Network Working Group
Internet-Draft
Expires: September 1, 2015

M. Amundsen
CA Technologies, Inc.
L. Richardson

M. Foster
Apiary

February 28, 2015

Application-Level Profile Semantics (ALPS)
draft-amundsen-richardson-foster-alps-01

Abstract

This document describes ALPS, a data format for defining simple descriptions of application-level semantics, similar in complexity to HTML microformats. An ALPS document can be used as a profile to explain the application semantics of a document with an application-agnostic media type (such as HTML, HAL, Collection+JSON, Siren, etc.). This increases the reusability of profile documents across media types.

Editorial Note (To be removed by RFC Editor)

Distribution of this document is unlimited. Comments should be sent to the IETF Media-Types mailing list (see [1]).

Status of This Memo

Table of Contents

1. Introduction 3
1.1. Notational Conventions 3
1.2. Motivation 4
1.2.1. Describing Domain-Specific Semantics 4
1.2.2. ALPS-based Server Implementations 4
1.2.3. ALPS-based Client Implementations 4
1.3. A Simple ALPS Example 5
1.4. Identifying an ALPS Document 9
2. ALPS Documents 10
2.1. Compliance 10
2.2. ALPS Document Properties 10
2.2.1. 'alps' 10
2.2.2. 'doc' 10
2.2.3. 'descriptor' 11
2.2.4. 'ext' 13
2.2.5. 'format' 13
2.2.6. 'href' 14
2.2.7. 'id' 14
2.2.8. 'link' 16
2.2.9. 'name' 16
2.2.10. 'rel' 17
2.2.11. 'rt' 17
2.2.12. 'type' 17
2.2.13. 'value' 18
2.2.14. 'version' 18
2.3. ALPS Representations 18
2.3.1. Sample HTML 18
2.3.2. XML Representation Example 19
2.3.3. JSON Representation Example 19
3. Applying ALPS documents to Existing Media Types 21
3.1. Linking to ALPS Documents 22

4. IANA Considerations 22
4.1. application/alps+xml 22
4.2. application/alps+json 24
5. Internationalization Considerations 25
6. Acknowledgements 25
7. References 25
7.1. Normative References 25
7.2. Informative References 25
Appendix A. Frequently Asked Questions 26
A.1. Why are there no URLs in ALPS? 26
A.2. Why is there no workflow component in the ALPS specification? 26
A.3. Why is there no way to indicate ranges for semantic descriptors? 26

1. Introduction

This document describes ALPS, a media type for defining simple descriptions of application-level semantics, similar in complexity to HTML microformats. These descriptions contain both human-readable and machine-readable explanations of the semantics. An ALPS document can be used as a profile to explain the application semantics of a document with an application-agnostic media type (such as HTML, HAL, Collection+JSON, Siren. etc.).

This document identifies a registry for ALPS documents, (The ALPS Profile Registry or APR). The details of this registry, its goals, and operations are covered in a separate document (TBD).

This document also identifies a process for authoring, publishing, and sharing normative human-readable instructions on applying an ALPS document as a profile to responses of a given media type. For example, a document that describes how to apply the semantics of an ALPS profile to an HTML document.

This document registers two media-type identifiers with the IANA: 'application/alps+xml' ('ALPS+XML') and 'application/alps+json' ('ALPS+JSON').

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in[RFC2119].

1.2. Motivation

When implementing a hypermedia client/server application using a general media type (HTML, Atom, Collection+JSON, etc.), client and server instances need to share an understanding of domain-specific information such as data element names, link relation values, and state transfer parameters. This information is directly related to the application being implemented (e.g. accounting, contact management, etc.) rather than the media type used in the representations.

1.2.1. Describing Domain-Specific Semantics

Instead of creating and registering an entirely new media type (i.e. 'application/accounting'), representation authors can create an ALPS document that describes a 'profile' of the target domain; one that explains the vital domain-specific semantic descriptors and state transitions. This profile can then be consistently applied to a wide range of media types by server implementors and successfully consumed by client applications. The focus on defining application-level semantics, independent of transfer protocol or media type, makes it possible to serve application-specific representations using an application-agnostic media type.

1.2.2. ALPS-based Server Implementations

Server implementors can use ALPS documents as a basis for building domain-specific solutions without having to create their own custom media type or re-invent the vocabulary and transition set for a common domain (e.g. accounting, microblogging, etc.). Using a preexisting ALPS profile as a guide, servers can map internal data to commonly-understood semantic descriptors and state transitions, increasing the likelihood that existing client applications (those who share the same understanding of the ALPS document) will be able to successfully interact with that server.

1.2.3. ALPS-based Client Implementations

Armed with a document's ALPS profile, client applications can associate the ALPS descriptor 'id' and/or 'name' attribute values with the appropriate elements within the document. Client applications can 'code for the profile' and better adjust to detailed changes to the response layout, or even the wholesale replacement of one media type with another.

1.3. A Simple ALPS Example

Below is an ALPS document that describes elements of a simple request/response interaction in a contact management application. The profile defines a semantic descriptor called 'contact', and three subordinate descriptors ('fullName', 'email', and 'phone').

The ALPS document also defines a single, safe state transition, to be represented by a hypermedia control (e.g. HTML.GET form) with the 'id' value of 'collection.' This hypermedia control has one input value ('nameSearch'). When executed, the response will contain one or more 'contact' type items.

```
 <alps version="1.0">
  <doc format="text">A contact list.</doc>

  <!-- a hypermedia control for returning contacts -->
  <descriptor id="collection" type="safe" rt="contact">
    <doc>
      A simple link/form for getting a list of contacts.
    </doc>
    <descriptor id="nameSearch" type="semantic">
      <doc>Input for a search form.</doc>
    </descriptor>
  </descriptor>

  <!--  a contact: one or more of these may be returned -->
  <descriptor id="contact" type="semantic">
    <descriptor id="item" type="safe">
      <doc>A link to an individual contact.</doc>
    </descriptor>
    <descriptor id="fullName" type="semantic" />
    <descriptor id="email"    type="semantic" />
    <descriptor id="phone"    type="semantic" />
  </descriptor>
</alps>
```

ALPS Contact Profile document

Implementing the ALPS profile above requires implementing the descriptors defined by the ALPS document. In this case, there are two 'top level' descriptors: the safe state transition ('collection') and the semantic descriptor 'contact'. Below is a single HTML document that shows both these elements in a representation.

```
<html>
  <head>
    <link href="http://alps.io/profiles/contact"
```

```
        rel="profile" />
      <link href="http://alps.io/profiles/contact#contact"
        rel="type" />
    </head>
    <body>
      <form class="collection"
        method="get"
        action="http://example.org/contacts/">
        <label>Name:</label>
        <input name="nameSearch" value="" />
        <input type="submit" value="Search" />
      </form>

      <table>
        <tr class="contact">
          <td>
            <a href="http://example.org/contacts/1"
              rel="item">
              <span class="fullName">Ann Arbuckle</span>
            </a>
          </td>
          <td>
            <span class="email">aa@example.org</span>
          </td>
          <td>
            <span class="phone">123.456.7890</span>
          </td>
        </tr>

        <tr>
          <td>
            <a href="http://example.org/contacts/100"
              rel="item">
              <span class="fullName">Zelda Zackney</span>
            </a>
          </td>
          <td>
            <span class="email">zz@example.org</span>
          </td>
          <td>
            <span class="phone">098.765.4321</span>
          </td>
        </tr>
      </table>
    </body>
  </html>
```

HTML ALPS Contact Representation

HTML representations implement most ALPS elements using HTML's 'class' attribute. The 'collection' ID has become the CSS class of an HTML form's submit button. The 'contact' ID has become the CSS class of the TR elements in an HTML table. The subordinate descriptors 'fullname','email', and 'phone' are rendered as the TD elements of each TR.

This HAL document uses the same profile to express the same application-level semantics as the HTML document.

```
<resource href="http://example.org/contacts/">
  <link href="http://alps.io/profiles/contacts#contact"
    rel="type" />
  <link rel="collection"
    href="http://example.org/contacts/{?nameSearch}"
    templated="true" />
  <resource rel="item" href="http://example.org/contacts/1">
    <link href="http://alps.io/profiles/contacts#contact"
      rel="type" />
    <fullName>Ann Arbuckle</fullName>
    <email>aa@example.org</email>
    <phone>123.456.7890</phone>
  </resource>
  <resource rel="item" href="http://example.org/contacts/100">
    <link href="http://alps.io/profiles/contacts#contact"
      rel="type" />
    <fullName>Zelda Zackney</fullName>
    <email>zz@example.org</email>
    <phone>987.664.3210</phone>
  </resource>
</resource>
```

HAL XML Contacts Representation

In a HAL representation, all state transitions ('collection' and 'item', in this case) are represented as link relations. All data descriptors ('fullName', 'email', and 'phone') are represented as XML tags named after the descriptors.

This Collection+JSON document uses the ALPS profile to express the same application-level semantics as the HTML and HAL documents.

```
{
  "collection" : {
    "version" : "1.0",
    "href" : "http://example.org/contacts/",

    "links" : [
```

```
        {
          "rel" : "profile",
          "href" : "http://alps.io/profiles/contacts"
        },
        {
          "rel" : "type",
          "href" : "http://alps.io/profiles/contacts#contact"
        }
      ],

      "queries" : [
        {
          "rel" : "collection",
          "rt" : "contact",
          "href" : "http://example.org/contacts/",
          "data" : [
            {
              "name" : "nameSearch",
              "value" : "",
              "prompt" :  "Search Name"
            }
          ]
        }
      ],

      "items" : [
        {
          "href" : "http://example.org/contacts/1",
          "rel" : "item",
          "rt" : "contact",
          "data" : [
            {"name" : "fullName", "value" : "Ann Arbuckle"},
            {"name" : "email", "value" : "aa@example.org"},
            {"name" : "phone", "value" : "123.456.7890"}
          ],
          "links" : [
            {
              "rel" : "type",
              "href" : "http://alps.io/profiles/contacts#contact"
            }
          ]
        },
        {
          "href" : "http://example.org/contacts/100",
          "rel" : "item",
          "rt" : "contact",
          "data" : [
            {
```

```
              "name" : "fullName",
              "value" : "Zelda Zackney"
            },
            {
              "name" : "email",
              "value" : "zz@example.org"
            },
            {
              "name" : "phone",
              "value" : "987.654.3210"
            }
          ],
          "links" : [
            {
              "rel" : "type",
              "href" : "http://alps.io/profiles/contacts#contact"
            }
          ]
        }
      ]
    }
  }
```

Collection+JSON Contacts Representation

The descriptor 'collection' has become the link relation associated with a Collection+JSON query. The descriptors 'fullName', 'email', and 'phone' have become the names of key-value pairs in the items in a Collection+JSON collection.

1.4. Identifying an ALPS Document

An ALPS vocabulary is identified by a unique URL. This URL SHOULD be assumed to be dereferencable. All ALPS URLs MUST be unique and all ALPS documents intended for public consumption SHOULD be registered in an ALPS Registry [TK: add text on where/how to find registries -mamund].

In order to reduce load on servers responding to ALPS document requests, it is RECOMMENDED that servers use cache control directives that instruct client apps to locally cache the results. Clients making these ALPS document requests SHOULD honor the server's caching directives.

2. ALPS Documents

An ALPS document contains a machine-readable collection of identifying strings and their human-readable explanations. An ALPS document can be represented in either XML or JSON format. This section identifies the general elements and properties of an ALPS document, their meaning, and their use, independent of how the document is represented. Section 2.3 provides specific details on constructing a valid ALPS document in XML and in JSON format.

2.1. Compliance

An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements. An implementation that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements is said to be 'unconditionally compliant'; one that satisfies all the MUST level requirements but not all the SHOULD level requirements is said to be 'conditionally compliant.'

2.2. ALPS Document Properties

The ALPS media type defines a small set of properties. These properties appear in both the XML and JSON formats. Below is a list of the properties that can appear in an ALPS document.

2.2.1. 'alps'

Indicates the root of the ALPS document. This property is REQUIRED, and it SHOULD have one or more 'descriptor' child properties.

Examples:

XML: <alps>...</alps>

JSON: { "alps" : ... }

2.2.2. 'doc'

A text field that contains free-form, usually human-readable, text. The 'doc' element MAY have two properties: 'href' and 'format'. If the 'href' property appears it SHOULD contain a dereferencable URL that points to human-readable text. If the 'format' property appears it SHOULD contain one of the following values: 'text', 'html', 'asciidoc', or 'markdown'. Any program processing 'doc' elements SHOULD honor the 'format' directive and parse/render the content appropriately. If the value in the 'format' property is not recognized and/or supported, the processing program MUST treat the

content as plain text. If no 'format' property is present, the content SHOULD be treated as plain text.

XML: <doc format="html"> <h1>Date of Birth</h1> ... </doc>

JSON: { "doc" : { "format" : "text" : "value" : "Date of Birth ..." } }

A 'doc' element SHOULD appear as a child of 'descriptor'. When present, it describes the meaning and use of the related 'descriptor'.

XML: <descriptor ... > <doc>...</doc> </descriptor>

JSON: { "descriptor" : { { "doc" : { "value" : "..." } ... } }

The 'doc' element MAY appear as a child of 'alps'. When present, it describes the purpose of the ALPS document as a whole.

XML: <alps> <doc>...</doc> ... </apls>

JSON: { "alps : "doc" : { "value" : "..." }, ... }

2.2.3. 'descriptor'

A 'descriptor' element defines the semantics of specific data elements or state transitions that MAY exist in an associated representation.

One or more 'descriptor' elements SHOULD appear as children of 'alps'. It may also appear as a child of itself; that is, the 'descriptor' property may be nested.

The 'descriptor' property SHOULD have either an 'id' or 'href' attribute. It MAY have both. Additionally, the 'descriptor' MAY have any of the following attributes:

1. 'doc'

2. 'ext'

3. 'name'

4. 'type'

If present, the 'href' property MUST be a dereferenceable URL, that points to another 'descriptor' either within the current ALPS document or in another ALPS document.

If 'descriptor' has an 'href' attribute, then 'descriptor' is inheriting all the attributes and sub-properties of the descriptor pointed to by 'href'. When 'descriptor' has a property defined locally, that property value takes precedence over any inherited property value. Since there is no limit to the nesting of elements -- even ones linked remotely -- it is important to process 'all descriptor' chains starting from the bottom to make sure you have collected all the available properties and have established the correct value for each of them.

If 'descriptor' is declared at the top level of an ALPS document, then a client SHOULD assume that 'descriptor' can appear anywhere in a runtime message.

If 'descriptor' is nested, i.e. declared as a child of another descriptor, then:

1. A client SHOULD assume them to appear in any sibling 'descriptor' element and recursively in their child descriptors.

2. A client SHOULD NOT assume that it can appear anywhere outside of parent descriptor, unless it was explicitly referenced by another descriptor in 'href' attribute. In that case the same rules are applied to 'descriptor' containing 'href' attribute.

2.2.3.1. 'Descriptors and Link Relation Types'

When a representation is generated that includes state transitions, valid values for link relation types are:

1. A registered IANA link relation type (e.g. rel="edit", a short string).

2. An extension link relation type as defined by [RFC5988] whose value is the fully-qualified URI of an associated document describing the relation type. This includes URI fragment identifiers of ALPS descriptors (e.g. rel="http://alps.io/profiles/item#purchased-by", a URI) per the conventions of Section 2.2.7.2.

3. The 'id' property of a state transition descriptor of an associated ALPS document (e.g. rel="purchased-by", a short string) per the conventions of section Section 2.2.7.1 and Section 2.2.7.3 if the representation includes an ALPS profile.

2.2.4. 'ext'

The 'ext' element can be used to extend the ALPS document with author-specific information. It provides a way to customize ALPS documents with additional properties not covered in this specification. This is an OPTIONAL element.

The 'ext' element has the following properties.

1. 'id'

2. 'href'

3. 'value'

The 'id' property is REQUIRED. The 'href' is RECOMMENDED and it SHOULD point to documentation that explains the use and meaning of this 'ext' element. The 'value' property is OPTIONAL. The content is undetermined; its meaning and use SHOULD be explained by the document found by de-referencing the 'href' property.

Examples:

XML: <ext id="directions" href="http://alps.io/ext/directions" value="north south east west" >

JSON: { "ext" : { "id" : "directions", "href" : "http://alps.io/ext/directions", value="north south east west" } }

The 'ext' element MAY appear as a child of the following elements:

1. 'alps'

2. 'descriptor'

Since the 'ext' element has no specific meaning within this specification, it MUST be ignored by any application that does not understand its meaning.

2.2.5. 'format'

Indicates how the text content should be parsed and/or rendered. This specification identifies a range of possible values for 'format':

- 'text', for plain text, MUST be supported.

- 'html', for HTML, SHOULD be supported.

- 'asciidoc', for AsciiDoc, MAY be supported.

- 'markdown', per The text/markdown Media Type [I-D.ietf-appsawg-text-markdown], MAY be supported.

Any other values for this attribute are undefined and SHOULD be treated as plain text. If the program does not recognize the value of the 'format' property and/or the 'format' property is missing, the content SHOULD be treated as plain text.

This property MAY appear as an attribute of the 'doc' element.

2.2.6. 'href'

Contains a resolvable URL.

When it appears as an attribute of a 'descriptor', 'href' points to another 'descriptor' either within the existing ALPS document as a fragment or in another ALPS document as an absolute URL. The URL MUST contain a fragment per Section 2.2.7.2 referencing the related 'descriptor'.

When it appears as an attribute of 'ext', 'href' points to an external document which provides the defintion of the extension.

When it appears as an attribute of 'link', 'href' points to an external document whose relationship to the current document or 'descriptor' is described by the associated 'rel' property.

When it appears as an attribute of 'doc', 'href' points to a document that contains human-readable text that describes the associated 'descriptor' or ALPS document.

2.2.7. 'id'

A document-wide unique identifier for the related element. This SHOULD appear as an attribute of a 'descriptor'.

The value of this attribute MAY be used as an identifier in the related runtime hypermedia representation. In the example below the ALPS descriptor with an 'id' of 'q' is used to identify an HTML input element:

'id' in ALPS... <descriptor id="q" type="semantic" />

...becomes the 'class' in HTML <input class="q" type="text" value="" />

It should be noted that the exact mapping from ALPS elements (e.g. 'id') to elements within a particular media type (HTML, Collection+JSON, etc.) is covered in separate documents (to be specified).

2.2.7.1. ALPS 'id' and 'name' Properties

In some cases, media types support non-unique identifiers (e.g. HTML's 'name' property) or will allow the same identifier value for multiple elements in the same representation (e.g. <div id="search" ... /> and <input type="submit" class="search" .../> and <input name="search" ... />). In those cases, translating that representation into ALPS documents could result in multiple 'id' properties with the same value.

To avoid this, ALPS document designers can add the 'name' property to a 'descriptor' to hold the common value ('search') while still using the 'id' property to hold a document-wide unique value. For example:

```
<!-- HTML -->
<div id="search">
  <form action="..." method="get">
    <input name="search" value="..." type="text" />
    <input type="submit" class="search" />
  </form>
</div>
```

HTML Representation of a Search Transition

```
<!-- ALPS -->
<descriptor id="search-block" type="semantic" name="search">
  <descriptor id="search-form" type="safe" name="search">
    <descriptor id="search-data" type="semantic" name="search" />
  </descriptor>
</descriptor>
```

ALPS Description of the same Search Transition

2.2.7.2. Fragment Identifiers and 'id'

When applied to an ALPS document, a URI fragment identifier points to the 'descriptor' whose 'id' is the value of the fragment. For example, the fragment identifier 'customer' in the URI http://example.com/my-alps-document#customer refers to an ALPS 'descriptor' with 'id' set to 'customer'.

A relative URL with a fragment identifier within an ALPS document (e.g. href="#customer") refers to a local 'descriptor' within the document containing the reference.

The complete URI to an ALPS 'descriptor' (including the fragment) forms an 'abstract semantic type' identifier. This is a resolvable URI (URL) that can be used to indicate the type of a resource; for instance, it can be used as the value of the IANA-registered relation type 'type'.

2.2.7.3. Link Relation Values and 'id' or 'name'

Since a state transition 'descriptor' may define a relation type value, it is important to avoid creating conflicts with existing IANA-registered values. If the resulting link relation type is the same as a registered relation type, the descriptor MUST not change the meaning of the IANA relation type.

Further, since the 'id' of a 'descriptor' may define a link relation value per Section 2.2.3.1, if a conflict exists in defining such a descriptor's document-wide unique 'id' with another 'descriptor', the conflicting 'descriptor' MUST define a unique 'id' and MAY specify a 'name' property to resolve the conflict.

If it is unclear whether a registered link relation type in a representation document refers to a relation registered with IANA or a relation registered in an ALPS profile, the semantics of that link are undefined.

2.2.8. 'link'

An element that identifies a link between the current ALPS element and some other (possibly external) resource. MAY be a child element of the 'alps' and the 'descriptor' elements.

The 'link' element MUST define the two attributes 'href' and 'rel'.

2.2.9. 'name'

Indicates the name of the 'descriptor' as found in generic representations. It MAY appear as a property of 'descriptor'.

This is used when the name of the 'descriptor' is used as an 'id' value elsewhere in the ALPS document. For instance, if a single ALPS document defines a semantic descriptor (data element) called 'customer' and a safe descriptor (transition element) also called 'customer', they cannot both have 'id="customer"' in the ALPS

document. One of them needs to have some other 'id', and to set 'name="customer"'.

The use of the 'name' property usually indicates an ambiguity in the application semantics. Thus, it SHOULD only be used when creating an ALPS profile that describes an existing design.

2.2.10. 'rel'

Contains a [RFC5988] approved value: either an extension relation type (a URI) or a registered relation type (a short string).

Appears as a property of'link'.

2.2.11. 'rt'

Indicates the resource type that will be returned when executing the specified network request. The 'rt' attribute SHOULD appear only on a 'descriptor' with a 'type' value of 'safe', 'unsafe', or 'idempotent.'

The 'rt' attribute is OPTIONAL and is an opaque string and MAY match the 'id' of an existing'descriptor'.

2.2.12. 'type'

Indicates the type of hypermedia control to which the element is applied within the resulting representation. This SHOULD appear for each 'descriptor' element. The four valid values are:

'semantic' A state element (e.g. HTML.SPAN, HTML.INPUT, etc.).

'safe' A hypermedia control that triggers a safe, idempotent state transition (e.g. HTTP.GET or HTTP.HEAD).

'idempotent' A hypermedia control that triggers an unsafe, idempotent state transition (e.g. HTTP.PUT or HTTP.DELETE).

'unsafe' A hypermedia control that triggers an unsafe, non-idempotent state transition (e.g. HTTP.POST).

If no 'type' attribute is associated with the element, then 'type="semantic"' is implied.

2.2.13. 'value'

Contains a string value. It MAY appear as an attribute of the 'doc' and the 'ext' elements.

2.2.14. 'version'

Indicates the version of the ALPS specification used in the document. This SHOULD appear as a property of the 'alps' element. Currently the only valid value is '1.0'. If no value appears, then 'version="1.0"' is implied.

2.3. ALPS Representations

An ALPS document may be represented in either XML or JSON format. This section contains notes on how the ALPS elements and attributes appear in each format, along with examples to guide ALPS document authors.

2.3.1. Sample HTML

Below is a simple HTML document that contains a handful of semantic descriptors and transition instructions. This document was generated from the XML and JSON ALPS documents that follow. Use this HTML document as a guide when evaluating the XML and JSON examples.

```
<!-- sample HTML document -->
<html>
  <head>
    <link rel="profile" href="http://alps.io/documents/search" />
  </head>
  <body>
    <form class="search" action="..." method="get">
      <input type="text" name="search" value="..." />
        <select name="resultType">
          <option value="summary" />
          <option value="detailed" />
        </select>
      <input type="submit" />
    </form>
  </body>
</html>
```

HTML Sample

2.3.2. XML Representation Example

In the XML version of an ALPS document, the following ALPS properties always appear as XML elements: 'alps', 'doc', 'descriptor', and 'ext'. All other ALPS properties appear as XML attributes.

2.3.2.1. Complete XML Representation

Below is an example of an application/alps+xml representation.

```
<?xml version="1.0"?>
<alps version="1.0">
  <doc href="http://example.org/samples/full/doc.html" />

  <descriptor id="search" type="safe">
    <doc format="text">A search form with two inputs.</doc>
    <descriptor href="#resultType" />
    <descriptor id="value" name="search" type="semantic">
      <doc>input for search</doc>
    </descriptor>
  </descriptor>

  <descriptor id="resultType" type="semantic">
    <doc>results format</doc>
    <ext
      href="http://alps.io/ext/range"
      value="summary,detail" />
  </descriptor>
</alps>
```

Complete XML Representation

2.3.3. JSON Representation Example

When representing ALPS documents in JSON format, the 'descriptor' and 'ext' properties are always expressed as arrays of anonymous objects - even when there is only one member in the array.

For example:

```
"descriptor" : [
  {
    "id" : "value",
    "name" : "search",
    "type" : "descriptor",
    "doc" : { "value" : "input for search" }
  },
  { "href" : "#resultType" }
]
```

Arrays in ALPS+JSON

The 'doc' property is always expressed as a named object.

For example:

```
{
  "doc" : {
    "format" : "text",
    "value" : "Rules are important"
  }
}
```

Descriptions in ALPS+JSON

2.3.3.1. Complete JSON Representation

Below is a example of the application/alps+json representation of an ALPS document.

```
{
  "alps" : {
    "version" : "1.0",
    "doc" : {
      "href" : "http://example.org/samples/full/doc.html"
    },
    "descriptor" : [
      {
        "id" : "search",
        "type" : "safe",
        "doc" : {"value" :
          "A search form with a two inputs"
        },
        "descriptor" : [
          {
            "id" : "value",
            "name" : "search",
            "type" : "descriptor",
            "doc" : { "value" : "input for search" }
          },
          { "href" : "#resultType" }
        ]
      },
      {
        "id" : "resultType",
        "type" : "descriptor",
        "description" : {"value" : "results format"},
        "ext" : [
          {
            "href" : "http://alps.io/ext/range",
            "value" : "summary,detail"
          }
        ]
      }
    ]
  }
}
```

Complete ALPS+JSON Representation

3. Applying ALPS documents to Existing Media Types

An ALPS document can be applied to many existing media types as long as there exists an agreed mapping between ALPS and the target media type. Section 1.3 gave some informative examples of this. Normative, up-to-date guidance on applying ALPS documents to existing media types are available at the official ALPS Web site at

(http://alps.io/docs/mapping). [TK : this page does not yet exist. -mamund]

Not all media types can faithfully represent all ALPS descriptors. For instance, the 'application/json' media type has no standard way of representing hyperlinks. The [details of how to apply ALPS to such a media type will necesarily be incomplete, and it will not be possible to represent some aspects of an ALPS profile in documents in that media type.

3.1. Linking to ALPS Documents

To indicate that an ALPS profile describes the semantics of some representation document, the representation document SHOULD be linked to the ALPS document. The 'profile' link relation [RFC6906] MUST be used when creating this link. If the media type of the representation document has no native ability to link to other resources, or no ability to express link relations, the HTTP header 'Link' [RFC5988] MAY be used to connect the representation document and the ALPS profile. If the media type of the representation document defines a parameter for linking the document to a profile, that parameter MAY be used to connect the representation document and the ALPS profile.

A single representation document may be described by more than one ALPS profile. If two ALPS profiles give conflicting semantics for the same element, the document linked to earlier in the representation SHOULD take precedence. A profile linked to using the 'Link' header takes precedence over a profile linked to within the representation document itself. A profile linked to using a media type parameter takes precedence over a profile linked to using the 'Link' header and a profile linked to within the representation document itself.

4. IANA Considerations

This specification establishes two media types: 'application/alps+xml' and 'application/alps+json'

4.1. application/alps+xml

Type name: application

Subtype name: alps+xml

Required parameters: None

Optional parameters:

charset This parameter has identical semantics to the charset parameter of the 'application/xml' media type as specified in[RFC3023].

profile A whitespace-separated list of IRIs identifying specific constraints or conventions that apply to an ALPS document. A profile must not change the semantics of the resource representation when processed without profile knowledge, so that clients both with and without knowledge of a profiled resource can safely use the same representation. The profile parameter may also be used by clients to express their preferences in the content negotiation process. It is recommended that profile IRIs are dereferenceable and provide useful documentation at that IRI.

Encoding considerations:

binary Same as encoding considerations of application/xml as specified in[RFC3023].

Security considerations: This format shares security issues common to all XML content types. It does not provide executable content. Information contained in ALPS documents do not require privacy or integrity services.

Interoperability considerations: ALPS is not described by a DTD and applies only the well-formedness rules of XML. It should only be parsed by a non-validating parser.

Published specification: This Document

Applications that use this media type: Various

Additional information:

magic number(s): none

file extensions: .xml

macintosh type file code: TEXT

object idenfiers: none

person to contact for further information:

Name: Mike Amundsen

Email: mca@amundsen.com

Intended usage: Common

Author/change controller: Mike Amundsen

4.2. application/alps+json

Type name: application

Subtype name: alps+json

Required parameters: None

Optional parameters:

profile A whitespace-separated list of IRIs identifying specific constraints or conventions that apply to an ALPS document. A profile must not change the semantics of the resource representation when processed without profile knowledge, so that clients both with and without knowledge of a profiled resource can safely use the same representation. The profile parameter may also be used by clients to express their preferences in the content negotiation process. It is recommended that profile IRIs are dereferenceable and provide useful documentation at that IRI.

Encoding considerations: binary

Security considerations: This media type shares security issues common to all JSON content types. See [RFC4627] Section #6 for additional information. ALPS+JSON does not provide executable content. Information contained in ALPS+JSON documents do not require privacy or integrity services.

Interoperability considerations: None

Published specification: This Document

Applications that use this media type: Various

Additional information:

magic number(s): none

file extensions: .json

macintosh type file code: TEXT

object idenfiers: none

person to contact for further information:

Name: Mike Amundsen

Email: mca@amundsen.com

Intended usage: Common

Author/change controller: Mike Amundsen

5. Internationalization Considerations

[TK]

[[CREF1: insert text (consider rfc 5987)]]

6. Acknowledgements

The following people made contributions to this specification:

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.

[RFC6906] Wilde, E., "The 'profile' Link Relation Type", RFC 6906, March 2013.

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, July 2014.

7.2. Informative References

[I-D.ietf-appsawg-text-markdown] Leonard, S., "The text/markdown Media Type", draft-ietf-appsawg-text-markdown-05 (work in progress), December 2014.

Appendix A. Frequently Asked Questions

A.1. Why are there no URLs in ALPS?

ALPS is meant to describe a service in a universal way. The same ALPS description document can be used by many ALPS-compliant servers. Since each service implementation is in charge of their own URL space, ALPS descriptions do not include URLs. See URI Design and Ownership [RFC7320] for more on this principle.

When implementing ALPS-compliant servers, implementors are free to use any URL design they wish. All that is required is that implementors use the same ALPS profile descriptor 'id' and 'name' properties in the representations. When implementing ALPS-compliant client applications, the URLs will be supplied at runtime by the server represetentations. Client apps only need to recognize the descriptor 'id' and 'name' values from the referenced ALPS profile document.

A.2. Why is there no workflow component in the ALPS specification?

ALPS is not designed to describe workflows or execution paths for a service. Instead, ALPS is designed to describe a shared set of data and actions elements that server MAY implement in order to create a service. Each action descriptor (where the descriptor's type property is set to 'safe', 'unsafe', or 'idemponent') SHOULD describe a state transition that a ALPS-compliant client application can invoke when it is available. Servers are free to implement the transitions they find useful and to arrange them in any order they wish. ALPS-compliant client applications SHOULD be able to recognize these descriptors when they appear and are free to act upon them directly, render them for humans to invoke, or ignore/hide them completely.

A.3. Why is there no way to indicate ranges for semantic descriptors?

For most all service implementations, there are cases where it would be helpful to document a range of possible values for a semantic element. For example, when implementing the descriptor {"id":"size", ...}, one service might want to indicate the list of supported values such as: 'small', 'meduim', 'large', etc. However, another service might have a very different list of possible values such as 'standard', 'oversized', 'undersized', etc. And there may be a service that only supports a single value here and will always supply it ('onesize').

Since ALPS is meant to provide a single description that can be used by multiple services, establishing ranges within the ALPS description

is considered over-constraining service implementations. Services are free to supply this information within representations at run time. But including them in the global ALPS profile is discouraged.

Authors' Addresses

Mike Amundsen
CA Technologies, Inc.

EMail: mca@amundsen.com
URI: http://amundsen.com

Leonard Richardson

EMail: leonardr@segfault.org
URI: http://crummy.com

Mark W. Foster
Apiary

EMail: mwf@fosrias.com