calext
Internet-Draft
Obsoletes: 9555 (if approved)
Updates: 6350 (if approved)
Intended status: Standards Track
Expires: 15 October 2026

M. Loffredo
IIT-CNR/Registro.it
R. Stepanek
Fastmail

13 April 2026

JSContact: Converting from and to vCard
draft-ietf-calext-rfc9555bis-00

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. This document obsoletes RFC 9555 and updates its definitions for JSContact version "2.0".

Note

This note is to be removed before publishing as an RFC.

Differences from RFC 9555 are documented in Appendix A.

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 15 October 2026.

Copyright Notice

Table of Contents

1. Introduction

1.1. 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.

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].

1.2. Scope and Goals

This document outlines how to convert between the vCard [RFC6350] and JSContact [RFC9553] data formats. It describes which elements are common to both, but also highlights where the two formats differ. For each common element, it defines a conversion rule and includes an example of how to convert that element. All vCard and JSContact elements currently registered at IANA are in scope, but not all of these elements are common to both formats.

For elements that have no counterpart in the other format, it is the goal of this document to define how to preserve them during conversion, but in general it is not the goal to achieve this by defining new standard elements. Instead, this document updates [RFC6350] and [RFC9553] by defining vCard and JSContact elements to preserve elements that have no equivalent in the other format. These conversion-specific properties are defined in Section 5.2.1 for vCard, and Section 4.1.1 for JSContact.

This document obsoletes [RFC9555]. For contents that remain unchanged, it includes all verified errata for the obsoleted document. The conversion rules are updated for JSContact version "2.0", defined in [I-D.ietf-calext-jscontact-uid].

Section 4 replaces the JSContact "vCardProps", "vCardParams" and "vCardParam" properties originally defined in [RFC9555] with the new "vCard" property. The new property allows for edge cases where the former elements were insufficient. For vCard, Section 5 replaces the PROP-ID [RFC9554] (Section 4.7) parameter with the new JSID parameter. Both changes align the conversion-specific elements of vCard and JSContact with those defined for iCalendar and JSCalendar in [I-D.ietf-calext-jscalendar-icalendar], allowing for code reuse for implementations that support these formats.

1.3. How to Read the Examples

Later sections contain examples that illustrate how to convert between the vCard and JSContact data formats. The notation of these examples is such that their main points should be clear to the reader, but their contents can also be parsed for automated testing. The following sections define the notation for such examples.

1.3.1. vCard Examples

A vCard example contains either an extract or a complete representation of vCard data. The following rules apply:

* An example that only contains vCard properties implicitly represents a vCard (e.g. the BEGIN:VCARD and END:VCARD lines are implicit).

* The vCard implicitly contains all mandatory properties with some value. If they are not included in the example, then their actual value is irrelevant for the example. The default value for the VERSION property is "4.0".

Figure 1 contains three examples, all of which represent the same vCard data. In the first example, the BEGIN and END lines and the mandatory properties are implicit. In the second example, only the BEGIN and END lines of the vCard are implicit. The third example depicts a complete vCard, nothing is implicit.

```
EMAIL:foo@example.com
VERSION:4.0
FN:Foo
EMAIL:foo@example.com
BEGIN:VCARD
VERSION:4.0
FN:Foo
EMAIL:foo@example.com
END:VCARD
```

Figure 1: Examples for implicit and explicit vCard notation

A line containing just the value ... stands for any other properties that might be present in the vCard but are irrelevant for this example. Figure 2 illustrates this as an alternative representation for the examples of Figure 1.

```
BEGIN:VCARD
...
EMAIL:foo@example.com
```

Figure 2: Example for additional properties represented by ...

A line starting with a single space represents the continuation of a folded content line (Section 3.2 of [RFC6350]). Figure 3 illustrates this.

```
FN:Joannes Chrysostomus Wolfg
 angus Theophilus Mozart
```

Figure 3: Example for a folded content line

1.3.2. JSContact Examples

A JSContact example always represents a Card object, even if the example only depicts other JSContact objects and properties.

JSContact objects are depicted either explicitly or implicitly. An explicit JSContact object starts and ends with braces. An implicit JSContact object omits braces, it only consists of JSON name/value pairs, separated by comma.

An implicit JSContact object is assumed to be of type Card, unless it contains the @type property with a different value. It is assumed to contain all mandatory properties with some value; if they are not depicted, their actual value is irrelevant for the main point of the example. The default version of the Card is "2.0".

Figure 4 illustrates this with multiple examples, all of which represent the same JSContact data. The first example contains an implicit JSContact object of type Card. The second example contains an explicit Name object but the Card object containing it is omitted. The third example contains the full Card object, nothing is omitted.

```
"name": {
  "full": "John Doe"
}
```

```
"@type": "Name",
"full": "John Doe"
```

```
{
  "@type": "Card",
  "version": "2.0",
  "name": {
    "full": "John Doe"
  }
}
```

Figure 4: Examples for implicit and explicit JSContact object notation

A property with name "..." and value "" stands for additional properties that might be present in a JSContact object but are irrelevant for this example, including mandatory properties. Figure 5 illustrates this as an alternative representation for the examples of Figure 4.

```
"name": {
  "full": "John Doe",
  "...": ""
}
```

Figure 5: Example for additional properties represented by ...

2. Converting vCard to JSContact

This section contains the conversion rules from the vCard to the JSContact Card.

2.1. General Rules

2.1.1. The Card uid Property

The UID property converts to the "uid" property of the Card object, see Section 2.3.46. In contrast to JSContact version "2.0", the Card "uid" property was mandatory in version "1.0".

Implementations that convert a vCard without a UID property [RFC6350] (Section 6.7.6) to a Card of version "2.0" or higher MUST NOT generate a unique identifier as value for the uid property [RFC9553] (Section 2.1.9).

When converting a vCard without UID property to JSContact version "1.0", implementations MUST generate a value for the uid property. Generating unique identifiers is implementation-specific. An implementation SHOULD generate the same value when generating the same Card multiple times. Section 2 of [I-D.ietf-calext-jscontact-uid] describes why this is problematic. Consequently, implementations SHOULD NOT convert to version "1.0" Card objects.

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 JSID parameter (see Section 5.1.2), 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.

2.1.3. Converting Unknown Elements

vCards may contain elements for which no IANA-registered JSContact property is defined, such as extended properties and parameters (see Section 6.10 of [RFC6350]). Section 4.1.1 defines the "vCard" property to represent arbitrary vCard elements in JSContact.

2.2. 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 4.1.1.

2.2.1. 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 2.3.1. To preserve the verbatim value of the ALTID parameter, set the JSContact properties defined in Section 4.

2.2.2. 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]).

2.2.3. 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.

2.2.4. 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]).

2.2.5. 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]).

2.2.6. 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]).

2.2.7. 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.

2.2.8. 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]). Also see the conversion of the GEO property in Section 2.3.18.

2.2.9. 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 MAY preserve the original group name in the "parameters" property of the VCardProperty object (Section 4.1.1).

```
item1.TEL;VALUE=uri:tel:+1-555-555-5555
item1.X-ABLabel:Test
"phones": {
  "p1": {
    "number": "tel:+1-555-555-5555",
    "label": "test"
  }
},
"vCard": {
  "convertedProperties": {
    "phones/p1/number": {
      "name": "tel",
      "parameters": {
        "group": "item1"
      }
    },
    "phones/p1/label": {
      "name": "x-ablabel",
      "parameters": {
        "group": "item1"
      }
    }
  }
}
```

Figure 6: Example of How to Preserve the Group Name for a Known Property

```
item2.X-FOO:bar
"vCard": {
  "properties": [
    ["x-foo", {
      "group": "item2"
    }, "unknown", "bar"]
  ]
]
```

Figure 7: Example of How to Preserve the Group Name for an Unknown Property

2.2.10. 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.

2.2.11. 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. 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. Implementations that convert to a JSContact profile [I-D.ietf-calext-jscontact-profiles] MUST adhere to the profile-specific restrictions, if any.

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 8 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 8: 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 9 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 9: LANGUAGE Conversion Example: Property without Language

2.2.12. 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 of [RFC9553]).

2.2.13. 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.

2.2.14. MEDIATYPE

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

2.2.15. 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 2.2.15).

The related N or ADR property is defined by the vCard ALTID parameter. The conversion rules for the N (Section 2.3.29) and ADR (Section 2.3.1) 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 2.2.11.

```
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 10: PHONETIC Conversion Example

2.2.16. PID

The PID parameter (Section 5.5 of [RFC6350]) does not convert to a standard JSContact element. Implementations MAY convert it to the "parameters" property of the VCardProperty object (Section 4.1.1).

2.2.17. PREF

The PREF parameter (Section 5.3 of [RFC6350]) converts to the pref property of that JSContact object, to which the vCard property having this parameter converts.

2.2.18. PROP-ID

The PROP-ID parameter (Section 4.7 of [RFC9554]) is deprecated. The JSID (Section 5.1.2) parameter replaces it.

The PROP-ID parameter MUST be ignored if the JSID parameter is set on the same property. If the JSID parameter is not set, implementations MAY convert the PROP-ID parameter to the key of the JSContact object to which the vCard property converts to.

```
TEL;PROP-ID=xyz;VALUE=uri:tel:+1-555-555-5555;ext=5555
"phones": {
  "xyz": {
    "number": "tel:+1-555-555-5555;ext=5555"
  }
}
```

Figure 11: PROP-ID Conversion Example

2.2.19. 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 2.2.15.

2.2.20. 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]).

2.2.21. SORT-AS

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

2.2.22. TYPE

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

2.2.23. 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 of [RFC9553]). Also see the conversion of the TZ property in Section 2.3.45.

2.2.24. USERNAME

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

2.2.25. 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 Section 4.1.1.

2.3. vCard Properties

2.3.1. ADR

The ADR property (Section 6.3.1 of [RFC6350]) converts to an Address object (Section 2.5.1 of [RFC9553]) in the Card object's addresses property.

Each component in the structured value of the ADR property converts to an AddressComponent in the Address components property. The component value converts to the AddressComponent value property, the component position determines the value of the AddressComponent kind property as follows:

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	Do not convert if the ADR property value includes extended ADR components defined in [RFC9554].
street address	name	Do not convert if the ADR property value includes extended ADR components defined in [RFC9554].
locality	locality	

region	region	
postal code	postcode	
country	country	
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 1: ADR Components Conversion

If the JSCOMPS (Section 5.1.1) parameter is set and valid, 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 ADR property converts to the same Address object as the GEO property (Section 2.3.18) and TZ property (Section 2.3.45) if either they are members of the same vCard property group, or if they are part of no group.

The PREF and TYPE parameters convert according to the rules defined in Section 2.2. 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 2.2. 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 12: ADR Conversion Example

See Section 5.1.1 for examples of using the JSCOMPS parameter for vCard-structured property values.

2.3.2. ANNIVERSARY

The ANNIVERSARY property (Section 6.2.6 of [RFC6350]) converts to an Anniversary object in the Card object's anniversaries property (Figure 13). The Anniversary kind is "wedding".

See Section 2.3.3 how to convert property values of type TIMESTAMP and DATE.

```
ANNIVERSARY:19860201
"anniversaries": {
  "ANNIVERSARY-1" : {
    "kind": "wedding",
    "date": {
      "year": 1986,
      "month": 2,
      "day": 1
    }
  }
}
```

Figure 13: ANNIVERSARY Conversion Example

2.3.3. BDAY

The BDAY property (Section 6.2.5 of [RFC6350]) converts to an Anniversary object in the Card object's anniversaries property (Figure 13). Its value converts to the Anniversary date property. The Anniversary kind is "birth".

If the property value type is DATE [RFC6350] (Section 4.3.5) and defines at least a year or a month and day, then the value converts to a PartialDate object. If the property value type is TIMESTAMP [RFC6350] (Section 4.3.5) then the value converts to the utc property of the Timestamp object. Otherwise it does not convert to a standard JSContact element.

A BDAY property and the BIRTHPLACE property (see Section 2.3.4) convert to the same Anniversary object if their ALTID parameter values match or are not set on either property.

```
BDAY:19531015T231000Z
BIRTHPLACE:
 123 Main Street\nAny Town, CA 91921-1234\nU.S.A.
```

```
"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."
    }
  }
}
```

Figure 14: BDAY Conversion Example

2.3.4. BIRTHPLACE

The BIRTHPLACE property (Section 2.1 of [RFC6474]) converts to an Anniversary object in the Card object's anniversaries property (Figure 13). Its value converts to an Address object in the Anniversary place property. The Anniversary kind is "birth".

A property value of type TEXT converts to the Address object's full property. A property value of type URI using the "geo:" URI scheme converts to the Address object's coordinates property. A property having any other value type does not convert to an Anniversary object.

See Section 2.3.3 for an example and how to convert a BIRTHPLACE property together with a BDAY property.

2.3.5. 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 15). 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 2.2.

```
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 15: CALADRURI Conversion Example

2.3.6. 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 16). 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 2.2.

```
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 16: CALURI Conversion Example

2.3.7. CATEGORIES

The CATEGORIES property (Section 6.7.1 of [RFC6350]) converts to a set of entries of the Card object's keywords property (Figure 17). 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 17: CATEGORIES Conversion Example

2.3.8. CLIENTPIDMAP

The CLIENTPIDMAP property (Section 6.7.7 of [RFC6350]) does not convert to a standard JSContact element. Implementations MAY convert it to the "properties" property of the VCard object (Section 4.1.1).

2.3.9. 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 18). 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 2.2.

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

Figure 18: CONTACT-URI Conversion Example

2.3.10. CREATED

The CREATED property (Section 3.1 of [RFC9554]) converts to the Card object's created property (Figure 19).

```
CREATED:19940930T143510Z
```

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

Figure 19: CREATED Conversion Example

2.3.11. 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 20). 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 2.2.

```
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 20: EMAIL Conversion Example

2.3.12. DEATHDATE

The DEATHDATE property (Section 2.3 of [RFC6474]) converts to an Anniversary object in the Card object's anniversaries property (Figure 13). Its value converts to the Anniversary date property. The Anniversary kind is "death".

See Section 2.3.3 how to convert property values of type TIMESTAMP and DATE.

A DEATHDATE property and the DEATHPLACE property (see Section 2.3.13) convert to the same Anniversary object if their ALTID parameter values match or are not set on either property.

```
DEATHDATE:19960415
DEATHPLACE:
 5 Court Street\nNew England, ND 58647\nU.S.A.
```

```
"anniversaries": {
  "ANNIVERSARY-2" : {
    "kind": "death",
    "date": {
      "year": 1996,
      "month": 4,
      "year": 15
    },
    "place": {
      "full": "5 Court Street\nNew England, ND 58647\nU.S.A."
    }
  }
}
```

Figure 21: DEATHDATE Conversion Example

2.3.13. DEATHPLACE

The DEATHPLACE property (Section 2.2 of [RFC6474]) converts to an Anniversary object in the Card object's anniversaries property (Figure 13). Its value converts to an Address object in the Anniversary place property. The Anniversary kind is "death".

A property value of type TEXT converts to the Address object's full property. A property value of type URI using the "geo:" URI scheme converts to the Address object's coordinates property. A property having any other value type does not convert to an Anniversary object.

See Section 2.3.12 for an example and how to convert a DEATHPLACE property together with a DEATHDATE property.

2.3.14. 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 22). The PersonalInfo object's kind property is set to "expertise".

The INDEX and LEVEL parameters convert according to the rules defined in Section 2.2.

```
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 22: EXPERTISE Conversion Example

2.3.15. 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 23). 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 2.2.

```
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 23: FBURL Conversion Example

2.3.16. FN

The FN property (Section 6.2.1 of [RFC6350]) converts to the Name object's full property (Figure 24). If the LANGUAGE parameter is set, then the FN property converts as outlined in Section 2.2.11. 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 do not convert to a standard JSContact element. Implementations MAY convert them to the "properties" property of the VCard object (Section 4.1.1).

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

Figure 24: FN Conversion Example

2.3.17. 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 4.1.1. 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].

2.3.18. GEO

The GEO property (Section 6.5.2 of [RFC6350]) converts to the Address object's coordinates property (Section 2.5.1 of [RFC9553]).

See Section 2.3.1 how to convert the GEO property together with the ADR and TZ property.

2.3.19. GRAMGENDER

The GRAMGENDER property (Section 3.2 of [RFC9554]) converts to the SpeakToAs object's grammaticalGender property (Figure 25).

```
GRAMGENDER:NEUTER
"speakToAs": {
  "grammaticalGender": "neuter"
}
```

Figure 25: GRAMGENDER Conversion Example

2.3.20. 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 26). The PersonalInfo object's kind property is set to "hobby".

The INDEX and LEVEL parameters convert according to the rules defined in Section 2.2.

```
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 26: HOBBY Conversion Example

2.3.21. 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 27). The "uri" property is set to the IMPP value.

Implementations MAY indicate that the OnlineService object converted from the IMPP property by adding an entry for the "uri" property of the OnlineService object in the "vCard/convertedProperties" property.

The SERVICE-TYPE, USERNAME, PREF, and TYPE parameters convert according to the rules defined in Section 2.2.

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

```
"onlineServices": {
  "OS-1": {
    "uri": "xmpp:alice@example.com",
    "pref": 1
  }
},
"vCard": {
  "convertedProperties": {
    "onlineServices/OS-1/uri": {
      "name": "impp"
    }
  }
}
```

Figure 27: IMPP Conversion Example

2.3.22. 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 28). The PersonalInfo object's kind property is set to "interest".

The INDEX and LEVEL parameters convert according to the rules defined in Section 2.2.

```
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 28: INTEREST Conversion Example

2.3.23. 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 29). 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 2.2.

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

Figure 29: KEY Conversion Example

2.3.24. KIND

The KIND property (Section 6.1.4 of [RFC6350]) converts to the kind property (Figure 30). 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 30: KIND Conversion Example

2.3.25. 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 31). 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 2.2.

```
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 31: LANG Conversion Example

2.3.26. LANGUAGE

The LANGUAGE property (Section 3.3 of [RFC9554]) converts to the Card object's language property (Figure 32).

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

Figure 32: LANGUAGE Conversion Example

2.3.27. 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 33). 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 2.2.

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

Figure 33: LOGO Conversion Example

2.3.28. MEMBER

The MEMBER property (Section 6.6.5 of [RFC6350]) converts to the Card object's members property (Figure 34). 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 34: Group Example

2.3.29. 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 structured value of the N property converts to a NameComponent in the Name components property. The component value converts to the NameComponent value property, the component position determines the value of the NameComponent kind property as follows:

N component	NameComponent kind	Remarks
Family name	surname	Ignore any value that also occurs in the Secondary surname component.
Given name	given	
Additional name	given2	
Honorific prefix	title	
Honorific suffix	credential	Ignore any value that also occurs in the Generation component.
Secondary surname	surname2	
Generation	generation	

Table 2: N Components Conversion

If the JSCOMPS (Section 5.1.1) 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 2. 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 35: N Conversion Example

See Section 5.1.1 for examples of using the JSCOMPS parameter for vCard-structured property values.

2.3.30. 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 36). The name property is set to the NICKNAME property value.

The PREF and TYPE parameters convert according to the rules defined in Section 2.2.

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

Figure 36: NICKNAME Conversion Example

2.3.31. 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 37).

The ALTID and LANGUAGE parameters convert according to the rules defined in Section 2.2.

```
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 37: NOTE Conversion Example

2.3.32. 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 38). 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 2.2.

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 2.2.

```
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 38: ORG Conversion Example

2.3.33. 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 39). 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 2.2.

```
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 39: ORG-DIRECTORY Conversion Example

2.3.34. 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 40). 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 2.2.

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

Figure 40: PHOTO Conversion Example

2.3.35. PRODID

The PRODID property (Section 6.7.3 of [RFC6350]) converts to the Card object’s prodId property (Figure 41).

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

Figure 41: PRODID Conversion Example

2.3.36. PRONOUNS

The PRONOUNS property (Section 3.4 of [RFC9554]) converts to the SpeakToAs object’s pronouns property (Figure 42).

```
PRONOUNS;PREF=2:they/them
PRONOUNS;PREF=1:xe/xir
"speakToAs": {
  "pronouns": {
    "PRONOUNS-1": {
      "pronouns": "they/them",
      "pref": 2
    },
    "PRONOUNS-2": {
      "pronouns": "xe/xir",
      "pref": 1
    }
  }
}
```

Figure 42: PRONOUNS Conversion Example

2.3.37. RELATED

The RELATED property (Section 6.6.6 of [RFC6350]) converts to the Card object's relatedTo property (Figure 43). 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 4.1.1.

```
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 43: RELATED Conversion Example

2.3.38. REV

The REV property (Section 6.7.4 of [RFC6350]) converts to the Card object's updated property (Figure 44).

```
REV:19951031T222710Z
"updated": "1995-10-31T22:27:10Z"
```

Figure 44: REV Conversion Example

2.3.39. ROLE

The ROLE property (Section 6.6.2 of [RFC6350]) converts to a Title object (Section 2.2.5 of [RFC9553]) in the Card object's titles property (Figure 45). The Title object kind is "role". The ROLE property value converts to the Title object name property.

If the vCard contains an ORG property and either the ROLE and ORG property are not part of a vCard property group, or they are both in the same group, then the Title organizationId property is set to the identifier to which the ORG property converts in the Card object's organizations property.

The ALTID and LANGUAGE parameters convert according to the rules defined in Section 2.2.

```
group1.ROLE:Project Leader
group1.ORG:ABC, Inc.
"titles": {
  "ROLE-1": {
    "kind": "role",
    "name": "Project Leader",
    "organizationId": "ORG-1"
  }
},
"organizations": {
  "ORG-1": {
    "name": "ABC, Inc."
  }
}
```

Figure 45: ROLE Conversion Example

2.3.40. 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 46). If the SOCIALPROFILE property value is of type URI, then the "uri" property is set, otherwise, the "user" property is set.

Implementations MAY indicate that the OnlineService object converted from the SOCIALPROFILE property by adding an entry for the "uri" or "user" property of the OnlineService object in the "vCard/convertedProperties" property.

The SERVICE-TYPE, USERNAME, PREF, and TYPE parameters convert according to the rules defined in Section 2.2.

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

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

Figure 46: SOCIALPROFILE Conversion Example

2.3.41. 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 47). 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 2.2.

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

Figure 47: SOUND Conversion Example

2.3.42. 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 48). 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 2.2.

```
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 48: SOURCE Conversion Example

2.3.43. 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 49).

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
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 2.2.

```
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 49: TEL Conversion Example

2.3.44. TITLE

The TITLE property (Section 6.6.1 of [RFC6350]) converts to a Title object (Section 2.2.5 of [RFC9553]) in the Card object's titles property (Figure 50). The TITLE property value converts to the Title object name property.

The same rules as in Section 2.3.39 apply for setting the organizationId property of the Title object.

The ALTID and LANGUAGE parameters convert according to the rules defined in Section 2.2.

```
TITLE:Research Scientist
"titles": {
  "TITLE-1": {
    "kind": "title",
    "name": "Research Scientist"
  }
}
```

Figure 50: TITLE Conversion Example

2.3.45. TZ

The TZ property (Section 6.5.1 of [RFC6350]) converts to 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".

See Section 2.3.1 how to convert the TZ property together with the ADR and GEO property.

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 4.1.1.

2.3.46. UID

The UID property (Section 6.7.6 of [RFC6350]) converts to the Card object's uid property (Figure 51).

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

Figure 51: UID Conversion Example

2.3.47. 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 52). 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 2.2.

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

Figure 52: URL Conversion Example

2.3.48. VERSION

The VERSION property (Section 6.7.9 of [RFC6350]) does not convert to a standard JSContact element. Implementations MAY convert it to the "properties" property of the VCard object (Section 4.1.1).

2.3.49. 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 2.2.9.

```
item1.TEL;VALUE=uri:tel:+1-555-555-5555
item1.X-ABLabel:foo
"phones": {
  "p1": {
    "number": "tel:+1-555-555-5555",
    "label": "foo"
  }
}
```

Figure 53: X-ABLabel Conversion Example

2.3.50. XML

The XML property (Section 6.1.5 of [RFC6350]) does not convert to a standard JSContact element. Implementations MAY convert it to the "properties" property of the VCard object (Section 4.1.1).

3. Converting JSContact to vCard

This section defines the conversion rules for each JSContact object type. Each subsection lists the properties of one object type and the relevant document sections for their conversion to vCard.

3.1. Address

The Address object converts to the ADR property. Each AddressComponent object in the components property [RFC9553] (Section 2.5.1.1) converts to a component in the structured value of the ADR property.

AddressComponent kind	ADR component	Remarks
apartment	apartment	Section 2.3.1, and remarks below
block	block	Section 2.3.1, and remarks below
building	building	Section 2.3.1, and remarks below
country	country	Section 2.3.1
direction	direction	Section 2.3.1, and remarks below
district	district	Section 2.3.1, and remarks below
floor	floor	Section 2.3.1, and remarks below
landmark	landmark	Section 2.3.1, and remarks below
locality	locality	Section 2.3.1
name	street address	Section 2.3.1, and remarks below
number	street number	Section 2.3.1, and remarks below
postcode	postal code	Section 2.3.1
postOfficeBox	post office box	Section 2.3.1
region	region	Section 2.3.1

room	room	Section 2.3.1, and remarks below
subdistrict	subdistrict	Section 2.3.1, and remarks below

Table 4: AddressComponent conversion

Remarks:

* In addition to setting the "block", "direction", "district", "landmark", "name", "number", and "subdistrict" components, implementations MAY combine these values to set the "street address" component in the ADR property value.

* In addition to setting the "apartment", "building", "floor" and "room" components, implementations MAY combine these values to set the "extended address" component in the ADR property value.

The other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.5.1.1	TYPE	Section 2.2.22
coordinates	[RFC9553], Section 2.5.1.1	GEO	Sections 2.2.8 and 2.3.18, and remarks below
countryCode	[RFC9553], Section 2.5.1.1	CC	Section 2.2.5
defaultSeparator	[RFC9553], Section 2.5.1.1	JSCOMPS	Section 5.1.1
full	[RFC9553], Section 2.5.1.1	LABEL	Section 2.2.12
isOrdered	[RFC9553], Section 2.5.1.1	JSCOMPS	Section 5.1.1
phoneticScript	[RFC9553], Section 2.5.1.1	SCRIPT	Section 2.2.19
phoneticSystem	[RFC9553], Section 2.5.1.1	PHONETIC	Section 2.2.15
pref	[RFC9553], Section 2.5.1.1	PREF	Section 2.2.17
timeZone	[RFC9553], Section 2.5.1.1	TZ	Sections 2.2.23 and 2.3.45, and remarks below

Table 5: Properties of the Address object

Remarks:

* The "coordinates" and "timeZone" properties SHOULD convert to the GEO and TZ parameters, but implementations MAY alternatively convert them to the same-named properties e.g. to preserve original vCard data.

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.2. Anniversary

The Anniversary object converts to one of the following properties, determined by the Anniversary kind property value:

* If the kind is "wedding" and the Anniversary date property is set, then it converts to the ANNIVERSARY (Section 2.3.2) property. How to convert the place property is implementation-specific.

* If the kind is "birth", then the date property value converts to the BDAY (Section 2.3.3) property and the place property converts to the BIRTHPLACE (Section 2.3.4) property.

* If the kind is "death", then the date property value converts to the DEATHDATE (Section 2.3.12) property and the place property converts to the DEATHPLACE (Section 2.3.13) property.

A date property value of type PartialDate converts to the DATE property value type. The calendarScale property converts to the CALSCALE parameter. A date property value of type Timestamp converts to the TIMESTAMP property value type by converting the "utc" property. How to convert a place property value having any Address property other than the "full" property set is implementation-specific.

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.3. Calendar

The Calendar object converts to one of the following properties, determined by the Calendar kind property value. If the kind is "calendar" then it converts to the CALURI (Section 2.3.6) property. If the kind is "freeBusy" then it converts to the FBURL (Section 2.3.15) property.

Its uri property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.4.1	TYPE	Section 2.2.22
label	[RFC9553], Section 2.4.1	X-ABLabel (Property)	Section 2.3.49
mediaType	[RFC9553], Section 2.4.1	MEDIATYPE	Section 2.2.14
pref	[RFC9553], Section 2.4.1	PREF	Section 2.2.17

Table 6: Properties of the Calendar object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.4. Card

The Card object converts to a vCard [RFC6350].

Its properties other than the localizations property convert as follows:

Name	Reference	Property (or Parameter)	See
addresses	[RFC9553], Section 2.5.1.1	ADR	Address object (Section 3.1)
anniversaries	[RFC9553], Section 2.8.1	ANNIVERSARY	Anniversary object (Section 3.2)
calendars	[RFC9553], Section 2.4.1	CALURI, FBURL	Calendar object (Section 3.3)
created	[RFC9553], Section 2.1.3	CREATED	Section 2.3.10

directories	[RFC9553], Section 2.6.2	SOURCE, ORG-DIRECTORY	Directory object (Section 3.6)
emails	[RFC9553], Section 2.3.1	EMAIL	EmailAddress object (Section 3.7)
keywords	[RFC9553], Section 2.8.2	CATEGORIES	Section 2.3.7
kind	[RFC9553], Section 2.1.4	KIND	Section 2.3.24
language	[RFC9553], Section 2.1.5	LANGUAGE	Section 2.3.26
links	[RFC9553], Section 2.6.3	CONTACT-URI, URL	Link object (Section 3.9)
media	[RFC9553], Section 2.6.4	PHOTO, LOGO, SOUND	Media object (Section 3.10)
members	[RFC9553], Section 2.1.6	MEMBER	Section 2.3.28
name	[RFC9553], Section 2.2.1.1	N	Name object (Section 3.11)
nicknames	[RFC9553], Section 2.2.2	NICKNAME	Nickname object (Section 3.12)
notes	[RFC9553], Section 2.8.3	NOTE	Note object (Section 3.13)
onlineServices	[RFC9553], Section 2.3.2	IMPP, SOCIALPROFILE	OnlineService object (Section 3.14)

organizations	[RFC9553], Section 2.2.3	ORG	Organization object (Section 3.15)
personalInfo	[RFC9553], Section 2.8.4	EXPERTISE, HOBBY, INTEREST	PersonalInfo object (Section 3.16)
phones	[RFC9553], Section 2.3.3	TEL	Phone object (Section 3.17)
preferredLanguages	[RFC9553], Section 2.3.4	LANG	Section 2.3.25
prodId	[RFC9553], Section 2.1.7	PRODID	Section 2.3.35
relatedTo	[RFC9553], Section 2.1.8	RELATED	Section 2.3.37
schedulingAddresses	[RFC9553], Section 2.4.2	CALADRURI	SchedulingAddress object (Section 3.18)
speakToAs	[RFC9553], Section 2.2.4	GRAMGENDER, PRONOUNS	SpeakToAs object (Section 3.19)
titles	[RFC9553], Section 2.2.5	TITLE	Title object (Section 3.20)
uid	[RFC9553], Section 2.1.9	UID	Section 2.3.46
updated	[RFC9553], Section 2.1.10	REV	Section 2.3.38

Table 7: Properties of the Card object

The localizations property converts by converting the properties in the PatchObject using the same rules as outlined in Table 7. The LANGUAGE parameter (Section 2.2.11) MUST be set on each resulting property with the parameter value being the language key of the localizations property. The ALTID parameter (Section 2.2.1) MUST be set and its value be equal for all properties having the same JSON pointer path in the localizations property and the non-localized Card.

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.5. CryptoKey

The CryptoKey object converts to the KEY property (Section 2.3.23).

Its uri property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.6.1	TYPE	Section 2.2.22
label	[RFC9553], Section 2.6.1	X-ABLabel (Property)	Section 2.3.49
mediaType	[RFC9553], Section 2.6.1	MEDIATYPE	Section 2.2.14
pref	[RFC9553], Section 2.6.1	PREF	Section 2.2.17

Table 8: Properties of the CryptoKey object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.6. Directory

The Directory object converts to one of the following properties, determined by the Directory kind property value. If the kind is "entry" then it converts to the SOURCE (Section 2.3.42) property. If the kind is "directory" then it converts to the ORG-DIRECTORY (Section 2.3.33) property.

Its uri property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.6.2	TYPE	Section 2.2.22
label	[RFC9553], Section 2.6.2	X-ABLabel (Property)	Section 2.3.49
listAs	[RFC9553], Section 2.6.2	INDEX	Section 2.2.10
mediaType	[RFC9553], Section 2.6.2	MEDIATYPE	Section 2.2.14
pref	[RFC9553], Section 2.6.2	PREF	Section 2.2.17

Table 9: Properties of the Directory object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.7. EmailAddress

The EmailAddress object converts to the EMAIL (Section 2.3.11) property.

Its address property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.3.1	TYPE	Section 2.2.22
label	[RFC9553], Section 2.3.1	X-ABLabel (Property)	Section 2.3.49
pref	[RFC9553], Section 2.3.1	PREF	Section 2.2.17

Table 10: Properties of the EmailAddress object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.8. LanguagePref

The LanguagePref object converts to the LANG property (Section 2.3.25).

Its language property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.3.4	TYPE	Section 2.2.22
pref	[RFC9553], Section 2.3.4	PREF	Section 2.2.17

Table 11: Properties of the LanguagePref object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.9. Link

The Link object converts to one of the following properties, determined by the Link kind property value. If the kind property is not set then it converts to the URL property. (Section 2.3.47). If the kind is "contact" then it converts to the CONTACT-URI (Section 2.3.9) property.

Its uri property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.6.3	TYPE	Section 2.2.22
label	[RFC9553], Section 2.6.3	X-ABLabel (Property)	Section 2.3.49
mediaType	[RFC9553], Section 2.6.3	MEDIATYPE	Section 2.2.14
pref	[RFC9553], Section 2.6.3	PREF	Section 2.2.17

Table 12: Properties of the Link object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.10. Media

The Media object converts to one of the following properties, determined by the Media kind property value. If the kind is "photo" then it converts to the PHOTO property. (Section 2.3.34). If the kind is "logo" then it converts to the LOGO property (Section 2.3.27). If the kind is "sound" then it converts to the "SOUND property (Section 2.3.41).

Its uri property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.6.4	TYPE	Section 2.2.22
label	[RFC9553], Section 2.6.4	X-ABLabel (Property)	Section 2.3.49
mediaType	[RFC9553], Section 2.6.4	MEDIATYPE	Section 2.2.14
pref	[RFC9553], Section 2.6.4	PREF	Section 2.2.17

Table 13: Properties of the Media object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.11. Name

The Name object converts to the N property (Section 2.3.29). Each NameComponent object in the components property [RFC9553] (Section 2.2.1.1) converts to a component in the structured value of the N property.

NameComponent kind	N component	Remarks
credential	Honorific suffix	Section 2.3.29
generation	Generation	Section 2.3.29, and remarks below
given	Given name	Section 2.3.29
given2	Additional name	Section 2.3.29
surname	Family name	Section 2.3.29
surname2	Secondary surname	Section 2.3.29, and remarks below
title	Honorific prefix	Section 2.3.29

Table 14: NameComponent conversion

Remarks:

* In addition to converting the "generation" kind to the "Generation" component, implementations MAY also append the value to the "Honorific Suffixes" component of the N property value.

* In addition to converting the "surname2" kind to the "Secondary surname" component, implementations MAY also append the value to the "Surname" component of the N property value.

The full property converts to the FN property (Section 2.3.16).

The full property is optional, but the FN property is defined to be mandatory in [RFC6350]. How to set the FN property if the full property is not set is implementation-specific. Implementations MAY derive the full name from the NameComponents objects. If the isOrdered property value is "true", then this can be done by concatenating the name component values. Alternatively, an implementation can choose any other heuristic to generate the full name from its components such as [CLDRPersonName]. If the FN property value is derived, implementations SHOULD set the DERIVED parameter [RFC9554] (Section 4.4).

The other properties convert as follows:

Name	Reference	Parameter (or other)	See
defaultSeparator	[RFC9553], Section 2.2.1.1	JSCOMPS	Section 5.1.1
phoneticScript	[RFC9553], Section 2.2.1.1	SCRIPT	Section 2.2.19
phoneticSystem	[RFC9553], Section 2.2.1.1	PHONETIC	Section 2.2.15
isOrdered	[RFC9553], Section 2.2.1.1	JSCOMPS	Section 5.1.1
sortAs	[RFC9553], Section 2.2.1.1	SORT-AS	Section 2.2.21

Table 15: Properties of the Name object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.12. Nickname

The Nickname object converts to NICKNAME property (Section 2.3.30).

Its name property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.2.2	TYPE	Section 2.2.22
pref	[RFC9553], Section 2.2.2	PREF	Section 2.2.17

Table 16: Properties of the Nickname object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.13. Note

The Note object converts to the NOTE property (Section 2.3.31).

Its note property converts to the property value.

The Author object set as the value of its author property converts as follows:

Name	Reference	Parameter (or other)	See
name	[RFC9553], Section 2.8.3	AUTHOR-NAME	Section 2.2.3
uri	[RFC9553], Section 2.8.3	AUTHOR	Section 2.2.2

Table 17: Properties of the Author object

Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
created	[RFC9553], Section 2.8.3	CREATED	Section 2.2.6

Table 18: Properties of the Note object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.14. OnlineService

The OnlineService object converts to the IMPP property (Section 2.3.21) or SOCIALPROFILE property (Section 2.3.40). Which property to convert to is implementation-specific. Implementations SHOULD preserve the property name indicated in the vCard property (Section 4.1.1).

If its uri property is set, but neither the user or service properties are set, then implementations SHOULD choose IMPP if the uri property value is a URI with scheme "xmpp".

It other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.3.2	TYPE	Section 2.2.22
label	[RFC9553], Section 2.3.2	X-ABLabel (Property)	Section 2.3.49
pref	[RFC9553], Section 2.3.2	PREF	Section 2.2.17
service	[RFC9553], Section 2.3.2	SERVICE-TYPE	Section 2.2.20
user	[RFC9553], Section 2.3.2	USERNAME	Section 2.2.24

Table 19: Properties of the OnlineService object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.15. Organization

The Organization object converts to the ORG property. (Section 2.3.32)

Its name property converts the first component in the structured property value. The name property of any OrgUnit object in its units property converts to additional components in the structured property value, in order to appearance.

Its sortAs property converts the first component in the structured value of the SORT-AS parameter (Section 2.2.21). The sortAs property of any OrgUnit object in its units property converts to additional components in the structured parameter value, in order to appearance.

Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.2.3	TYPE	Section 2.2.22

Table 20: Properties of the Organization object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.16. PersonalInfo

The PersonalInfo object converts to one of the following properties, determined by the PersonalInfo kind property value. If the kind is "expertise" then it converts to the EXPERTISE property. (Section 2.3.14). If the kind is "hobby" then it converts to the HOBBY property (Section 2.3.20). If the kind is "interest" then it converts to the "INTEREST property (Section 2.3.22).

Its value property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
listAs	[RFC9553], Section 2.8.4	INDEX	Section 2.2.10
level	[RFC9553], Section 2.8.4	LEVEL	Section 2.2.13

Table 21: Properties of the PersonalInfo object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.17. Phone

The Phone object converts to the TEL property (Section 2.3.43)

Its number property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.3.3	TYPE	Section 2.2.22
features	[RFC9553], Section 2.3.3	TYPE	Section 2.3.43
label	[RFC9553], Section 2.3.3	X-ABLabel (Property)	Section 2.3.49
pref	[RFC9553], Section 2.3.3	PREF	Section 2.2.17

Table 22: Properties of the Phone object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.18. SchedulingAddress

The SchedulingAddress object converts to the CALADRURI property (Section 2.3.5).

Its uri property converts to the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.4.2	TYPE	Section 2.2.22
label	[RFC9553], Section 2.4.2	X-ABLabel (Property)	Section 2.3.49
pref	[RFC9553], Section 2.4.2	PREF	Section 2.2.17

Table 23: Properties of the SchedulingAddress object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.19. SpeakToAs

The SpeakToAs object converts to the GRAMGENDER property (Section 2.3.19) and PRONOUNS property. (Section 2.3.36).

Its grammaticalGender property converts to the property value of the GRAMGENDER property.

Its pronouns property converts to PRONOUNS properties, one for each entry. The pronouns property of the Pronouns object converts the property value. Its other properties convert as follows:

Name	Reference	Parameter (or other)	See
contexts	[RFC9553], Section 2.2.4	TYPE	Section 2.2.22
pref	[RFC9553], Section 2.2.4	PREF	Section 2.2.17

Table 24: Properties of the Pronouns object

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.20. Title

The Title object converts to one of the following properties, determined by the Title kind property value. If the kind is "title" or not set then it converts to the TITLE property. (Section 2.3.44). If the kind is "role" then it converts to the ROLE property (Section 2.3.39).

Its name property converts the property value. Its organizationId property converts by assigning the TITLE property and the ORG property to the same vCard property group (see Section 2.3.39).

Other properties MAY be converted to JSPROP properties (Section 5.2.1).

3.21. New vCard Parameters

4. Updates to JSContact

4.1. New Properties

4.1.1. vCard

Name:
vCard

Context:
A Card object

Type:
VCard (optional)

Description
This property contains information about vCard data that got converted to a JSContact Card object. It allows for preserving arbitrary vCard elements, such as extension properties and parameters. This specification defines the VCard object type for converting a vCard to a JSContact Card object. Future specifications MAY define additional value types for this property.

A VCard object has the following properties:

@type: String This specifies the type of this object. This MUST be VCard, if set.

convertedProperties: String[VCardProperty] (optional) This contains conversion-related information about the vCard properties that got partially or fully converted to JSContact. Each key defines the path to a JSContact element. The value for each key contains information about the vCard property which converted to the JSContact element located at that key. Unless otherwise defined for a specific vCard property, the path points to that JSContact object property to which the vCard property value converted to.

The key MUST be a valid key of a PatchObject as defined in Section 1.4.3 of [RFC9553]. The key MAY point into a nested object.

properties: *[][] (optional) This contains properties of the vCard which did not convert to standard JSContact elements. The value MUST be a list of vCard properties formatted in jCard as defined in Section 3.3 of [RFC7095].

A VCardProperty object has the following properties:

@type: String This specifies the type of this object. This MUST be VCardProperty, if set.

name: String (mandatory) This is the name of the vCard property in lowercase.

valueType: String (optional) This is the name of the vCard property value type in lowercase.

parameters: String[String|String[]] (optional) This contains parameters of the vCard property. The value MUST comply with vCard parameters formatted in jCard as defined in Section 3.4 of [RFC7095].

The example in Figure 54 describes how to convert unknown vCard elements to the "vCard" property. The "X-FOO" parameter set on the FN property illustrates how to preserve an unknown vCard parameter on a property that does convert to a standard JSContact element. The "X-BAR" property and "X-BAZ" component illustrate how to convert entirely unknown vCard properties.

```
BEGIN:VCARD
VERSION:4.0
FN;X-FOO=bar:test
...
X-BAR:bam
...
"name": {
  "full": "test"
},
"vCard": {
  "version": "4.0",
  "convertedProperties": {
    "name/full": {
      "name": "fn",
      "parameters": {
        "x-foo": "bar"
      }
    }
  },
  "properties": [
    [
      "x-bar",
      {},
      "unknown",
      "bam"
    ]
  ],
}
```

Figure 54: Converting unknown vCard elements to the vCard property

5. Updates to vCard

5.1. New Parameters

5.1.1. 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 55 and 56 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 57 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 special characters 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 55: 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 56: 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,\, ;10;s, ;11;3":;;54321 Oak St;Reston;;;;;;;Oak St;54321;;;;;;
```

Figure 57: Example of Separator Entries

5.1.2. JSID

Parameter Name:
JSID

Purpose:
Specifies the JSContact identifier of the associated property.

Description:
This parameter specifies the JSON object key of the JSContact object to which a vCard property converts to. For example, this parameter set on an EMAIL property defines the key of the EmailAddress object in the Card object's "emails" property (see below for an example).

The value of the JSID parameter MUST be a valid key according to the definition of the JSContact property to which the property that this parameter is set on converts to. Typically, this requires the value to be a valid Id [RFC9553] (Section 1.4.1).

Format Definition:

```
jsid-param  = "JSID" "=" param-value
```

Example(s):

```
EMAIL;JSID=xyz:jane_doe@example.com
"emails": {
  "xyz": {
    "address": "jane_doe@example.com"
  }
}
```

Figure 58: Converting the JSID parameter on an EMAIL property

5.1.3. JSPTR

Parameter name: JSPTR

Purpose: To specify a pointer to a JSContact property.

Description: This parameter is set on a JSPROP (Section 5.2.1) property. Its value points to a JSContact property. The parameter has a single value that MUST be a valid PatchObject key as defined in Section 1.4.3 of [RFC9553]. The parameter value MUST be quoted and is case-sensitive.

Format definition:

```
jsptr-param  = "JSPTR" "=" DQUOTE *QSAFE-CHAR DQUOTE
```

Example(s): For further examples, see Section 5.2.1.

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

5.2. New vCard Properties

5.2.1. 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 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 59: 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 60: 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 61: Nested Vendor-Specific Property Example with a Path Relative to Card

6. 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].

7. IANA Considerations

TBD

8. References

8.1. Normative References

[I-D.ietf-calext-jscontact-profiles]
Stepanek, R. and M. Loffredo, "Protocol-Specific Profiles for JSContact", Work in Progress, Internet-Draft, draft-ietf-calext-jscontact-profiles-14, 17 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-calext-jscontact-profiles-14>.

[I-D.ietf-calext-jscontact-uid]
Stepanek, R., "JSContact Version 2.0: A JSON Representation of Contact Data", Work in Progress, Internet-Draft, draft-ietf-calext-jscontact-uid-07, 19 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-calext-jscontact-uid-07>.

[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>.

[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, March 2024, <https://www.rfc-editor.org/info/rfc9553>.

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

[RFC9555] Loffredo, M. and R. Stepanek, "JSContact: Converting from and to vCard", RFC 9555, DOI 10.17487/RFC9555, May 2024, <https://www.rfc-editor.org/info/rfc9555>.

8.2. 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>.

[I-D.ietf-calext-jscalendar-icalendar] Stepanek, R., "JSCalendar: Converting from and to iCalendar", Work in Progress, Internet-Draft, draft-ietf-calext-jscalendar-icalendar-22, 22 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-calext-jscalendar-icalendar-22>.

[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, 7 June 2018, <https://datatracker.ietf.org/doc/html/draft-calconnect-vobject-i18n-00>.

Appendix A. Differences from RFC 9555

This section documents all significant differences from RFC 9555. Insignificant differences, such as formatting, grammar or typos are not documented.

A.1. Applied Errata

The following errrata was applied to to this document:

* Errata 8555 (https://www.rfc-editor.org/errata/eid8555)

A.2. Changed Conversion Rules

A.2.1. vCard to JSContact

* Converting the UID property to the "uid" property is redefined for JSContact version "2.0". See Section 2.1.1.

A.3. Changes to JSContact

A.3.1. Obsoleted Properties

The following JSContact properties became obsolete:

vCardProps
: This property is deprecated, the "properties" property of the newly defined VCard type (Section 4.1.1) replaces it. Figure 62 illustrates this using the same example asSection 2.15.1 of [RFC9555].

```
"vCardProps": [
  ["x-foo", { }, "unknown", "bar"]
]
"vCard": {
  "properties": [
    ["x-foo", { }, "unknown", "bar"]
  ]
}
```

Figure 62: Example for Representing the Obsolete vCardProps Property

vCardParams
: This property is deprecated, the "parameters" property of the newly defined VCardProperty type (Section 4.1.1) replaces it. Figure 63 illustrates this using the same example as Section 2.15.2 of [RFC9555].

```
      "emails": {
        "email1": {
          "address": "jane_doe@example.com",
          "vCardParams": {
            "x-foo": "Bar"
          }
        }
      }
      "emails": {
        "email1": {
          "address": "jane_doe@example.com"
        }
      }
      "vCard": {
        "convertedProperties": {
          "emails/email1/address": {
            "name": "email",
            "parameters": {
              "x-foo": "Bar"
            }
          }
        }
      }
```

Figure 63: Example for Representing the Obsolete vCardParams Property

vCardName
This property is deprecated, the "name" property of the newly defined VCardProperty type (Section 4.1.1) replaces it. Figure 64 illustrates this using the same example as Section 2.15.3 of [RFC9555].

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

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

Figure 64: Example for Representing the Obsolete vCardName Property

A.3.2. Obsoleted Types

The following JSContact type definitions became obsolete:

JCardProp
: This type is deprecated because the "vCardProps" property became deprecated. Also see Appendix A.3.1, Paragraph 2.

A.4. Changes to vCard

A.4.1. Obsoleted Parameters

The following vCard parameters became obsolete:

PROP-ID
: This parameter is deprecated, the newly defined JSID parameter (Section 5.1.2) replaces it. Section 2.2.18 defines how to process the PROP-ID in presence and absence of the JSID parameter.

Acknowledgements

The definition and examples of the PHONETIC (Section 2.2.15) and SCRIPT (Section 2.2.19) 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