Registration Protocols Extensions
Internet-Draft
Intended status: Standards Track
Expires: 1 November 2026

M. Loffredo
IIT-CNR/Registro.it
G. Brown
ICANN

30 April 2026

Using JSContact in Registration Data Access Protocol (RDAP) JSON Responses
draft-ietf-regext-rdap-jscontact-25

Abstract

This document describes an RDAP extension which represents entity contact information in JSON responses using JSContact.

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 1 November 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

This document specifies an extension to the Registration Data Access Protocol (RDAP) that allows RDAP servers to use JSContact [RFC9553] to represent the contact information associated with entities in RDAP responses, instead of jCard [RFC7095]. It also describes the process by which an RDAP server can transition from jCard to JSContact. RDAP query and response extensions are defined to facilitate the transition process.

1.1. Rationale

According to the feedback from RDAP Pilot Working Group [RDAP-PILOT-WG], a group of RDAP server implementers representing registries and registrars of generic TLDs, the most commonly raised implementation concern for both servers and client implementers, was related to the use of jCard [RFC7095] to represent the contact information associated with entities. Working Group members reported jCard to be unintuitive, complicated to implement for both clients and servers, and incompatible with best practices for RESTful APIs.

JSContact [RFC9553] provides a simpler and more efficient representation for contact information with regard to time and effort saved in processing it. In addition, similarly to jCard, it provides a means to represent internationalized and unstructured contact information. Support for internationalized contact information has been recognised being necessary to facilitate the future internationalisation of registration data directory services.

1.2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. JSContact

The JSContact specification defines a data model and JSON representation of contact information that can be used for data storage and exchange in address book or directory applications. It aims to be an alternative to the vCard data format [RFC6350] and to be unambiguous, extendable and simple to process. In contrast with jCard, it is not a direct mapping from the vCard data model and expands semantics where appropriate.

JSContact differs from jCard in that it:

* follows an object-oriented rather than array-oriented approach;
* is simple to process;
* requires no extra work in serialization/deserialization from/to a data model;
* includes no "jagged" arrays;
* prefers maps rather than arrays to implement collections.

[RFC9555] provides guidance on the conversion of jCard into JSContact, and vice versa. Appendix A shows JSContact counterparts for the most commonly used jCard properties in an RDAP response.

3. Using JSContact in RDAP

3.1. JSContact Profile for RDAP

Since JSContact is a generalized representation of contact data, many of its capabilities are not optimized for use in RDAP responses. This document defines a usage profile for JSContact that simplifies the implementation for both clients and servers. The JSContact profile for RDAP is compliant with the rules described in [I-D.ietf-calext-jscontact-profiles]. The profile properties are formally listed in Section 3.1.14. The information required to register the profile is described in Section 6.4.

All types mentioned in this section refer to the JSContact specification [RFC9553] (e.g., Card is the JSContact Card type).

3.1.1. Extension Identifier

Entity objects in RDAP responses MAY include a "jscontact_card" member whose value is a Card object instead of the "vCardArray" member defined in [RFC9083].

Servers returning the "jscontact_card" member in their response MUST include the string "jscontact" in the "rdapConformance" array.

3.1.2. Extension Version Identifier

This extension supports the following versioning types as defined in [I-D.ietf-regext-rdap-versioning]:

* Opaque Versioning: The Opaque Extension Versioning Identifier is "jscontact".
* Point Versioning: The Point Extension Versioning Identifier is "jscontact-0.4".

Note that this extension identifier’s version value may differ from the Card version member value. In fact, the former is related to changes in this RDAP extension, while the latter is related to changes in the JSContact specification.

NOTE: Please change the Point Versioning value to "jscontact-1.0" prior to publication as an RFC.

3.1.3. Version

The information required to register the Card "version" member value in the "RDAP JSON Values" registry [RFC9083] is described in Section 6.2.

3.1.4. Kind

The Card "kind" member value MUST be "individual" (default) or "org" to represent an individual or an organization, respectively.

3.1.5. Language

The Card "language" member SHOULD be set when localizations are specified (i.e. the "localizations" member is not null).

3.1.6. Name

The Card "name" member MUST include the "full" member and MAY include the "components" member to specify the name components of an individual.

The NameComponent type MUST include only the "kind" and "value" members. The "kind" member value MUST be "given" or "surname".

3.1.7. Organizations

The Organization type MUST include only the "org" member.

3.1.8. Addresses

The Address type MUST include at least one between the "full", "components" and "countryCode" members.

The AddressComponent type MUST include only the "kind" and "value" members. The "kind" member value MUST be "name", "locality", "region", "postcode" or "country".

3.1.9. Emails

The EmailAddress type MUST include only the "address" member.

3.1.10. Phones

The Phone type MUST include the "number" member and MAY include the "features" member.

The PhoneFeature type value MUST be "voice" or "fax". When the "features" member is missing, the phone number is assumed to be a voice number.

3.1.11. Links

The Link type MUST include the "uri" member and MAY include the "kind" member. The "kind" member value MUST be "contact".

3.1.12. Map Keys

Since most of the JSContact collections are represented as maps, map keys must be defined. A JSContact map key MUST comply with the definition of the Id type (see Section 1.4.1 of [RFC9553]). To aid interoperability, RDAP providers MUST use the following keys related to string values and labels defined in [RFC5733]:

* "org" in the "organizations" map when there is a single <contact:org> element;
* "addr" in the "addresses" map when there is a single <contact:addr> element;
* "email" in the "emails" map for the <contact:email> element;
* "voice" in the "phones" map for the <contact:voice> element;
* "fax" in the "phones" map for the <contact:fax> element.

In cases where both internationalized and localized versions of organization, postal address, and email exist, the related map MUST include the internationalized version, while the localized version MUST be included in the localizations map, as described in Section 3.1.13.

Additional keys of the "links" map that MUST be used to represent contact links which result from converting jCard "uri" and "contact-uri" properties are:

* "url" used for the unique or preferred contact url (e.g. a web site). The "kind" member of the Link object MUST NOT be set;
* "contact-uri" used for the contact uri as defined in [RFC8605]. The "kind" member of the Link object MUST be set to "contact".

The information required to register the JSContact Id values in the "RDAP JSON Values" registry [RFC9083] is described in Section 6.2.

Any additional key used for the above maps SHOULD conform with the following scheme: "<registered key>-<sequential number>". For example, a phones map containing three voice numbers would contain the following keys: "voice", "voice-1", "voice-2".

The keys SHOULD remain stable until the associated contact information is changed. Any gaps in the numbering MAY only occur when redaction is applied.

3.1.13. Localizations

If present, the localized versions of name, organization, postal address and email MUST be added to the "localizations" map. With reference to the definition of localization in [RFC9553], an RDAP response with JSContact content MUST expand all localizations (i.e. a nested PatchObject key like "{key1}/{key2}/.../{keyN}" is not allowed). The following is an elided example of an RDAP entity lookup response including a Card that includes localized version for name, organization and postal address (See PDF for non-ASCII character string).

```
...
"jscontact_card": {
  "@type": "Card",
  "version": "2.0",
  "language": "en",
  "name": {
    "full": "Vasya Pupkin"
  },
  "organizations": {
    "org": {
      "name": "My Company"
    }
  },
  "addresses": {
    "addr": {
      "components": [
        { "kind": "name", "value": "1 Street" },
        { "kind": "locality", "value": "Kyiv" }
      ],
      "countryCode": "UA"
    }
  },
  "localizations": {
    "ua": {
      "addresses": {
        "addr": {
          "components": [
           { "kind": "name", "value": "1, У л и ц а" },
           { "kind": "locality", "value": "К и е в" }
          ],
          "countryCode": "UA"
        }
      },
      "name": {
        "full": "В а с я  П у п к и н"
      },
      "organizations": {
        "org": {
          "name": "М о я  К о м п а н и я"
        }
      }
    }
  }
}
...
```

Figure 1: Example of handling localizations in JSContact

3.1.14. Profile Properties

The properties of the JSContact profile registered in the JSContact Profile registry as described in Section 6.4 are the following:

Property Name	Property Context	Restricted Attributes	Restricted Enum Values	Restricted PatchObject Keys
language	Card			
kind	Card		individual, org	
name	Card			
full	Name	mandatory		
components	Name			
kind	NameComponent		given, surname	
value	NameComponent			
organizations	Card			
name	Organization			
addresses	Card			
full	Address			
countryCode	Address			
components	Address			
kind	AddressComponent		name, locality, region, postcode, country	
value	AddressComponent			
emails	Card			

address	EmailAddress				
phones	Card				
features	Phone		voice, fax		
number	Phone				
links	Card				
kind	Link		contact		
uri	Link				
localizations	Card			yes	

Table 1

For each property, the "Restricted Property Type" field is empty.

To be consistent with what is stated in Section 2.1 of [RFC9083], clients SHOULD ignore JSContact properties other than these.

3.1.15. Example of using JSContact in RDAP response

The following is an example of an RDAP entity including a Card.

```
{
   "rdapConformance": [
      "rdap_level_0",
      "jscontact"
   ],
   "objectClassName": "entity",
   "handle":"XXXX",
   "jscontact_card":{
     "@type": "Card",
     "version": "2.0",
     "name": {
       "full": "Joe User",
       "components": [
         { "kind": "given", "value": "Joe" },
         { "kind": "surname", "value": "User" }
       ]
     },
     "organizations": {
       "org": {
         "name": "Org Example"
```

```
            }
          },
           "addresses": {
             "addr": {
               "components": [
                 { "kind": "name", "value": "Main Street 1" },
                 { "kind": "locality",
                   "value": "Ludwigshafen am Rhein" },
                 { "kind": "region", "value": "Rhineland-Palatinate" },
                 { "kind": "postcode", "value": "67067" },
                 { "kind": "country", "value": "Germany" }
               ],
               "countryCode": "DE"
             },
             "addr-1": {
               "full": "Somewhere Street 1 Mutterstadt 67112 Germany"
             }
           },
          "phones": {
            "voice": {
              "number": "tel:+49-1522-3433333"
            },
            "fax": {
              "features": { "fax": true },
              "number": "tel:+49-30-901820"
            }
          },
          "emails": {
            "email": {
              "address": "joe.user@example.com"
            }
          },
          "links": {
            "url": {
              "uri": "https://www.example.com"
            },
            "contact-uri": {
              "kind": "contact",
              "uri": "mailto:contact@example.com"
            }
          }
        },
        "roles":[ "registrar" ],
        "publicIds":[
          {
            "type":"IANA Registrar ID",
            "identifier":"1"
          }
```

```
        ],
        "links":[
          {
            "value":"https://example.com/entity/XXXX",
            "rel":"self",
            "href":"https://example.com/entity/XXXX",
            "type" : "application/rdap+json"
          }
        ],
        "events":[
          {
            "eventAction":"registration",
            "eventDate":"1990-12-31T23:59:59Z"
          }
        ],
        "asEventActor":[
          {
            "eventAction":"last changed",
            "eventDate":"1991-12-31T23:59:59Z"
          }
        ]
      }
```

Figure 2: Example of using JSContact in RDAP response

3.2. Request for JSContact

The client MAY request the server to include the "jscontact_card" member in the RDAP response by using only one of the following HTTP elements:

* the "versioning" query parameter as defined in [I-D.ietf-regext-rdap-versioning];
* the RDAP media type's "exts_list" parameter as defined in [I-D.ietf-regext-rdap-x-media-type].

Their usage in the transtion from jCard to JSContact is further explained in Section 4.2.2.2.

3.3. Reverse Search Properties

The use of JSContact updates the mappings of two reverse search properties, namely "fn" and "email", defined in [RFC9536]. Such new mappings are registered in the Reverse Search Mapping registry as described in Section 6.3.

4. Transition Considerations

4.1. RDAP Features Supporting the Transition Process

RDAP allows servers to communicate service information to clients through notices. According to Section 4.3 of [RFC9083], an RDAP response may contain one or more notice objects. Each notice may include a set of link objects, which can be used to provide clients with references. These link objects may have a "rel" member which defines the relationship between resources, as described in Section 4 of [RFC8288]. The transition process outlined in this document uses the "alternate" link relation type.

The information about the specifications used in the construction of the response is also described by the strings that appear in the "rdapConformance" array of the RDAP response.

4.2. Transition Procedure

The transition procedure consists of three contiguous stages. During the procedure, the presence of the string "jscontact" in the "rdapConformance" array indicates that JSContact is returned instead of jCard. The date and time format used to notify clients about the stages of this procedure is defined in [RFC3339].

4.2.1. Goals

The procedure described in this document aims to achieve the following goals:

* only one contact representation would be included in the response;
* the response would always conform to [RFC9083] because:
  - since the "jscontact_card" member is a response extension, its presence would be signaled by the string "jscontact" in the "rdapConformance" array;
  - since "vcardArray" member is optional in a response, its absence would be allowed;
* clients would be informed about the transition timeline;
* the backward compatibility would be ensured throughout the transition;
* servers and clients could perform their transitions independently.

4.2.2. Transition Stages

4.2.2.1. Stage 1: only jCard provided

This stage corresponds to providing jCard as the default contact card [RFC9083]. The RDAP server is not able to provide an alternate format for contacts. The "rdapConformance" array MUST NOT contain the string "jscontact".

4.2.2.2. Stage 2: jCard sunset

During this stage, the server returns jCard by default, but the RDAP server will return JSContact if the client requests it through one of the methods described in Section 3.2. The "rdapConformance" array MUST contain the string "jscontact" if JSContact is returned.

From this stage onward, the RDAP server MUST include the string "jscontact" in the "rdapConformance" array of the help response to signal clients that JSContact can be returned instead of jCard.

If JSContact is not requested, the RDAP server SHOULD include a notice whose type member is set to "jCard sunset end". This notice contains a description that reports the jCard sunset end date and time, and an OPTIONAL link to the corresponding JSContact-based response requested via the method used by the client to negotiate extensions (Section 3.2). If neither method was used, the RDAP server includes links for both methods (Figure 3).

```
"notices": [
  {
    "type": "jCard sunset end",
    "description": ["2025-12-31T23:59:59Z"],
    "links": [
      {
        "value": "https://example.net/entity/XXXX",
        "rel": "alternate",
        "type": "application/rdap+json",
        "href":
          ".../entity/XXXX?versioning=versioning-0.6,jscontact-0.4"
      },
      {
        "value": "https://example.net/entity/XXXX",
        "rel": "alternate",
        "type":
          "application/rdap+json;exts_list=\"rdap_level_0 jscontact\"",
        "href": "https://example.net/entity/XXXX"
      }
    ]
  }
]
```

Figure 3: jCard sunset - JSContact not requested

4.2.2.3. Stage 3: jCard deprecation

This stage corresponds to providing JSContact as the default format for contact data. The "rdapConformance" array always contains the string "jscontact". The request for JSContact through any method described in Section 3.2 MUST be ignored.

The RDAP server MUST include the string "noJcard" in the "rdapConformance" array of the help response to signal clients that jCard is no longer returned.

The RDAP server SHOULD also include a notice whose type member is set to "jCard deprecation" whose description reports the verbatim string "jCard has been deprecated" (Figure 4).

```
"notices": [
  {
    "type": "jCard deprecation",
    "description": ["jCard has been deprecated"],
  }
]
```

Figure 4: jCard deprecation - jCard has been deprecated

4.2.2.4. Length

The length of the jCard sunset period is not fixed by this specification. Anyway, RDAP providers are RECOMMENDED to monitor the server log to figure out whether the declared time needs to be changed to meet client requirements.

5. Implementation Status

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

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

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

5.1. IIT-CNR/Registro.it RDAP Server

* Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
* Location:https://rdap.pubtest.nic.it/
* Description: This implementation includes support for RDAP queries using data from the public test environment of .it ccTLD.
* Level of Maturity: This is an "alpha" test implementation.
* Coverage: This implementation includes all of the features described in this specification.
* Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

5.2. IIT-CNR/Registro.it RDAP Client

* Responsible Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it
* Location: https://web-rdap.pubtest.nic.it/
* Description: This is a Javascript web-based RDAP client. RDAP responses are retrieved from RDAP servers by the browser, parsed into an HTML representation, and displayed in a format improving the user experience. RDAP responses containing JSContact objects are handled identically to those containing jCard objects. Raw versions of RDAP responses including either jCard or JSContact objects are provided.
* Level of Maturity: This is an "alpha" test implementation.
* Coverage: This implementation includes all of the features described in this specification.
* Contact Information: Francesco Donini, francesco.donini@iit.cnr.it

5.3. client.rdap.org

* Location: https://client.rdap.org/
* Description: This is a web-based "single page" RDAP client. RDAP responses are retrieved from RDAP servers by the browser, and parsed into an HTML representation. RDAP responses containing JSContact objects are handled identically to those containing jCard objects.
* Level of Maturity: This is an "alpha" test implementation.
* Coverage: This implementation implements client support for parsing JSContact objects in RDAP responses.
* Contact Information: Gavin Brown, feedback@rdap.org

5.4. CentralNic Registry

* Responsible Organization: CentralNic Group PLC
* Location: https://rdap.centralnic.com/{tld}
* Description: This server is the product RDAP service for all top-level domains on the CentralNic registry platform.
* Level of Maturity: Production quality.
* Coverage: This implementation includes all of the features described in this specification.
* Contact Information: support@centralnic.com

5.5. ICANN RDAP

* Responsible Organization: ICANN
* Location: https://github.com/icann/icann-rdap
* Description: ICANN RDAP contains an RDAP client, server, and common libraries.
* Level of Maturity: Mature.
* Coverage: This implementation includes all the features described in this specification, except support for semantic versioning and the provision of reverse search mapping.
* Contact Information: globalsupport@icann.org

6. IANA Considerations

6.1. RDAP Extensions Registry

IANA is requested to register the following values in the RDAP Extensions Registry [IANA.rdap-extensions]:

* Extension identifier: jscontact
* Registry operator: Any
* Published specification: This document.
* Contact: IETF <iesg@ietf.org>
* Intended usage: This extension identifier is used as prefix for the "jscontact_card" member returned in the JSON response [RFC9553].

* Extension identifier: noJcard
* Registry operator: Any
* Published specification: This document.
* Contact: IETF <iesg@ietf.org>
* Intended usage: This extension identifier is used by the server to signal clients that jCard is no longer returned.

6.2. RDAP JSON Values Registry

With regard to the fields of the RDAP JSON Values Registry [IANA.rdap-json-values], the "JSContact version value" type SHALL be used to register the RDAP values for the Card version member while the "JSContact Id value" type SHALL be used to register the RDAP values for the JSContact Id type as defined in Section 3.1.12.

IANA is requested to register the following values in this registry:

Value: jCard sunset end
Type: notice and remark type
Description: This notice signals clients that JSContact can be returned upon request.
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

Value: jCard deprecation
Type: notice and remark type
Description: This notice signals clients that jCard is no longer returned.
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

Value: 2.0
Type: JSContact version value
Description: The JSContact version [I-D.ietf-calext-jscontact-uid] to use in implementing this specification.
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

Value: org
Type: JSContact Id value
Description: The key in the JSContact "organizations" map identifiying the contact organization.
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

Value: addr
Type: JSContact Id value
Description: The key in the JSContact "addresses" map identifiying the unique or preferred contact postal address.
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org

Reference: This document

Value: email
Type: JSContact Id value
Description: The key in the JSContact "emails" map identifiying the unique or preferred contact email address.
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

Value: voice
Type: JSContact Id value
Description: The key in the JSContact "phones" map identifiying the unique or preferred contact voice number.
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

Value: fax
Type: JSContact Id value
Description: The key in the JSContact "phones" map identifiying the unique or preferred contact fax number.
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

Value: url
Type: JSContact Id value
Description: The key in the JSContact "links" map identifiying the unique or preferred url.
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

Value: contact-uri
Type: JSContact Id value
Description: The key in the JSContact "links" map identifiying the contact uri as defined in [RFC8605].
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

6.3. RDAP Reverse Search Mapping Registry

IANA is requested to register the following entries in the RDAP Reverse Search Mapping Registry [IANA.rdap-reverse-search-mapping]:

Searchable Resource Type: domains, nameservers, entities, ips,

autnums
Related Resource Type: entity
Property: fn
Property Path: $.entities[*].jscontact_card.[name.full,
localizations.*.name.full]
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

Searchable Resource Type: domains, nameservers, entities
Related Resource Type: entity
Property: email
Property Path: $.entities[*].jscontact_card.[emails.email.address,
localizations.*.emails.email.address]
Registrant Name: IETF
Registrant Contact Information: iesg@ietf.org
Reference: This document

6.4. JSContact Profile Registry

IANA is requested to register the following entry in the "JSContact Profile" registry [I-D.ietf-calext-jscontact-profiles]:

Name: rdap
Profile Version: 1
Reference: Section 3.1.14

7. Operational Considerations

This section describes operational considerations for deploying JSContact in RDAP, taking into account the request mechanisms defined in Section 3.2 and the transition approach described in Section 4.

7.1. Representation Selection

As described in Section 3.2, clients can request the "jscontact_card" member using either the "versioning" query parameter or the RDAP media type "exts_list" parameter.

Servers are expected to return a representation that is consistent with both the client request and the applicable transition stage defined in Section 4.2.2.

In particular, jCard is returned in Stage 1, jCard is the default in Stage 2 with JSContact available upon request, and JSContact is the only representation returned in Stage 3.

If a request for JSContact cannot be satisfied (e.g., due to unsupported or unknown extension parameters), the request is processed according to general RDAP behavior, and the default representation is returned.

7.2. rdapConformance Signaling

The "rdapConformance" array provides an indication of the representation used, as described in Section 3.1.1 and Section 4.2.

Servers are expected to ensure that the presence or absence of the "jscontact" identifier accurately reflects the response content and is consistent across all endpoints, including help responses as required in Section 4.2.2.2 and Section 4.2.2.3.

7.3. Data Consistency

JSContact and jCard represent the same information in different ways. As described in Appendix A, mapping between the two formats is not always structurally equivalent.

Operators should ensure that both representations are derived from a consistent data source and that the semantic meaning of the data is preserved, even when structural differences exist.

Particular attention may be needed for localized dat (Section 3.1.13) and for the use of map keys (Section 3.1.12), to avoid inconsistencies.

7.4. Transition Monitoring

As stated in Section 4.2.2.4, the duration of the transition is not fixed.

Operators are encouraged to monitor service usage (e.g., server logs) to assess client adoption of JSContact and to determine whether adjustments to the transition timeline are appropriate.

7.5. Caching

Because different representations of the same resource may be returned depending on request parameters (Section 3.2) and transition stage (Section 4), caching mechanisms may need to account for these differences.

Operators should consider whether cache keys and intermediaries correctly distinguish between representations to avoid serving unexpected results.

7.6. Error Handling

Error handling follows the general RDAP principles defined in [RFC9083] and the extension versioning behavior described in Section 5.1 of [I-D.ietf-regext-rdap-versioning] and Section 3 of [I-D.ietf-regext-rdap-x-media-type].

In particular, requests for unsupported or unknown extensions or extension versions do not result in an error response and are instead ignored.

If a request for JSContact cannot be satisfied, the server behavior is determined by the applicable transition stage defined in Section 4.2.2. In Stage 2, the server returns the default representation (jCard), while in Stage 3 JSContact is always returned.

More generally, when extension negotiation does not succeed, servers may return the default representation in accordance with RDAP specifications.

7.7. Interoperability

Interoperability issues may arise during the transition due to differences in client support for request mechanisms (Section 3.2) or interpretation of rdapConformance signaling and notices (Section 4.1).

Testing with multiple client implementations can help identify and mitigate such issues.

7.8. Operational Resilience

The transition procedure described in Section 4.2 allows servers and clients to evolve independently.

Operators may adjust deployment strategies, including the duration of transition stages, based on operational experience and client behavior.

8. Privacy Considerations

Unlike jCard, the formatted name as well as any other personally identifiable information is not required in JSContact 1.0 [RFC9553]. JSContact version 2.0 [I-D.ietf-calext-jscontact-uid] makes also uid property optional. Therefore, with reference to what is described in [RFC9537], properties can be redacted by Removal Method.

9. Security Considerations

The extension described in this document does not provide any security services beyond those described by [RFC9083].

10. Acknowledgements

The authors would like to acknowledge the following individuals for their contributions to this document: Jasdip Singh, Andrew Newton, Marc Blanchet, Rick Wilhelm, Pawel Kowalik and Francesco Donini.

11. References

11.1. 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>.

[I-D.ietf-opsawg-rfc5706bis]
Claise, B., Clarke, J., Farrel, A., Barguil, S., Pignataro, C., and R. Chen, "Guidelines for Considering Operations and Management in IETF Specifications", Work in Progress, Internet-Draft, draft-ietf-opsawg-rfc5706bis-04, 15 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-rfc5706bis-04>.

[I-D.ietf-regext-rdap-versioning]
Gould, J., Keathley, D., and M. Loffredo, "Versioning in the Registration Data Access Protocol (RDAP)", Work in Progress, Internet-Draft, draft-ietf-regext-rdap-versioning-05, 21 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-regext-rdap-versioning-05>.

[I-D.ietf-regext-rdap-x-media-type]
Newton, A. and J. Singh, "The "exts_list" Parameter for the RDAP Media Type", Work in Progress, Internet-Draft,

draft-ietf-regext-rdap-x-media-type-05, 2 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-regext-rdap-x-media-type-05>.

[IANA.rdap-extensions]
IANA, "IANA RDAP Extensions Registry", <https://www.iana.org/assignments/rdap-extensions>.

[IANA.rdap-json-values]
IANA, "IANA RDAP JSON Values Registry", <https://www.iana.org/assignments/rdap-json-values>.

[IANA.rdap-reverse-search-mapping]
IANA, "IANA RDAP Reverse Search Mapping Registry", <https://www.iana.org/assignments/rdap-reverse-search-mapping>.

[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>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <https://www.rfc-editor.org/info/rfc5733>.

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

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

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

[RFC9083] Hollenbeck, S. and A. Newton, "JSON Responses for the Registration Data Access Protocol (RDAP)", STD 95, RFC 9083, DOI 10.17487/RFC9083, June 2021, <https://www.rfc-editor.org/info/rfc9083>.

[RFC9536] Loffredo, M. and M. Martinelli, "Registration Data Access Protocol (RDAP) Reverse Search", RFC 9536, DOI 10.17487/RFC9536, April 2024, <https://www.rfc-editor.org/info/rfc9536>.

[RFC9553] Stepanek, R. and M. Loffredo, "JSContact: A JSON Representation of Contact Data", RFC 9553, DOI 10.17487/RFC9553, May 2024, <https://www.rfc-editor.org/info/rfc9553>.

[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>.

11.2. Informative References

[RDAP-PILOT-WG] ICANN RDAP Pilot WG, "RDAP Pilot Report", April 2019, <https://www.icann.org/en/system/files/files/rdap-pilot-report-25apr19-en.pdf>.

[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>.

[RFC9535] Gssner, S., Ed., Normington, G., Ed., and C. Bormann, Ed., "JSONPath: Query Expressions for JSON", RFC 9535, DOI 10.17487/RFC9535, February 2024, <https://www.rfc-editor.org/info/rfc9535>.

[RFC9537] Gould, J., Smith, D., Kolker, J., and R. Carney, "Redacted Fields in the Registration Data Access Protocol (RDAP) Response", RFC 9537, DOI 10.17487/RFC9537, March 2024, <https://www.rfc-editor.org/info/rfc9537>.

Appendix A. jCard-JSContact Mapping

Provided that the keys defined in Section 3.1.12 are used for the JSContact maps, the mapping between the most commonly used jCard properties in an RDAP response and their JSContact counterparts is shown in the following. The mapping is done through the use of JSONPath expressions [RFC9535].

jCard property: fn
Reference: Section 6.2.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]==’fn’)][3]
JSContact property: Name.full
Reference: Section 2.2.1 of [RFC9553]
Path: $..jscontact_card.[name.full, localizations.*.name.full]

jCard property: "family name" component of n
Reference: Section 6.2.2 of [RFC6350]
Path: $..vcardArray[1][?(@[0]==’n’)][3][0]
JSContact property: "postOfficeBox" NameComponent of Name.components
Reference: Section 2.5.1 of [RFC9553]
Path:
$..jscontact_card.[name.components[?(@.kind==’surname’)].value,
localizations.*.name.components[?(@.kind==’surname’)].value]

jCard property: "given name" component of n
Reference: Section 6.2.2 of [RFC6350]
Path: $..vcardArray[1][?(@[0]==’n’)][3][1]
JSContact property: "given" NameComponent of Name.components
Reference: Section 2.5.1 of [RFC9553]
Path: $..jscontact_card.[name.components[?(@.kind==’given’)].value,
localizations.*.name.components[?(@.kind==’given’)].value]

jCard property: n
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]==’n’)]
JSContact property: Name
Reference: Section 2.5.1 of [RFC9553]
Path: $..jscontact_card.[name, localizations.*.name]

jCard property: org
Reference: Section 6.6.4 of [RFC6350]
Path: $..vcardArray[1][?(@[0]==’org’)][3]
JSContact property: Organization.name
Reference: Section 2.2.3 of [RFC9553]
Path: $..jscontact_card.[organizations.org.name,
localizations.*.organizations.org.name]

jCard property: tel with type="voice"

```
   Reference:  Section 6.4.1 of [RFC6350]
   Path:  $..vcardArray[1][?(@[1].type==’voice’)][3]
   JSContact property:  Phone.number
   Reference:  Section 2.3.3 of [RFC9553]
   Path:  $..jscontact_card.phones.voice.number

   jCard property:  tel with type="fax"
   Reference:  Section 6.4.1 of [RFC6350]
   Path:  $..vcardArray[1][?(@[1].type==’fax’)][3]
   JSContact property:  Phone.number
   Reference:  Section 2.3.3 of [RFC9553]
   Path:  $..jscontact_card.phones.fax.number

   jCard property:  email
   Reference:  Section 6.4.2 of [RFC6350]
   Path:  $..vcardArray[1][?(@[0]==’email’)][3]
   JSContact property:  Email.address
   Reference:  Section 2.3.1 of [RFC9553]
   Path:  $..jscontact_card.[emails.email.address,
      localizations.*.emails.email.address]

   jCard property:  "label" parameter of adr
   Reference:  Section 6.3.1 of [RFC6350]
   Path:  $..vcardArray[1][?(@[0]==’adr’)][1].label
   JSContact property:  Address.full
   Reference:  Section 2.5.1 of [RFC9553]
   Path:  $..jscontact_card.[addresses.adr.full,
      localizations.*.addresses.adr.full]

   jCard property:  "post office box" component of adr
   Reference:  Section 6.3.1 of [RFC6350]
   Path:  $..vcardArray[1][?(@[0]==’adr’)][3][1]
   JSContact property:  "postOfficeBox" AddressComponent of
      Address.components
   Reference:  Section 2.5.1 of [RFC9553]
   Path:
      $..jscontact_card.[addresses.addr.components[?(@.kind==’postOfficeBox’)].value,
      localizations.*.addresses.addr.components[?(@.kind==’postOfficeBox’)].value]

   jCard property:  "street address" component of adr
   Reference:  Section 6.3.1 of [RFC6350]
   Path:  $..vcardArray[1][?(@[0]==’adr’)][3][2]
   JSContact property:  "name" AddressComponent of Address.components
   Reference:  Section 2.5.1 of [RFC9553]
   Path:
      $..jscontact_card.[addresses.addr.components[?(@.kind==’name’)].value,
      localizations.*.addresses.addr.components[?(@.kind==’name’)].value]
```

jCard property: "locality" component of adr
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]==’adr’)][3][3]
JSContact property: "locality" AddressComponent of Address.components
Reference: Section 2.5.1 of [RFC9553]
Path:
$..jscontact_card.[addresses.addr.components[?(@.kind==’locality’)].value,
localizations.*.addresses.addr.components[?(@.kind==’locality’)].value]

jCard property: "region" component of adr
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]==’adr’)][3][4]
JSContact property: "region" AddressComponent of Address.components
Reference: Section 2.5.1 of [RFC9553]
Path:
$..jscontact_card.[addresses.addr.components[?(@.kind==’region’)].value,
localizations.*.addresses.addr.components[?(@.kind==’region’)].value]

jCard property: "postal code" component of adr
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]==’adr’)][3][5]
JSContact property: "postcode" AddressComponent of Address.components
Reference: Section 2.5.1 of [RFC9553]
Path:
$..jscontact_card.[addresses.addr.components[?(@.kind==’postcode’)].value,
localizations.*.addresses.addr.components[?(@.kind==’postcode’)].value]

jCard property: "country name" component of adr
Reference: Section 6.3.1 of [RFC6350]
Path: $..vcardArray[1][?(@[0]==’adr’)][3][6]
JSContact property: "country" AddressComponent of Address.components
Reference: Section 2.5.1 of [RFC9553]
Path:
$..jscontact_card.[addresses.addr.components[?(@.kind==’country’)].value,
localizations.*.addresses.addr.components[?(@.kind==’country’)].value]

jCard property: "cc" parameter of adr
Reference: Section 3.1 of [RFC8605]
Path: $..vcardArray[1][?(@[0]==’adr’)][1].cc
JSContact property: Address.countryCode
Reference: Section 2.5.1 of [RFC9553]
Path: $..jscontact_card.[addresses.addr.countryCode,
localizations.*.addresses.addr.countryCode]

jCard property: adr
Reference: Section 6.3.1 of [RFC6350]

```
   Path:  $..vcardArray[1][?(@[0]=='adr')]
   JSContact property:  Address
   Reference:  Section 2.5.1 of [RFC9553]
   Path:  $..jscontact_card.[addresses.addr,
      localizations.*.addresses.addr]

   jCard property:  contact-uri
   Reference:  Section 2.1 of [RFC8605]
   Path:  $..vcardArray[1][?(@[0]=='contact-uri')]
   JSContact property:  Link.uri
   Reference:  Section 2.6.3 of [RFC9553]
   Path:  $..jscontact_card.links.contact-uri.uri

   jCard property:  url
   Reference:  Section 6.7.8 of [RFC6350]
   Path:  $..vcardArray[1][?(@[0]=='url')]
   JSContact property:  Link.uri
   Reference:  Section 2.6.3 of [RFC9553]
   Path:  $..jscontact_card.links.url.uri
```

Appendix B. Change Log

B.1. Change from 00 to 01

1. Changed category from "Best Current Practice" to "Standards Track"
2. Replaced the example of Figure 2
3. Changed the title of the "Migration from JCard to JSCard" section to "Transition Considerations"
4. Added "Query Parameters" section.
5. Updated Section 6
6. Updated Section 9
7. Rearranged the description of stage 1 in Section 4.2.2
8. Changed the names of the transition stages 1 and 2
9. Corrected examples
10. Changed the rdapConformance tag "jscard_level_0" to "jscard"
11. Removed the "Best Practices for deprecating a REST API features" section,but added a useful reference.

B.2. Change from 01 to 02

1. Removed the sentence "which cannot be represented using jCard" in Section 1.1.

B.3. Change from 02 to 03

1. Updated section "Conventions Used in This Document".
2. Updated the contact in "IANA Considerations" section.

3. Changed the reference draft-loffredo-calext-jscontact-vcard to draft-ietf-calext-jscontact-vcard.
4. Added reference to RFC8174.
5. Other minor edits.

B.4. Change from 03 to 04

1. Updated the reference draft-dalal-deprecation-header to draft-ietf-httpapi-deprecation-header.

B.5. Initial WG version

1. Ported from draft-loffredo-regext-rdap-jcard-deprecation-04 renamed to draft-ietf-regext-rdap-jscontact-00.

B.6. Change from 00 to 01

1. Updated Section 3 and Figure 2.

B.7. Change from 01 to 02

1. Updated Section 2 and Figure 2.

B.8. Change from 02 to 03

1. Replaced references to obsolete RFC7482 and RFC7483 with RFC9082 and RFC9083.
2. Updated Section 3 and Figure 2.

B.9. Change from 03 to 04

1. Changed the references to Internet Drafts.
2. Added an example showing how localizations are treated in JSContact.
3. Changed the position of section "Goals" in Section 4.2.
4. Added three more implementations to Section 5.
5. Changed the rdapConformance tag "jscard" to "jscard_0"
6. Added clarifications addressing the feedback provided by Jasdip Singh about version -03.
7. Added Section 10.
8. Other minor edits.

B.10. Change from 04 to 05

1. Updated Figure 2 to make it compliant with draft-ietf-jmap-jscontact-09.

B.11. Change from 05 to 06

1. Reviewed the notices presented in stages.

B.12. Change from 06 to 07

1. Corrected the JSON Pointer expressions in Figure 1.
2. Other minor edits.

B.13. Change from 07 to 08

1. Corrected a nit in Figure 1.
2. Removed the reference to draft-ietf-httpapi-deprecation-header.
3. Replaced the "deprecation" link relation type with "related".
4. Moved the references to JSContact drafts to the "Normative References" section.

B.14. Change from 08 to 09

1. Updated the references to JSContact drafts due to the transfer from JMAP to CalExt.

B.15. Change from 09 to 10

1. Updated Figure 2 to make it compliant with draft-ietf-calext-jscontact-02.

B.16. Change from 10 to 11

1. Added Appendix "jCard-JSContact Mapping".

B.17. Change from 11 to 12

1. Renamed the "jscard" property to "jscard_0".
2. Corrected JSONPath expressions in Appendix A.

B.18. Change from 12 to 13

1. Reverted the names of response extension, the rdapConformance tag and extension identifier from "jscard_0" to "jscard".
2. Corrected Figure 1 by removing initial '/' character from JSONPath notations related to localizations.

B.19. Change from 13 to 14

1. Corrected Figure 2 by replacing "online" property with "cryptoKeys" and "links" properties.

B.20. Change from 14 to 15

1. Corrected Figure 1, Figure 2 and Appendix A to make them compliant with draft-ietf-calext-jscontact-06.
2. Removed mention of JSContact CardGroup object.
3. Renamed and changed Section 3.
4. Added Section 6.2.

B.21. Change from 14 to 15

1. Corrected Figure 1, Figure 2 and Appendix A to make them compliant with draft-ietf-calext-jscontact-06.
2. Removed mention of JSContact CardGroup object.
3. Renamed and changed Section 3.
4. Added Section 6.2.

B.22. Change from 15 to 16

1. Replaced JSContact "type" with "kind" in figures.
2. Removed "jCard deprecation" stage and "jcard" query parameter.
3. Renamed "jCard deprecated" stage into "jCard deprecation".
4. Rephrased Section 9.
5. Corrected JSContact examples based on draft-ietf-calext-jscontact-11.
6. Fixed nits.

B.23. Change from 16 to 17

1. Fixed Figure 1, Figure 2, Section 6.3 and Appendix A to make them compliant with draft-ietf-calext-jscontact-16 and purge them of the JSContact properties which are hardly used in RDAP.
2. Rearranged Section 3.
3. Added other mapping correspondences to Appendix A.

B.24. Change from 17 to 18

1. Updated references.
2. Updated Gavin Brown’s contact info.
3. Removed from the notices field the optional link documenting the transition procedure.
4. Added text to Section 2.
5. Added a reference to RFC9553 section defining the Id type.
6. Tagged all BCP14 keywords.
7. Changed the example in Figure 2.
8. Added Section 3.1.6.
9. Changed Section 4.2.2.4 by removing the sentence about the duration of the transition process.

B.25. Change from 18 to 19

1. Turned the draft status into "Experimental".
2. Changed "@version" to "version" in the examples.
3. Rearranged the content of the "Query Parameters" section and renamed it into "Request for JSContact".
4. Added Section 3.1.2.
5. Added Section 3.2.
6. Rearranged Section 4.1.
7. Rearranged Figure 3.
8. Removed from Section 4.2.2.3 the "FOR DISCUSSION" item.
9. Added to Section 4.2.2.3 a notice signaling the jCard deprecation.
10. Changed the fixed literal to redact the URI in Section 9.
11. Removed an informative reference and the first paragraph of Section 4.2 including it.

B.26. Change from 19 to 20

1. Renamed the "jscard" extension identifier into "jscontact".
2. Renamed the "jscard" property into "jscontact_card".
3. Replaced any instance of the verbatim string "JSCard" with "JSContact".
4. Rephrased Section 3.1.3 by removing "1.0" as a mandatory value for the JSContact version property in RDAP and adding the value to the "RDAP JSON Values" registry.
5. Corrected link in Figure 3.
6. Minor editorial changes.

B.27. Change from 20 to 21

1. Added the "Experimental Status" section.
2. Changed the controller of IANA registies from IESG to IETF.
3. Replaced "rdap-x" with "rdap" in the rdap-x media type.
4. Minor editorial changes.

B.28. Change from 21 to 22

1. Changed JSContact version "1.0" to "2.0".
2. Removed any reference to the JSContact uid property.
3. Rephrased text of Section 9.
4. Removed JSContact properties from examples that are not included in the RDAP profile for JSContact.
5. Added a section about the registration of the JSContact profile for RDAP.
6. Added ips and autnums to the searchable resource types of Section 6.3.
7. Minor editorial changes.

B.29. Change from 22 to 23

1. Revised the "Experimental Status" section.
2. Revised the registration of the JSContact profile for RDAP. Added a section that formally lists the profile properties.
3. Added the handling and registration of noJcard extension identifier.
4. Changed the notice used in the transaction process by changing the "title" property into "type" and registered the values in the "RDAP JSON Values" registry.
5. Changed the naming scheme for map keys other than those predefined and clarified that those keys should be stable.
6. Revised the "Security Considerations" section and renamed it into "Privacy Considerations". Added a "Security Considerations" section presenting no substantial implication on scurity.
7. Updated the "Acknowledgements" section.
8. Minor editorial changes.

B.30. Change from 23 to 24

1. Changed the status to "Standards Track".
2. Removed the "Experimental Status" section and arranged references accordingly.
3. Corrected example in Figure 3.
4. Added Section 5.5.
5. Added references to IANA registries.

B.31. Change from 24 to 25

1. Added "country" to the allowable values listed in Section 3.1.8 for the "kind" property of the AddressComponent object.
2. Aligned content of Section 3.1.2 to version -06 of [I-D.ietf-regext-rdap-versioning].
3. Added Section 7 for compliance with [I-D.ietf-opsawg-rfc5706bis]
4. Corrected nits.

Authors' Addresses

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

Gavin Brown
ICANN
12025 Waterfront Drive, Suite 300
Los Angeles, CA 90292
United States of America
Email: gavin.brown@icann.org
URI: https://www.icann.org/