure 14](#)). The name property is set to the NICKNAME property value.

The PREF and TYPE parameters convert according to the rules as defined in [Section 2.3](#).

NICKNAME:Johnny

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

Figure 14: NICKNAME conversion example

2.5.7. PHOTO

A PHOTO property converts to an entry in the media property (Figure 15). The entry value is a Media object whose kind property is set to photo and uri property is set to the PHOTO value.

The PREF and MEDIATYPE parameters convert according to the rules as 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 15: PHOTO conversion example

2.6. Delivery Addressing Properties

2.6.1. ADR

The ADR property converts to an Address object in the addresses property. Each component in the ADR property structured value converts to an AddressComponent in the Address components property.

[I-D.ietf-calext-vcard-jscontact-extensions] defines new component for the ADR property. Implementations **SHOULD** set these new components, even if all their values are the empty string.

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

ADR component	AddressComponent kind	Remarks
	postOfficeBox	

ADR component	AddressComponent kind	Remarks
post office box		[RFC6350] recommends this component not be set, but this now is disputable given the new components. Instead, set this component and use the new ADR value format defined in [I-D.ietf-calext-vcard-jscontact-extensions].
extended address	apartment (see Remarks)	<p>To vCard: Set the values of the following components:</p> <ul style="list-style-type: none"> *room *floor *apartment *building <p>From vCard: Ignore if the ADR structured value is of the format defined in [I-D.ietf-calext-vcard-jscontact-extensions]. Otherwise convert to apartment.</p>
street address	name (see Remarks)	<p>To vCard: Set the values of the following components:</p> <ul style="list-style-type: none"> *number *name *block *direction *landmark *subdistrict *district <p>From vCard: Ignore if the ADR structured value is of the format defined in [I-D.ietf-calext-vcard-jscontact-extensions]. Otherwise convert to name.</p>
locality	locality	
region	region	
postal code	postcode	
apartment	apartment	Defined in [I-D.ietf-calext-vcard-jscontact-extensions].
block	block	

ADR component	AddressComponent kind	Remarks
		Defined in [I-D.ietf-calext-vcard-jscontact-extensions] .
building	building	Defined in [I-D.ietf-calext-vcard-jscontact-extensions] .
direction	direction	Defined in [I-D.ietf-calext-vcard-jscontact-extensions] .
district	district	Defined in [I-D.ietf-calext-vcard-jscontact-extensions] .
floor	floor	Defined in [I-D.ietf-calext-vcard-jscontact-extensions] .
landmark	landmark	Defined in [I-D.ietf-calext-vcard-jscontact-extensions] .
room	room	Defined in [I-D.ietf-calext-vcard-jscontact-extensions] .
street number	number	Defined in [I-D.ietf-calext-vcard-jscontact-extensions] .
subdistrict	subdistrict	Defined in [I-D.ietf-calext-vcard-jscontact-extensions] .

Table 2: ADR components conversion

If the [JSCOMPS](#) ([Section 3.3.1](#)) parameter is set, then the Address `isOrdered` property value is "true", and the `defaultSeparator` and any separator name components are set according to the parameter value. The components list **MUST** adhere to the order of the JSCOMPS parameter value.

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

The LABEL parameter converts to the Address full property.

The GEO parameter converts to the Address coordinates property.

The TZ parameter converts to the Address `timeZone` property.

The CC parameter as defined in [\[RFC8605\]](#) converts to the Address `countryCode` property.

The PREF and TYPE parameters convert according to the rules as defined in [Section 2.3](#). The ADR-specific values of the TYPE parameter defined in Sections 5.1 and 5.2 of [\[I-D.ietf-calext-vcard-jscontact-extensions\]](#) convert to the corresponding entries of the `contexts` property as defined in Section 2.5.1 of [\[I-D.ietf-calext-jscontact\]](#).

The ALTID and LANGUAGE parameters convert according to the rules as defined in [Section 2.3](#). Each possible language-dependent alternative is represented as an entry of the PatchObject map where the key references the full property.

```
ADR;TYPE=work;CC=US;;;54321 Oak St;Reston;VA;20190;USA;;;Oak St;54321;;  
  
"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 16: 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

An EMAIL property converts to an entry in the emails property ([Figure 17](#)). The entry value is an EmailAddress object. The address property is set to the EMAIL value.

The PREF and TYPE parameters convert according to the rules as 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 17: EMAIL conversion example

2.7.2. IMPP

An IMPP property converts to an entry in the `onlineServices` property ([Figure 18](#)). The entry value is an `OnlineService` object. 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 as 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 18: IMPP conversion example

2.7.3. LANG

A `LANG` property converts to an entry in the `preferredLanguages` property ([Figure 19](#)). The entry value is a `LanguagePref` object. The `LanguagePref` `language` property value is the `LANG` property value.

The `PREF` and `TYPE` parameters convert according to the rules as 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 19: LANG conversion example

2.7.4. LANGUAGE

The LANGUAGE property converts to the language property ([Figure 20](#)).

```

LANGUAGE:de-AT

"language": "de-AT"

```

Figure 20: LANGUAGE conversion example

Note: This property is defined in [\[I-D.ietf-calext-vcard-jscontact-extensions\]](#).

2.7.5. SOCIALPROFILE

A SOCIALPROFILE property converts to an entry in the onlineServices property ([Figure 21](#)). The entry value is an OnlineService object. The vCardName property is set to "socialprofile" or can be omitted. If the value type of the SOCIALPROFILE is URI, the uri property is set to the SOCIALPROFILE value. Otherwise, the user property is set to the SOCIALPROFILE value.

The SERVICE-TYPE, USERNAME, PREF and TYPE parameters convert according to the rules as defined in [Section 2.3](#).

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

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

Figure 21: SOCIALPROFILE conversion example

Note: This property is defined in [\[I-D.ietf-calext-vcard-jscontact-extensions\]](#).

2.7.6. TEL

A TEL property converts to an entry in the phones property ([Figure 22](#)). The entry value is a Phone object.

The TEL-specific values of the TYPE parameter convert to the features property keys as outlined in [table Table 3](#). Note that [Section 6.4.1](#) of [\[RFC6350\]](#) defines the default TYPE of TEL to be voice, but the default Phone features property is absent by default. Accordingly, an implementation **SHOULD** only set the Phone 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
video	video
voice	voice

Table 3: TEL TYPE conversion

The value of the TEL property converts to the Phone number property.

The PREF and TYPE parameters convert according to the rules as 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 22: TEL conversion example

2.8. Geographical Properties

2.8.1. GEO

This converts to the coordinates property of the Address object. Also see [Section 2.8.3](#) to determine which Address object instance to convert to.

2.8.2. TZ

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

A value of type UTC-OFFSET converts to the timeZone property in the Address object if the offset has zero minutes and the hour offset is in the range $-12 \leq 14$:

*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 are set on any other ADR, GEO and TZ properties.

2.9. Organizational Properties

2.9.1. CONTACT-URI

A CONTACT-URI property as defined in [[RFC8605](#)] is represented as an entry of the links property ([Figure 23](#)). The entry value is a Link object whose kind property is set to contact and uri property is set to the CONTACT-URI value.

The PREF and TYPE parameters convert according to the rules as 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 23: CONTACT-URI conversion example

2.9.2. LOGO

A LOGO property converts to an entry in the media property ([Figure 24](#)). The entry value is a Media object whose kind property is set to logo and uri property is set to the LOGO value.

The PREF and TYPE parameters convert according to the rules as 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 24: LOGO conversion example

2.9.3. MEMBER

The uids of the contact cards composing the group are included in the members property ([Figure 25](#)).

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

```
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 25: Group example

2.9.4. ORG

An ORG property converts to an entry in the organizations property ([Figure 26](#)). The entry value is an Organization object whose name property contains the organizational name, and the units property is an array of OrgUnit objects each containing the organizational unit name in the name property.

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, LANGUAGE parameters convert according to the rules as 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 as 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 26: ORG conversion example

2.9.5. RELATED

This converts to an entry in the relatedTo property ([Figure 27](#)). The property value converts to the key in the relatedTo property. The TYPE parameters convert to the relation of the Relation object value. 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 27: RELATED conversion example

2.9.6. TITLE and ROLE

Both TITLE and ROLE properties are represented as entries of the titles property ([Figure 28](#)). The entry value is a Title object whose kind property is set to title or role for 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 member of a vCard property group and exactly one other ORG property also is part of that group.

The ALTID and LANGUAGE parameters convert according to the rules as defined in [Section 2.3](#).


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

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

Figure 28: TITLE and ROLE conversion example

2.10. Personal Information Properties

2.10.1. EXPERTISE

An EXPERTISE property as defined in [\[RFC6715\]](#) is represented as a PersonalInfo object in the personalInfo property ([Figure 29](#)). The kind property is set to "expertise".

The INDEX parameter converts according to the rules as 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 29: EXPERTISE conversion example

2.10.2. HOBBY

A HOBBY property as defined in [[RFC6715](#)] is represented as a PersonalInfo object in the personalInfo property ([Figure 30](#)). The kind property is set to "hobby".

The INDEX parameter converts according to the rules as 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 30: HOBBY conversion example

2.10.3. INTEREST

An INTEREST property as defined in [RFC6715] is represented as a PersonalInfo object in the personalInfo property (Figure 31). The kind property is set to "interest".

The INDEX parameter converts according to the rules as 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 31: INTEREST conversion example

2.10.4. ORG-DIRECTORY

The ORG-DIRECTORY property [RFC6715] converts to a Directory object in the directories property (Figure 32). The 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 as defined in Section 2.3.

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

Figure 32: ORG-DIRECTORY conversion example

2.11. Explanatory Properties

2.11.1. CATEGORIES

A CATEGORIES property converts to a set of entries of the keywords property ([Figure 33](#)). 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 map entries by order of preference.

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

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

Figure 33: CATEGORIES conversion example

2.11.2. CLIENTPIDMAP and PID Parameter

These convert to the [vCardProps](#) ([Section 2.15.1](#)) and [vCardParams](#) ([Section 2.15.2](#)) properties.

2.11.3. CREATED

The CREATED property converts to the created property ([Figure 34](#)).

```
CREATED:19940930T143510Z
```

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

Figure 34: CREATED conversion example

Note: This property is defined in [\[I-D.ietf-calext-vcard-jscontact-extensions\]](#).

2.11.4. NOTE

A NOTE property converts to a Note object in the notes map ([Figure 35](#)).

The ALTID and LANGUAGE parameters convert according to the rules as 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 35: NOTE conversion example

2.11.5. PROPID

The PROPID property converts to the prodId property ([Figure 36](#)).

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

Figure 36: PROPID conversion example

2.11.6. REV

The REV property converts to the updated property ([Figure 37](#)).

REV:19951031T222710Z

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

Figure 37: REV conversion example

2.11.7. SOUND

A SOUND property converts to an entry in the media property ([Figure 38](#)). The entry value is a Media object whose kind property is set to sound and uri property is set to the SOUND value.

The PREF and TYPE parameters convert according to the rules as 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 38: SOUND conversion example

2.11.8. UID

The UID property corresponds to the uid property ([Figure 39](#)).

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

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

Figure 39: UID conversion example

2.11.9. URL

A URL property converts to an entry in the links property ([Figure 40](#)). The entry value is a Link object whose uri property is set to the URL value.

The PREF and TYPE parameters convert according to the rules as 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 40: URL conversion example

2.11.10. VERSION

This converts to an entry in the [vCardProps](#) ([Section 2.15.1](#)) property.

2.11.11. X-ABLabel

This property is experimental but widely in use in existing vCard data. It converts to the label property of a JSContact object type. The X-ABLabel property is preceded by a vCard property group name, and the label converts to the JSContact object which got converted from a vCard property having 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 how to preserve the group name see [Section 2.3.8](#).

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

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

Figure 41: X-ABLabel conversion example

2.12. Security Properties

2.12.1. KEY

A KEY property converts to an entry in the cryptoKeys property ([Figure 42](#)). The entry value is a CryptoKey object whose uri property is set to the KEY value.

The PREF and TYPE parameters convert according to the rules as 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 42: KEY conversion example

2.13. Calendar Properties

2.13.1. CALADRURI

A CALADRURI property converts to an entry in the schedulingAddresses property ([Figure 43](#)). The entry value is a SchedulingAddress object whose uri property is set to the CALADRURI value.

The PREF parameter converts according to the rules as 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 43: CALADRURI conversion example

2.13.2. CALURI

A CALURI property converts to an entry in the calendars property ([Figure 44](#)). The entry value is a Calendar object whose kind property is set to calendar and uri property is set to the CALURI value.

The PREF and TYPE parameters convert according to the rules as 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 44: CALURI conversion example

2.13.3. FBURL

An FBURL property converts to an entry in the calendars property ([Figure 45](#)). The entry value is a Calendar object whose kind property is set to freeBusy and uri property is set to the FBURL value.

The PREF and TYPE parameters convert according to the rules as 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 45: FBURL conversion example

2.14. Extended Properties and Parameters

These 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. Property vCardProps

Name: vCardProps

Type: JCardProp[], where JCardProp denotes a jCard-encoded vCard property as defined in [Section 3.3](#) of [\[RFC7095\]](#).

Definition: This property is set on a JSContact object that represents a vCard. It contains properties that are set in the vCard represented by this JSContact object. Each entry in this list typically represents a vCard property for which no conversion to an IANA-registered JSContact property is defined.

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

```
item1.X-F00;X-BAR=Hello:World!
```

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

Figure 46: JSContact vCardProps example

2.15.2. Property vCardParams

Name: vCardParams

Type: String[String|String[]]

Definition: This property is set on a JSContact object that represents a vCard property. Its value **MUST** be a JSON object containing vCard property parameters, defined as array element 2 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 47: JSContact vCardParams example

2.15.3. Property vCardName

Name: vCardName

Type: String

Definition: This property is set on a JSContact object that represents a vCard property or parameter and its value contains the name of that vCard element. This allows to preserve the name of a vCard element when multiple elements convert the same JSContact object 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 OnlineService in JSContact. The vCardName property value indicates that the vCard source element was IMPP:

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

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

Figure 48: JSContact vCardName example

3. Converting JSContact to vCard

3.1. Conversion Rules

A Card 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 for each the PROP-ID parameter **MUST** be set to the Id of the converted value (see [Section 2.3.16](#)).

*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 full property is set, then implementations **MUST** use its value for the vCard FN property.

-If the Name full property is not set, then implementations **SHOULD** derive the full name from the Name components 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, they **MUST** set the FN property 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 object types 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: This property represents a JSContact property in vCard.

Value type: TEXT, also see Format Definition for value restrictions.

Conformance: The property 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. 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 [[I-D.ietf-calexext-jscontact](#)]. 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 MANDATORY 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 49: 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 50: 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 51: Nested vendor-specific property example with path relative to Card

3.3. New vCard Parameter

3.3.1. JSCOMPS

Parameter name: JSCOMPS

Purpose: This parameter 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 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 two 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 by default is zero, 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. [Figure 52](#) and [Figure 53](#) 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 COMMA (U+002C), followed by zero or more [Section 3.3](#) of param-value [[RFC6350](#)] characters, where the COMMA (U+002c) and SEMICOLON (U+003B) character **MUST** be escaped according to the rules defined in [Section 3.4](#) of [[RFC6350](#)]. [Figure 54](#) 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 [[I-D.ietf-calext-vcard-jscontact-extensions](#)].

-All other values count once each.

```
Format definition: = "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 special characters according to
```

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 52: Example for positional entries

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 53: Example for 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 54: Example for separator entries

3.3.2. JSPTR

Parameter name: JSPTR

Purpose: This parameter contains a JSON pointer [[RFC6901](#)] that relates the vCard [JSPROP](#) ([Section 3.2.1](#)) property to a JSContact 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-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 5](#) of [\[I-D.ietf-calext-jscontact\]](#) and [Section 9](#) of [\[RFC6350\]](#).

5. Acknowledgements

The definition and examples of the [PHONETIC](#) ([Section 2.3.13](#)) and [SCRIPT](#) ([Section 2.3.17](#)) parameters are based on the initial version of [\[I-D.calconnect-vobject-i18n\]](#).

6. IANA Considerations

6.1. New vCard Properties

IANA is requested to add the following entries to the "vCard Properties" registry, defined in [Section 10.3.1.](#) of [\[RFC6350\]](#).

Namespace	Property	Reference
	JSPROP	This document, Section 3.2.1

Table 4: New vCard Properties

6.2. New vCard Parameters

IANA is requested to add the following entries to the "vCard Parameters" registry, defined in [Section 10.3.2.](#) of [\[RFC6350\]](#).

Namespace	Parameter	Reference
	JSPTR	This document, Section 3.3.2

Table 5: New vCard Parameters

6.3. New JSContact Properties

IANA is requested to add the following entries to the "JSContact Properties" registry. The Since Version for all properties is 1.0. The Until Version for all properties is not set. All RFC section references are for this document. The change controller for all these properties is IETF.

Property Name	Property Type	Property Context	Reference or Description	Intended Usage
vCardName	String	Any JSContact object	Section 2.15.3	common
vCardParams			Section 2.15.2	common

Property Name	Property Type	Property Context	Reference or Description	Intended Usage
	String[String String[]]	Any JSContact object		
vCardProps	JCardProp[]	Card	Section 2.15.1	common

Table 6: Initial Contents of the "JSContact Properties" Registry

6.4. New JSContact Types

IANA is requested to add the following entries to the "JSContact Types" registry. The Since Version for all properties is 1.0. The Until Version for all properties is not set. All RFC section references are for this document. The change controller for all these properties is IETF.

Type Name	Reference or Description	Intended Usage
JCardProp	Section 2.15.1	common

Table 7: Additional Contents of the "JSContact Types" Registry

7. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol as defined in this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [[RFC7942](#)]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

7.1. CNR

- *Responsible Organization: National Research Council (CNR) of Italy
- *Location: <https://github.com/consigliunazionaleledellericerche/jscontact-tools>
- *Description: This implementation includes tools for JSContact creation, validation, serialization/deserialization, and conversion from vCard, xCard and jCard.
- *Level of Maturity: This is an "alpha" test implementation.
- *Coverage: This implementation includes all features described in this specification.
- *Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

8. References

8.1. Normative References

- [I-D.ietf-calext-jscontact] Stepanek, R. and M. Loffredo, "JSContact: A JSON representation of contact data", Work in Progress, Internet-Draft, draft-ietf-calext-jscontact-15, 18 September 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-calext-jscontact-15>>.
- [I-D.ietf-calext-vcard-jscontact-extensions] Stepanek, R. and M. Loffredo, "vCard Format Extension for JSContact", Work in Progress, Internet-Draft, draft-ietf-calext-vcard-jscontact-extensions-10, 31 August 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-calext-vcard-jscontact-extensions-10>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [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>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

- [CLDRPersonName] Davis, M., Edberg, P., Gillam, R., Kolisnychenko, A., McKenna, M., and others, "Technical Standard #35: Unicode Locale Data Markup Language (LDML) Part 8: Person Names, Version 43.1", July 2023, <<https://www.unicode.org/reports/tr35/tr35-personNames.html>>.
- [I-D.calconnect-vobject-i18n] Tse, R. H., Tam, P., and M. Douglass, "vObject Internationalization", Work in Progress, Internet-Draft, draft-calconnect-vobject-i18n-00, 7 June 2018, <<https://datatracker.ietf.org/doc/html/draft-calconnect-vobject-i18n-00>>.
- [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>>.

Appendix A. Reverse Rules of Converting vCard to JSContact Card

[Table 8](#) lists the relevant document sections for each JSContact object type and property.

JSContact Type	Property Name	Relevant Section(s)
Address	@type	not applicable
Address	components	Section 2.6.1 , Section 3.3.1
Address	contexts	Section 2.3.20
Address	coordinates	Section 2.3.7 , Section 2.8.1
Address	country	Section 2.6.1
Address	countryCode	Section 2.6.1
Address	defaultSeparator	Section 2.6.1 , Section 3.3.1
Address	full	Section 2.6.1
Address	isOrdered	Section 2.6.1 , Section 3.3.1
Address	locality	Section 2.6.1
Address	phoneticScript	Section 2.3.13 , Section 2.3.17
Address	phoneticSystem	Section 2.3.13
Address	postcode	Section 2.6.1
Address	pref	Section 2.3.15
Address	region	Section 2.6.1
Address	timeZone	Section 2.3.21 , Section 2.8.2
AddressComponent	phonetic	Section 2.3.13
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 2.3.20
Calendar	kind	Section 2.13.1 , Section 2.13.3
Calendar	label	Section 2.11.11
Calendar	mediaType	Section 2.3.12
Calendar	pref	Section 2.3.15
Calendar	uri	Section 2.13.1 , Section 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	Section 2.13.1 , Section 2.13.3
Card	created	Section 2.11.3
Card	directories	Section 2.4.3 , Section 2.10.4
Card	emails	Section 2.7.1
Card	keywords	Section 2.11.1

JSContact Type	Property Name	Relevant Section(s)
Card	kind	Section 2.4.2
Card	language	Section 2.7.4
Card	links	Section 2.9.1 , Section 2.11.9
Card	localizations	Section 2.3.10
Card	media	Section 2.5.7 , Section 2.9.2 , Section 2.11.7
Card	members	Section 2.9.3
Card	name	Section 2.5.5
Card	nicknames	Section 2.5.5
Card	notes	Section 2.11.4
Card	onlineServices	Section 2.7.2
Card	organizations	Section 2.9.4
Card	personalInfo	Section 2.10.1 , Section 2.10.2 , Section 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
CryptoKey	contexts	Section 2.3.20
CryptoKey	kind	not applicable
CryptoKey	label	Section 2.11.11
CryptoKey	mediaType	Section 2.3.12
CryptoKey	pref	Section 2.3.15
CryptoKey	uri	Section 2.12.1
Directory	@type	not applicable
Directory	contexts	Section 2.3.20
Directory	kind	Section 2.4.3 , Section 2.10.4
Directory	label	Section 2.11.11
Directory	listAs	Section 2.3.9
Directory	mediaType	Section 2.3.12
Directory	pref	Section 2.3.15
Directory	uri	Section 2.4.3 , Section 2.10.4
EmailAddress	@type	not applicable
EmailAddress	address	Section 2.7.1
EmailAddress	contexts	Section 2.3.20
EmailAddress	label	Section 2.11.11
EmailAddress	pref	Section 2.3.15
LanguagePref	@type	not applicable
LanguagePref	contexts	Section 2.3.20
LanguagePref	pref	Section 2.3.15

JSContact Type	Property Name	Relevant Section(s)
Link	@type	not applicable
Link	contexts	Section 2.3.20
Link	kind	Section 2.9.1 , Section 2.11.9
Link	label	Section 2.11.11
Link	mediaType	Section 2.3.12
Link	pref	Section 2.3.15
Link	uri	Section 2.9.1 , Section 2.11.9 >
Media	@type	not applicable
Media	contexts	Section 2.3.20
Media	kind	Section 2.5.7 , Section 2.9.2 , Section 2.11.7
Media	label	Section 2.11.11
Media	mediaType	Section 2.3.12
Media	pref	Section 2.3.15
Media	uri	Section 2.5.7 , Section 2.9.2 , Section 2.11.7
Name	@type	not applicable
Name	components	Section 2.5.5 , Section 3.3.1
Name	defaultSeparator	Section 2.5.5 , Section 3.3.1
Name	full	Section 2.5.2
Name	phoneticScript	Section 2.3.13 , Section 2.3.17
Name	phoneticSystem	Section 2.3.13
Name	isOrdered	Section 2.5.5 , Section 3.3.1
Name	sortAs	Section 2.3.19
NameComponent	@type	not applicable
NameComponent	kind	Section 2.5.5
NameComponent	phonetic	Section 2.3.13
NameComponent	value	Section 2.5.5
Nickname	@type	not applicable
Nickname	contexts	Section 2.3.20
Nickname	name	Section 2.5.5
Nickname	pref	Section 2.3.15
Note	@type	not applicable
Note	author	Section 2.3.2 , Section 2.3.3
Note	created	Section 2.3.5
Note	note	Section 2.11.4
OnlineService	@type	not applicable
OnlineService	contexts	Section 2.3.20
OnlineService	kind	Section 2.7.2 , Section 2.7.5
OnlineService	label	Section 2.11.11
OnlineService	pref	Section 2.3.15
OnlineService	service	Section 2.3.18
OnlineService	uri	Section 2.7.2 , Section 2.7.5
OnlineService	user	Section 2.3.22
OrgUnit	@type	not applicable
OrgUnit	name	Section 2.9.4
OrgUnit	sortAs	Section 2.3.19

JSContact Type	Property Name	Relevant Section(s)
Organization	@type	not applicable
Organization	contexts	Section 2.3.20
Organization	name	Section 2.9.4
Organization	sortAs	Section 2.3.19
Organization	units	Section 2.9.4
PartialDate	@type	not applicable
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	Section 2.10.1 , Section 2.10.2 , Section 2.10.3
PersonalInfo	listAs	Section 2.3.9
PersonalInfo	level	Section 2.3.11
PersonalInfo	value	Section 2.10.1 , Section 2.10.2 , Section 2.10.3
Phone	@type	not applicable
Phone	contexts	Section 2.3.20
Phone	features	Section 2.7.6
Phone	label	Section 2.11.11
Phone	number	Section 2.7.6
Phone	pref	Section 2.3.15
Pronouns	@type	not applicable
Pronouns	contexts	Section 2.3.20
Pronouns	pref	Section 2.3.15
Pronouns	pronouns	Section 2.5.4
Relation	@type	not applicable
Relation	relation	Section 2.9.5
Resource	@type	not applicable
SchedulingAddress	@type	not applicable
SchedulingAddress	contexts	Section 2.3.20
SchedulingAddress	label	Section 2.11.11
SchedulingAddress	pref	Section 2.3.15
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

JSContact Type	Property Name	Relevant Section(s)
Title	kind	Section 2.9.6
Title	name	Section 2.9.6
Title	organizationId	Section 2.9.6

Table 8: Conversion rules by JSContact property

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>