Activity Streams (http://activitystrea.ms)
Internet-Draft
Intended status: Standards Track
Expires: May 08, 2014

J. Snell, Ed.
IBM

November 04, 2013

JSON Activity Streams 2.0
draft-snell-activitystreams-05

Abstract

This specification details a model for representing potential and completed activities using the JSON format.

Author's Note

This draft is heavily influenced by the original JSON Activity Streams 1.0 specification that was originally co-authored by Martin Atkins, Will Norris, Chris Messina, Monica Wilkinson, Rob Dolin and James Snell. The author is very thankful for their significant contributions and gladly stands on their shoulders. Some portions of the original text of Activity Streams 1.0 are used in this document.

The Activity Streams 1.0 and 2.0 specifications are works produced by the Activity Streams Working Group (http://activitystrea.ms/) operating independently of the IETF. Discussion and feedback about this specification is invited and should be directed to the Activity Streams Mailing List (see https://groups.google.com/forum/#!forum/activity-streams).

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 http://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 May 08, 2014.

Table of Contents

1. Introduction

In the most basic sense, an "activity" is a semantic description of potential or completed actions. In the former case, the activity expresses what can be done with a particular object, while in the latter case, it expresses what has already been done.

It is the goal of this specification to provide a JSON-based syntax that is sufficient to express metadata about activities in a rich, human-friendly, machine-processable and extensible manner. This may include constructing natural-language descriptions or visual representations about the activity, associating actionable information with various types of objects, communicating or recording activity logs, or delegation of potential actions to other applications.

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].

1.1. Relationship to JSON Activity Streams 1.0

The JSON Activity Streams 1.0 [activitystreams-1.0] specification was published in May of 2011 and provided a baseline extensible syntax for the expression of completed activities. This specification builds upon that initial foundation by incorporating lessons learned through extensive implementation, community feedback and related work being performed in other standards development communities.

While the syntax defined by this specification diverges somewhat from that defined by JSON Activity Streams 1.0, the verbs, objectTypes, extensions and fundamental model defined by that original specification remain intact.

Refer to Section 6 for more detail about the differences between the 1.0 and 2.0 syntax and for a listing of specific backwards compatibility requirements.

This specification incorporates several existing extensions to the 1.0 syntax directly into the 2.0 model. These include portions of the Activity Streams 1.0 Base Schema [base-schema], Audience Targeting [audience], Responses [responses], and Priority [priority] extensions.

1.2. Relationship to JSON-LD 1.0

The JSON-based Serialization for Linked Data (JSON-LD) [W3C.WD-json-ld-20130411] describes a rich syntax for the serialization of semantically-rich metadata using the JSON format. While the updated Activity Streams representation provided by this document is not defined as a "JSON-LD Vocabulary", the syntax is designed to be closely compatible with JSON-LD.

There are a few differences between JSON-LD and the serialization syntax described here, specifically:

- JSON-LD uses certain field names with a leading "@" character, such as "@id" and "@language". In this specification, the leading "@" is omitted.

- While JSON-LD allows using relative IRI references in the values of "id" properties, this specification limits identifiers to absolute IRIs.

- While it is possible to derive a JSON-LD "@context" description for the Activity Streams 2.0 JSON syntax one is not normatively provided by this specification.

When processing an Activity Streams document as JSON-LD, the following rules apply:

- The "objectType" property MUST be treated as an alias of JSON-LD "@type".

- The "id" property MUST be treated as an alias of JSON-LD "@id".

- The "language" property MUST be treated as an alias of JSON-LD "@language".

- A JSON array used to convey Link (Section 3.4) values MUST be treated as an unordered JSON-LD @set (@container = @set).

- The JSON array value for the "items" property defined in Section 3.7 MUST be treated as an ordered JSON-LD @list (@container = @list).

- The "displayName", "title", "content" and "summary" properties defined in Section 3.1 and Section 3.6 MUST be treated as JSON-LD Language Maps (@container = @language).

1.3. Syntax Conventions

This specification defines a JSON-based [RFC4627] serialization syntax.

When serialized, absent properties are represented by either (a) setting the property value to null, or (b) by omitting the property declaration altogether at the option of the publisher; these representations are semantically equivalent. If a property has an array value, the absence of any items in that array MUST be represented by omitting the property entirely or by setting the value to null.

This specification uses IRIs [RFC3987]. Every URI [RFC3986] is also an IRI, so a URI may be used wherever an IRI is named. There are two special considerations: (1) when an IRI that is not also a URI is given for dereferencing, it MUST be mapped to a URI using the steps in Section 3.1 of [RFC3987] and (2) when an IRI is serving as an "id" value, it MUST NOT be so mapped.

Unless otherwise specified, all properties with date and time values MUST conform to the "date-time" production in [RFC3339], with an uppercase "T" character used to separate date and time, and an uppercase "Z" character in the absence of a numeric time zone offset. All such timestamps SHOULD be represented relative to Coordinated Universal Time (UTC).

2. Example Activities

Following are three examples of activities with varying degrees of detail.

2.1. Example 1: Minimal Activity

Expresses the statement "'urn:example:person:martin' posted 'http://example.org/foo.jpg'". No additional detail is given.

```
{
  "verb": "post",
  "actor": "urn:example:person:martin",
```

```
  "object": "http://example.org/foo.jpg"
}
```

2.2. Example 2: Basic activity with some additional detail

Expresses the statement "Martin Smith posted an article to the blog 'Martin's Blog' at 3:04 PM GMT on February 2, 2011." Some additional details about the article, actor and target blog are given.

```
{
  "verb": "post",
  "published": "2011-02-10T15:04:55Z",
  "language": "en",
  "actor": {
    "objectType": "person",
    "id": "urn:example:person:martin",
    "displayName": "Martin Smith",
    "url": "http://example.org/martin",
    "image": {
      "url": "http://example.org/martin/image.jpg",
      "mediaType": "image/jpeg",
      "width": 250,
      "height": 250
    }
  },
  "object" : {
    "objectType": "article",
    "id": "urn:example:blog:abc123/xyz"
    "url": "http://example.org/blog/2011/02/entry",
    "displayName": "Why I love Activity Streams"
  },
  "target" : {
    "objectType": "blog",
    "id": "urn:example:blog:abc123",
    "displayName": "Martin's Blog",
    "url": "http://example.org/blog/"
  }
}
```

2.3. Example 3: An extended activity

A more extensive, single-entry "Activity Stream" follows. In addition to containing a number of required and optional core properties, the example contains the additional, undefined extension properties "foo" and "foo2" for illustrative purposes only.

```
{
  "totalItems": 1,
  "items" : [
    {
      "verb": "post",
      "language": "en",
      "published": "2011-02-10T15:04:55Z",
      "foo": "some extension property",
      "generator": "http://example.org/activities-app",
      "provider": "http://example.org/activity-stream",
      "displayName": {
        "en": "Martin posted a new video to his album.",
        "ga": "Martin phost le fisean nua a albam."
      },
      "actor": {
        "objectType": "person",
        "id": "urn:example:person:martin",
        "displayName": "Martin Smith",
        "url": "http://example.org/martin",
        "foo2": "some other extension property",
        "image": {
          "url": "http://example.org/martin/image",
          "mediaType": "image/jpeg",
          "width": 250,
          "height": 250
        }
      },
      "object" : {
        "objectType": {
          "id": "http://example.org/Photo",
          "displayName": "Photo"
        },
        "id": "urn:example:album:abc123/my_fluffy_cat",
        "url": "http://example.org/album/my_fluffy_cat.jpg",
        "image": {
          "url": "http://example.org/album/my_fluffy_cat_thumb.jpg",
          "mediaType": "image/jpeg",
          "width": 250,
          "height": 250
        }
      },
      "target": {
        "objectType": {
          "id": "http://example.org/PhotoAlbum",
          "displayName": "Photo-Album"
        },
        "id": "urn:example.org:album:abc123",
        "url": "http://example.org/album/",
```

```
          "displayName": {
            "en": "Martin's Photo Album",
            "ga": "Grianghraif Mairtin"
          },
          "image": {
            "url": "http://example.org/album/thumbnail.jpg",
            "mediaType": "image/jpeg",
            "width": 250,
            "height": 250
          }
        }
      }
    ]
  }
```

3. Object Model

3.1. Object

The following "core properties" apply to all JSON objects serialized within an Activity Stream document.

Property	Value	Description
id	IRI	Provides a permanent, universally unique identifier for the object in the form of an absolute IRI [RFC3987]. Objects SHOULD contain a single "id" property. If an object does not contain an "id" property, consumers MAY use the value of the "url" property as a less-reliable, non-unique identifier.
objectType	Type value (Section 3.3)	Identifies the type of object. An object MAY contain a "objectType" property whose value is a Type value (Section 3.3). If no "objectType" property is specified, the object has no specific type.
language	[RFC5646] Language Tag	Establishes the default language assumed for human-readable, natural-language metadata values included in the object. An object MAY contain a "language" property whose value MUST be a [RFC5646] Language-Tag.
displayName	Natural	A simple human-readable, plain-text

	Language value (Section 3.2)	name for the object. HTML markup MUST NOT be included. An object MAY contain a "displayName" property. If the object does not specify a "objectType" property, the object SHOULD specify a "displayName".
url	Link (Section 3.4) value	A Link (Section 3.4) value describing a resource that provides a representation of the object. An object MAY contain a "url" property.

3.2. Natural Language Values

Natural Language values represent human-readable character sequences in one or more languages. They are expressed as either (1) a single JSON string or (2) a JSON dictionary mapping [RFC5646] Language-Tags to localized, equivalent translations of the same string value.

For instance, the "displayName" property in all objects is a Natural Language value.

A single String value using the default language:

```
{
  "language": "en",
  "displayName": "This is the title"
}
```

Multiple, language-specific values:

```
{
  "displayName": {
    "en": "This is the title",
    "fr": "C'est le titre",
    "sp": "Este es el titulo"
  }
}
```

Each key in the JSON dictionary MUST be an [RFC5646] Language Tag. The associated values MUST be Strings.

3.3. Type Values

Type values represent references to or descriptions of an abstract type. They are expressed as either: (1) a String conforming to either the "isegment-nz-nc" or "IRI" productions in [RFC3987] or (2) an Object (Section 3.1). When represented as a String, the use of relative references other than a simple name is not allowed. When represented as an Object, the "id" property MUST be specified.

Within the Activity Streams 2.0, Type values are used only by the "objectType" and "verb" properties.

Object type as a simple name (isegment-nz-nc):

```
{
  "objectType": "person",
  "displayName": "John"
}
```

Object type as an absolute IRI:

```
{
  "objectType": "http://example.org/Person",
  "displayName": "John"
}
```

Object type as an object:

```
{
  "objectType": {
    "id": "http://example.org/Person",
    "displayName": "Person"
  },
  "displayName": "John"
}
```

Because the second and third examples above each specify "http://example.org/Person", the two examples are considered to specify the same type.

Verb as a simple name (isegment-nz-nc):

```
{
  "verb": "post",
  "actor": "acct:john.doe@example.org",
  "object": "http://example.org/123"
}
```

Verb as an absolute IRI:

```
{
  "verb": "http://example.com/Upload",
  "actor": "acct:john.doe@example.org",
  "object": "http://example.org/123"
}
```

Verb as an object:

```
{
  "verb": {
    "id": "http://example.com/Upload",
    "displayName": "Upload"
  },
  "actor": "acct:john.doe@example.org",
  "object": "http://example.org/123"
}
```

Allowing verbs and object types to be represented as objects rather than simple names or IRIs is intended to simplify the use of extensions that an implementation might not have encountered previously. The object properties provide additional information and metadata about the new verb or object type.

It is important to note that because the "id" property is strictly limited to absolute IRI values, the object representation cannot be used to describe types with simple names.

3.4. Link Values

Link values represent references to other objects and resources. They are expressed as either: (1) a String containing an absolute or relative IRI, (2) an Object (Section 3.1), or (3) a JSON Array containing a mixture of IRIs or Objects (Section 3.1). Link values are closely related to the conceptual model of Links as established in [RFC5988].

For example, as defined previously, all objects (Section 3.1) can contain an "image" property whose value describes a graphical representation of the containing object. This property will typically be used to provide the URL to a JPEG, GIF or PNG type resource that can be displayed to the user. Any given object might have multiple such visual representations -- multiple screenshots, for instance, or the same image at different resolutions. Using Link values, there are essentially three ways of describing such references.

To reference a single image without any additional metadata, the link value can be expressed as a simple JSON string containing an absolute or relative IRI:

```
{
  "objectType": "application",
  "id": "http://example.org/application/123",
  "displayName": "My Application",
  "image": "http://example.org/application/123.png"
}
```

Alternatively, if additional metadata is required, the link can be expressed as an object containing the url property.

```
{
  "objectType": "application",
  "id": "http://example.org/application/123",
  "displayName": "My Application",
  "image": {
    "url": "http://example.org/application/123.png",
    "mediaType": "image/png",
    "height": 320,
    "width": 320
  }
}
```

If more than one link value is to be expressed, A JSON Array with a mix of string and object elements can be used:

```
{
  "objectType": "application",
  "id": "http://example.org/application/123",
  "displayName": "My Application",
  "image": [
    "http://example.org/application/abc.gif",
    {
      "url": "http://example.org/application/123.png",
      "mediaType": "image/png",
      "height": 320,
      "width": 320
    }
  ]
}
```

Individual items contained in such an array are independent of the others and no significance is given to the ordering of those items.

RFC 5988 defines that all Links have a "link relation" that describes the contextual purpose of the link. Within an object (Section 3.1), in the absence of a specific "rel" property within the link object itself, the name of the property whose value is a link serves as the "link relation". Any valid link relation value, as defined by RFC 5988, can be used as a property with a link value in any Activity Streams object, except where the link relation might conflict with any other property defined by this specification.

In the following example, two separate links are provided. The link relation of the first is "image", while the link relation of the second is "preview". Both links, however, can be used as alternative visual representations of the "application" object.

```
{
  "objectType": "application",
  "image": [
    "http://example.org/foo.jpg",
    {
      "url": "http://example.org/screens/1.jpg",
      "rel": "preview",
      "mediaType": "image/jpeg"
    }
  ]
}
```

When an object (Section 3.1) is used to represent a Link value, the following additional properties MAY be used:

Property	Value	Description
rel	RFC 5988 Link Relation	The RFC 5988 Link Relation associated with this link value. If absent, the name of the property is assumed to specify the link relation.
mediaType	MIME Media Type	The MIME media type of the resource being referenced.

3.5. Activity

Activity objects are specializations of the base Object (Section 3.1) type that provide metadata about potential or completed actions.

Within an Activity object, the "verb" property is used to identify the type of activity. All existing verb definitions used in JSON Activity Streams 1.0 implementations can continue to be used and retain their existing semantics. If the "verb" is not specified, the "objectType" property MAY be used as an alternative means of determining the activity type.

Activity objects extend the core object (Section 3.1) definition with the following additional, optional properties:

Property	Value	Description
verb	Type value (Section 3.3)	Identifies the type of activity. An activity SHOULD contain a "verb" property whose value is a Type value (Section 3.3). If the "verb" property is not specified, the activity MUST contain a "objectType" property.
actor	Link (Section 3.4) value	Describes one or more entities that either peformed or are expected to perform the activity.
object	Link (Section 3.4) value	Describes the primary object of the activity. For instance, in the activity, "John saved a movie to his wishlist", the object of the activity is "movie". An activity SHOULD contain an "object" property. If the "object" property is not contained, the primary object of the activity MAY be implied by context.

target	Link (Section 3.4) value	Describes the target of the activity. The precise meaning of the activity's target is dependent on the activities "verb", but will often be the object the English preposition "to". For instance, in the activity, "John saved a movie to his wishlist", the target of the activity is "wishlist". The activity target MUST NOT be used to identity an indirect object that is not a target of the activity.
result	Link (Section 3.4) value	Describes the result of the activity. For instance, if a particular action results in the creation of a new resource, the "result" property can be used to describe that new resource.
priority	Decimal Number between 0.00 and 1.00	An indicator of the relative priority or importance that the creator of an activity considers the it to have. Represented as a numeric decimal between 0.00 and 1.00, with two decimal places of precision. If the property is omitted or set to null, the assumption is that a default priority can be assumed. The value 0.00 represents the lowest possible priority while 1.00 represents the highest.

3.5.1. Considerations on the use of "priority"

The presence of the "priority" property does not impose any specific processing or display requirements on the part of any entity consuming the activity.

Expressing the value as a range of numeric decimal values is intended to provide the greatest level of flexibility in the expression and consumption of prioritization detail. It is expected that implementors consuming activity objects containing "priority" will utilize and expose the additional information in a number of different ways depending on the unique requirements of each application use case.

Many existing systems do not represent priority values as numeric ranges. Such systems might use fixed, labeled brackets such as "low", "normal" and "high" or "urgent". Similar mechanisms can be established, by convention, when using the "priority" property. In

typical use, it is RECOMMENDED that implementations wishing to work with such defined categories treat "priority" property values in the range 0.00 to 0.25 as "low" priority; values greater than 0.25 to 0.75 as "normal" priority; and values greater than 0.75 to 1.00 as "high" priority. Specific implementations are free to establish alternative conventions for the grouping of priority values with the caveat that such conventions likely will not be understood by all implementations.

3.5.2. Audience Targeting Properties

Every Activity has both a Primary and Secondary audience. The Primary audience consists of those entities either directly involved in the performance of the activity or who "own" the objects involved. The Secondary audience consists of the collection of entities sharing an interest in the activity but who are not directly involved (e.g. "followers").

For instance, suppose a social network of three individuals: Bob, Joe and Jane. Bob and Joe are each friends with Jane but not friends with one another. Bob has chosen to "follow" activities for which Jane is directly involved. Jane shares a file with Joe.

In this example, Jane and Joe are each directly involved in the file sharing activity and together make up the Primary Audience for that event. Bob, having an interest in activities involving Jane, is the Secondary Audience. Knowing this, a system that produces or consumes the activity can intelligently notify each person of the event.

While there are means, based on the verb, actor, object and target of the activity, to infer the primary audience for many types of activities, those do not work in every case and do not provide a means of identifying the secondary audience. The "to", "cc", "bto" and "bcc" properties MAY be used within an Activity to explicitly identify the Primary and Secondary audiences.

Property	Value	Description
to	Link (Section 3.4) value	Specifies the public primary audience.
cc	Link (Section 3.4) value	Specifies the public secondary audience.
bto	Link (Section 3.4) value	Specifies the private primary audience.
bcc	Link (Section 3.4) value	Specifies the private secondary audience.

The prototypical use case for an Activity containing these properties is the publication and redistribution of Activities through an intermediary. That is, an event source generates the activity and publishes it to the intermediary which determines a subset of events to display to specific individual users or groups. Such a determination can be made, in part, by identifying the Primary and Secondary Audiences for each activity.

When the event source generates the activity and specifies values for the to and cc fields, the intermediary SHOULD redistribute that event with the values of those fields intact, allowing any processor to see who the activity has been targeted to. This is precisely the same model used by the to and cc fields in email systems.

There are situations, however, in which disclosing the identity of specific members of the audience may be inappropriate. For instance, a user may not wish to let other users know that they are interested in various topics, individuals or types of events. To support this option, an event source generating an activity MAY use the "bto" and "bcc" properties to list entities to whom the activity should be privately targeted. When an intermediary receives an activity containing these properties, it MUST remove those values prior to redistributing the activity. The intent is that systems MUST consider entities listed within the "bto" and "bcc" properties as part of the Primary and Second audience but MUST NOT disclose that fact to any other party.

Audience targeting information included within an Activity only describes the intent of the activity creator. With clear exception given to the appropriate handling of "bto" and "bcc", this specification leaves it up to implementations to determine how the audience targeting information is used.

3.6. Additional Object Properties

The following "additional properties" MAY be used with any JSON Object serialized within an Activity Stream document.

Property	Value	Description
alias	IRI	Provides a contextually meaningful alternative label for the object in addition to the "id". For instance, within some systems, groups can be identified both by a unique global ID and a more "human-friendly" label such as "@friends" or "@network". The value of the "alias" property MUST match either the "isegment-nz-nc" or the "IRI" production in [RFC3987]. The use of a relative reference other than a simple name is not allowed.
attachments	Link (Section 3.4) value	A Link (Section 3.4) value referencing one or more objects associated with the containing object. These are similiar in concept to files attached to an email message.
author	Link (Section 3.4) value	A Link (Section 3.4) value referencing one or more entity that created or authored the object.
content	Natural Language value (Section 3.2)	A Natural-language description of the object encoded as a single JSON String containing HTML markup. Visual elements such as thumbnail images MAY be included.
duplicates	Link (Section 3.4) value	A Link (Section 3.4)value referencing one or more objects that are semantically equivalent to this object or duplicate this objects content. An object SHOULD contain a "duplicates" property when there are known objects, possibly in a different system, that are semantically equivalent or duplicate the content.
icon	Link (Section 3.4) value	A Link (Section 3.4) value referencing one or more visual, graphic representations of the

		object, intended for human consumption. The visual element SHOULD have an aspect ratio of one (horizontal) to one (vertical) and SHOULD be suitable for presentation at a small size.
image	Link (Section 3.4) value	A Link (Section 3.4) value referencing one or more visual, graphic represenations of the object. Unlike the "icon" property, there are no aspect ratio or display restrictions.
location	Link (Section 3.4) value	A Link (Section 3.4) value describing one or more physical or virtual locations associated with which the object.
published	[RFC3339] date-time	The date and time at which the object was published.
generator	Link (Section 3.4) value	A Link (Section 3.4) value referencing the application that generated the object.
provider	Link (Section 3.4) value	A Link (Section 3.4) value referencing the application that published the object. Note that this is not necessarily the same entity that generated the object.
summary	Natural Language value (Section 3.2)	A Natural-language summarization of the object encoded as a single JSON String containing a fragment of HTML markup. Visual elements such as thumbnail images can be included.
updated	[RFC3339] date-time	The date and time at which a previously published object has been modified.
startTime	[RFC3339] date-time	A date-time describing the actual or expected starting time of the object. When used within an Activity object, for instance, the "startTime" specifies the moment the activity began or is scheduled to begin.
endTime	[RFC3339] date-time	A date-time describing the actual or expected ending time of the object. When used within an Activity object, for instance, the "endTime" specifies the moment the activity concluded or is scheduled to conclude.

rating	Decimal Number between 1.0 and 5.0	A quality rating expressed as a number between 1.0 and 5.0 (inclusive) with one decimal place of precision.
tags	Link (Section 3.4) value	A Link (Section 3.4) value referencing one or more resources that are loosely associated with the containing object. The "tags" and "attachments" properties differ from one another in that the "tags" property asserts "association by reference" while "attachments" asserts "association by enclosure".
title	Natural Language (Section 3.2) value	A Natural-language title for the object expressed as a fragment of HTML markup. The "title" and "displayName" properties are closely related and overlap in function with the key difference being that "title" is permitted to contain HTML markup, while "displayName" is not.
duration	Integer or [RFC3339] duration	When the object describes a time-based resource, such as audio or video, the "duration" property indicates the approximate duration of time expressed as an either an RFC 3339 "duration" (e.g. a duration of 5 seconds is represented as "PT5S") or as a non-negative integer specifying the duration in seconds.
height	Integer	When the object describes a visual resource, such as an image, video or embeddable HTML page, the "height" property indicates the recommended display height in pixels.
width	Integer	When the object describes a visual resource, such as an image, video or embeddable HTML page, the "width" property indicates the recommended display width in pixels.
inReplyTo	Link (Section 3.4) value	A Link (Section 3.4) value identifying one or more other objects to which the containing object can be considered a response.
actions	Action (Section 3.6.1)	An optional Action (Section 3.6.1) value that describes potential activities that can be performed

	value	with the object.
scope	Link (Section 3.4) value	A Link (Section 3.4) value identifying one or more resources that define the total population of entities for which the object is considered to be relevant.

3.6.1. Action Values

The "actions" property on an Activity Streams object is used to describe the kinds of activities that can be taken with regards to the object. The value is expressed as a JSON dictionary mapping verbs to Link (Section 3.4) values referencing resources or objects that can be used to carry out those verbs.

For instance, a hypothetical object with "video" as the objectType might have "watch", "share" and "embed" as potential actions:

```
 {
   "objectType": "video",
   "id": "http://example.org/cats.mpg",
   "actions": {
     "watch": "movie://example.org/cats.mpg",
     "share": {
       "objectType": "service",
       "displayName": "My Sharing Service",
       "url": "http://example.net/share"
     },
     "embed": [
       "http://example.org/gadgets/video.xml?v=cats.mpg",
       {
         "objectType": "inline-html",
         "content": "<video ... />"
       }
     ]
   }
 }
```

Each key in the Action value MUST be a valid verb identifier conforming to either the "isegment-nz-nc" or "IRI" productions in [RFC3987] and be suitable for use as a value for Activity (Section 3.5) object's "verb" property. The value of each key MUST be a valid Link (Section 3.4) value.

3.7. Collection

Collection objects are a specialization of the base Object (Section 3.1) that contain a listing of other objects (Section 3.1) The Collection object is used primarily as the root of an Activity Streams document as described in Section 4, but can be used as the value of object properties.

Collections have both a logical model and a physical serialization. While the logical view of a collection might contain a large number of objects, any single serialized representation might include only a subset of those objects, with specific Link (Section 3.4) values used to reference additional serialized representations that include additional subsets. Such representations are known as "multi-page collections", with each serialized subset representing a single "page".

The value of the Collection object's "objectType" property MUST be "collection" unless the fact that the object is a collection can be determined by context.

Collection objects extend the core object (Section 3.1) definition with the following additional properties:

Property	Value	Description
totalItems	Integer	Non-negative integer specifying the total number of objects contained by the logical view of the collection. This number might not reflect the actual number of items serialized within the Collection object instance.
items	Array of Objects (Section 3.1)	An array containing a listing of Objects (Section 3.1) of any type.
itemsAfter	[RFC3339] date-time	A RFC 3339 date-time that indicates that the collection contains only items published or updated strictly after the date and time specified.
itemsBefore	[RFC3339] date-time	A RFC 3339 date-time that indicates that the collection contains only items published or updated strictly before the date and time specified.
itemsPerPage	Integer	A non-negative integer specifying the maximum number of items that

		will be included in the value of the items array.
startIndex	Integer	A non-negative integer value identifying the relative position within the logical view of collection of the first object contained in the items property. For instance, if there are 20 items that are considered to be members of a collection, but only the last 10 of those items are included in the items property, the value of startIndex would be 10.
first	Link (Section 3.4) value	A Link (Section 3.4) value referencing the furthest preceeding page of a multi-page collection.
last	Link (Section 3.4) value	A Link (Section 3.4) value referencing the furthest following page of a multi-page collection.
prev	Link (Section 3.4) value	A Link (Section 3.4) value referencing the immediately preceding page of the multi-page collection. Note that the property name previous can be used as an equivalent alternative; however implementations SHOULD use prev and MUST NOT use both prev AND previous within the same collection.
next	Link (Section 3.4) value	A Link (Section 3.4) value referencing the immediately following page of the multi-page collection.
current	Link (Section 3.4) value	A Link (Section 3.4) value referencing the page containing the items that have been updated or published most recently.
self	Link (Section 3.4) value	A Link (Section 3.4) value referencing this page.

3.7.1. Using Collections as Summary Values

It is a common practice to use Collection objects to provide summary information on the number of specific types of events that have occurred with respect to any given object. For instance, a "note" object may have been "shared" or "liked" a number of times by

different individuals. In such cases, the Collection object is used as a property value with the "totalItems" field used to indicate the total number of occurrences, the "items" property used to provide details for a subset of the most recent occurrences, and the "id" property used to reference a separate Activity Streams document providing additional information.

This specification defines the following properties that MAY be used within any object (Section 3.1) as "summary values":

Property	Value	Description
replies	Collection (Section 3.7)	Provides information about the set of objects that can be considered to be replies to the containing object.

In the following example, the "replies" property is used to indicate that a note has 10 responses, and provides information on the most recently received response:

```
{
  "objectType": "note",
  "id": "urn:example:note:1",
  "displayName": "A note about things",
  "content": "blah blah blah",
  "replies": {
    "url": "http://example.org/note/1/comments.json",
    "mediaType": "application/activity+json",
    "totalItems": 10,
    "items": [
      {
        "objectType": "note",
        "id": "urn:example:note:1:A",
        "content": "That's profound, man."
      }
    ]
  }
}
```

4. The Activity Stream JSON Document

The above defined JSON serialization can be used to represent activities, objects and media links in any context. This section

defines one particular use of the above formats to publish a JSON document representing an ordered listing of Activity objects.

Publishers using this format MUST produce a valid JSON document whose root value is a Collection (Section 3.7).

The MIME media type of this document MUST be "application/activity+json".

5. Reserved Object Types and Verbs

The following objectType and verb values are reserved for specific uses by this specification:

5.1. Object Types

Type	Description
activity	Objects specifying "objectType":"activity" conform to the Activity construct defined in Section 3.5.
verb	Objects specifying "objectType":"verb" provide metadata about an extension verb as defined in Section 3.3. The "id" attribute of such objects MUST be provided.
objectType	Objects specifying "objectType":"objectType" provide metadata about an extension objectType as defined in Section 3.3. The "id" attribute of such objects MUST be provided.
collection	Objects specifying "objectType":"collection" conform to the Collection construct defined in Section 3.7.

5.2. Verbs

Type	Description
post	The "post" verb describes the act of authoring an object and then publishing it online. The actor can be any entity; the object can be of any object type; and the target, if specified, can be of any object type. A target, however, is not required.

6. Deprecated Activity Streams 1.0 Syntax

The JSON syntax defined by this specification differs somewhat from that defined in the original JSON Activity Streams 1.0 [activitystreams-1.0] specification in ways that are not backwards compatible. Implementations can choose to continue supporting the JSON Activity Streams 1.0 syntax but SHOULD consider it to be deprecated. This means that while implementations MAY continue to consume the 1.0 syntax, they SHOULD NOT output the 1.0 syntax unless specifically interacting with older non-2.0 compliant implementations.

Specifically:

1. Implementations MUST use the "application/stream+json" MIME media type when producing a JSON serialization of an Activity Object conforming to the 1.0 syntax, and "application/activity+json" when producing a serialization conforming to the 2.0 syntax.

2. Implementations that process serializations of an Activity Object identified using either the "application/stream+json" or the more generic "application/json" MIME media type MUST follow the syntax and processing rules set by [activitystreams-1.0]. The 2.0 syntax and processing rules apply only when handling serializations using the "application/activity+json" media type.

3. This document redefines the "displayName", "title", "content" and "summary" properties as Natural Language values (Section 3.2), which means their values can be expressed as either a String or a JSON-LD Language Map. In the 1.0 syntax, these are expressed solely as String values. Because the 1.0 values are a valid subset allowed by this specification, implementations are not required to take any specific action to continue supporting those values.

4. This document redefines a large number of common properties defined originally as Objects in 1.0 as Link values (Section 3.4). This means the property values can be expressed as either an IRI String, an Object, or an Array of IRI Strings and Objects. Because the 1.0 values are a valid subset allowed by this specification, implementations are not required to take any specific action to continue supporting those values.

5. This specification replaces the "upstreamDuplicates" and "downstreamDuplicates" properties defined in the 1.0 syntax with a singular "duplicates" property with a Link value (Section 3.4). The "upstreamDuplicates" and "downstreamDuplicates" property values in 1.0 are defined as Arrays of strings. Implementations

MUST consider the union of these two values as an alias for the "duplicates" property.

By following these requirements, all JSON Activity Streams 1.0 serializations can be processed successfully by 2.0 implementations.

7. Comparison of Identifier Values

The values of "id" properties can be compared to determine if the identifiers represent duplicate content. The values MUST be compared on a character-by-character, case-sensitive basis. Comparisons MUST be based solely on the character strings themselves and MUST NOT rely on dereferencing the IRIs or URIs mapped from them.

As a consequence, two IRIs that resolve to the same resource but are not character-for-character identical will be considered different for the purposes of identifier comparison. In such cases, the "duplicates" property can be used to expressly relate such objects to one another.

8. Extensibility

Processors that encounter unfamiliar properties within any Activity Streams object MUST NOT stop processing or signal an error and MUST continue processing the items as if those properties were not present.

9. Security Considerations

Publishers or Consumers implementing Activity Streams as a stream of public data may also want to consider the potential for unsolicited commercial or malicious content and should take preventative measures to recognize such content and either identify it or not include it in their implementations.

Publishers should take reasonable measures to ensure potentially malicious user input such as cross-site scripting attacks are not included in the Activity Streams data they publish.

Consumers that re-emit ingested content to end-users MUST take reasonable measures if emitting ingested content to make sure potentially malicious ingested input is not re-emitted.

Consumers that re-emit ingested content for crawling by search engines should take reasonable measures to limit any use of their site as a Search Engine Optimization loophole. This may include converting un-trusted hyperlinks to text or including a rel="nofollow" attribute.

Consumers should be aware of the potential for spoofing attacks where the attacker publishes activities or objects with falsified property values with the intent of injecting malicious content, hiding or corrupting legitimate content, or misleading users.

Activity Streams are JSON Documents and are subject to the same security considerations described in [RFC4627].

Activity Streams implementations handle URIs. See Section 7 of [RFC3986].

Activity Streams implementations handle IRIs. See Section 8 of [RFC3987].

10. IANA Considerations

10.1. application/activity+xml Media Type

This specification registers the application/activity+json MIME Media Type:

Type name: application

Subtype name: activity+json

Required parameters: None

Optional parameters: "charset" : Specifies the character set encoding. If not specified, a default of "UTF-8" is assumed.

Encoding considerations: Resources that use the "application/activity+json" media type are required to conform to the "application/json" Media Type and are therefore subject to the same encoding considerations specified in Section 6 [RFC4627].

Security considerations: As defined in this specification

Published specification: This specification.

Applications that use this media type: JSON Activity Streams are implemented by a wide range of existing applications.

Additional information:

Magic number(s): N/A

File extension(s): N/A

Macintosh file type code(s): TEXT

Person & email address to contact for further information: James M Snell <jasnell@gmail.com>

Intended usage: COMMON

Restrictions on usage: None.

Author: James M Snell <jasnell@gmail.com>

Change controller: IESG

11. References

11.1. Normative References

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

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

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

[RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.

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

[W3C.WD-json-ld-20130411] Sporny, M., Kellogg, G., and M. Lanthaler, "JSON-LD 1.0", World Wide Web Consortium LastCall WD-json-ld-20130411, April 2013, <http://www.w3.org/TR/2013/WD-json-ld-20130411>.

[activitystreams-1.0] Snell, J., Atkins, M., Norris, W., Messina, C., Wilkinson, M., and R. Dolin, "JSON Activity Streams 1.0", May 2011, <http://activitystrea.ms/specs/json/1.0/>.

11.2. Informational References

[RFC6963] Saint-Andre, P., "A Uniform Resource Name (URN) Namespace for Examples", BCP 183, RFC 6963, May 2013.

[audience] Snell, J., "Audience Targeting for JSON Activity Streams", March 2012, <http://activitystrea.ms/specs/json/targeting/1.0/>.

[base-schema] Activity Streams Workgroup, "Activity Streams - Base Schema", September 2012, <https://github.com/activitystreams/activity-schema/blob/master/activity-schema.md>.

[priority] Snell, J., "Priority Extension for JSON Activity Streams", June 2012, <https://raw.github.com/jasnell/specs/master/activitystrea.ms/priority_extension.txt>.

[responses] Snell, J., "Responses for Activity Streams", March 2012, <http://activitystrea.ms/specs/json/replies/1.0/>.

Appendix A. Acknowledgements

The author wishes to thank the Activity Streams community and implementers for their support, encouragement, and enthusiasm including but not limited to: Abdul Qabiz, Adina Levin, Adrian Chan, Adriana Javier, Alan Hoffman, Alex Kessinger, Alexander Ovchinnikov, Alexander Zhuravlev, Alexandre Loureiro Solleiro, Amy Walgenbach, Andres Vidal, Angel Robert Marquez, Ari Steinberg, Arjan Scherpenisse, Arne Roomann-Kurrik, Beau Lebens, Ben Hedrington, Ben Metcalfe, Ben Werdmuller, Benjamin Goering, Bill de hOra, Bo Xing, Bob Aman, Bob Wyman, Brett Slatkin, Brian Walsh, Brynn Evans, Charlie Cauthen, Chris Chabot, Chris Messina, Chris Toomey, Christian Crumlish, Dan Brickley, Dan Scott, Daniel Chapman, Danny Ayers, Dare Obasanjo, Darren Bounds, David Cramer, David Nelson, David Recordon, DeWitt Clinton, Douglas Pearce, Ed Summers, Elias Bizannes, Elisabeth Norris, Eric Marcoullier, Eric Woods, Evan Prodromou, Gee-Hsien Chuang, Greg Biggers, Gregory Foster, Henry Saputra, Hillary Madsen, Howard Liptzin, Hung Tran, Ian Kennedy, Ian Mulvany, Ivan Pulleyn, Jacob Kim, James Falkner, James Pike, James Walker, Jason Kahn, Jason Kantz, Jeff Kunins, Jeff Martin, Jian Lin, Johannes Ernst, John Panzer, Jon Lebkowsky, Jon Paul Davies, Jonathan Coffman, Jonathan Dugan, Joseph Boyle, Joseph Holsten, Joseph Smarr, Josh Brewer, Jud Valeski, Julien Chaumond, Julien Genestoux, Jyri Engestroem, Kaliya

Hamlin, Kevin Marks, Laurent Eschenauer, Laurie Voss, Leah Culver, Libby Miller, Manu Mukerji, Mark Weitzel, Marko Degenkolb, Marshall Kirkpatrick, Martin Atkins, Martin Svensson, Marty Alchin, Mary Hoder, Matt Leventi, Matt Wilkinson, Matthias Mueller-Prove, Max Engel, Max Wegmueller, Melvin Carvalho, Michael Buckbee, Michael Chan, Michael Richardson, Michael Sullivan, Mike Macgirvin, Mislav Marohnić, Mo Jangda, Monica Wilkinson, Nate Benes, NeilFred Picciotto, Nick Howard, Nick Lothian, Nissan Dookeran, Nitya Narasimhan, Pablo Martin, Padraic Brady, Pat G. Cappalaere, Patrick Aljord, Peter Ferne, Peter Reiser, Peter Saint-Andre, Phil Wolff, Philip (flip) Kromer, Richard Cunningham, Richard Zhao, Rick Severson, Robert Hall, Robert Langbert, Robert Dolin, Robin Cover, Ryan Boyd, Sam Sethi, Scott Raymond, Scott Seely, Simon Grant, Simon Wistow, Stephen Garcia, Stephen Sisk, Stephen Paul Weber, Steve Ivy, Steve Midgley, Steven Livingstone-Perez, Sylvain Carle, Sylvain Hellegouarch, Tantek Celik, Tatu Saloranta, Tim Moore, Timothy Young, Todd Barnard, Tosh Meston, Tyler Gillies, Will Norris, Zach Copley, and Zach Shepherd.

Appendix B. Processing as JSON-LD

While the Activity Streams 2.0 syntax is designed to be compatible with JSON-LD, in order to successfully process an Activity Streams document as JSON-LD, a "@context" description needs to be provided. The following example illustrates an Activity Streams document that can be processed as JSON-LD containing Schema.org defined metadata elements.

```
{
  "@context": {
    "@vocab": "http://activitystrea.ms/spec/2.0/",
    "verb": "@type",
    "objectType": "@type",
    "id": "@id",
    "actor": "http://schema.org/Action/performedBy",
    "object": "http://schema.org/BuyAction/bought",
    "purchase": "http://schema.org/BuyAction",
    "person": "http://schema.org/Person",
    "book": "http://schema.org/Book"
  },
  "verb" : "purchase",
  "id" : "urn:example:purchase:123/abc",
  "displayName": "John purchased 'A Tale of Two Cities'",
  "startTime" : "2013-04-02T12:31Z",
  "endTime" : "2013-04-02T12:31Z",
  "actor": {
    "objectType": "person",
    "displayName": "John Doe"
```

```
  },
  "object": {
    "objectType": "book",
    "displayName": "A Tale of Two Cities"
  }
}
```

Appendix C. Motivational Use Cases

This specification defines a number of syntax changes relative to the JSON Activity Streams 1.0 specification. The sections that follow describe some of the general motivations for these changes with illustrative examples.

C.1. Internationalization (i18n)

The JSON Activity Streams 1.0 syntax has no inherent notion of a "language context". That is, the core syntax has no internal mechanism a publisher can use to identify the language used when constructing the Activity Streams document. Nor are there any existing mechanisms at the JSON syntax level that an Activity Streams implementation can inherit. This specification introduces the "language" property and Natural Language Value concepts to fill this gap.

Imagine a scenario with a service that receives Activity objects from users and republishes those to a distributed audience of interested parties. This service spans international boundaries and the users speak a multitude of different languages. Within this system, a native English speaker might subscribe to notifications about activities posted by a native French speaker.

For instance, let's suppose that our native French speaker posts the following activity to this system:

```
POST /activity/feed HTTP/1.1
Content-Type: application/activity+xml

{
  "verb": "post",
  "language": "fr",
  "object": {
    "objectType": "article",
    "displayName": "Un exemple basique",
    ...
  }
}
```

The system receives this activity post and prepares to notify our native English speaking user. Knowing that this user prefers English and does not speak a word of French, the system can inspect the Activity and detect automatically that a translation ought to be provided. Rather than replacing the original French text, however, the service can simply add in the English translation along side it.

```
{
  "verb": "post",
  "language": "fr",
  "actor": {
    "id": "urn:example:person:abc",
    "displayName": "Jean Valjean"
  },
  "object": {
    "type": "article",
    "displayName": {
      "fr": "Un exemple basique",
      "en": "A basic example"
    }
    ...
  }
}
```

It is also possible for a Natural Language Value to express alternative same-language representations of a string that utilize different writing systems or regions. For instance, it is common for Japanese translations to provide equivalent ideographic (kanji) and phonetic (katakana or hiragana) alternatives:

```
{
  "title": {
    "ja-Hani": "...",
```

```
      "ja-Kana": "..."
    }
  }
```

C.2. Extensibility (e11y)

Arguably, the two most important extensibility points in the Activity Streams format are the object type and verb properties. Implementations are free to come up with their own types and verbs at any point. While such extensibility is extremely powerful, it comes with a cost. Namely, implementations that encounter previously unknown verbs and object types may not have enough to knowledge about those to do anything significant with them.

For instance, the most common use case for Activity Streams today is the generation of a human-readable "activity feed" that translates Activity objects into sentences just as "John uploaded a new photo" or "Jane checked in at a hotel", etc. Given an extension verb such as "http://example.org/whatever", an implementation might not have sufficient information about that verb to generate a readable sentence describing the activity that occurred.

With Activity Streams 1.0, a number of different approaches have been tried to address this problem, but all of the solutions essentially deal with the need to provide additional metadata about extension verbs and object types so that an implementation can dynamically learn and adapt. The notion of "type values" is added by this specification to specifically deal with this issue.

For example, suppose I have an implementation that generates Activity objects that use a new extension verb "urn:example:verbs:upload". Knowing that consumers of these objects might not have encountered this verb before, I want to make it possible for those implementations to automatically discover metadata about the new verb. To do so, I can use a type value to provide some basic information.

```
{
  "verb": {
    "id": "urn:example:verbs:upload",
    "url": "http://example.org/verbs.json",
    "mediaType": "application/ld+json",
    "displayName": {
      "en": "upload",
      "fr": "televersement"
    },
    "alias": "post"
```

```
  },
  "actor": {
    "type": "person",
    "displayName": "John"
  },
  "object": {
    "type": "photo",
    "displayName": "cats.jpg"
  }
}
```

An implementation receiving this has several choices. It could choose to ignore everything other than the verb's identifier, treating it generically as one would have to today using the 1.0 syntax; or, it could inspect the metadata provided and notice that the extension verb can be treated generally as an alias of "post" or displayed in English as "upload" and in French as "televersement"; or, it can choose to attempt discovering more information about the verb by dereferencing the provided URL.

The point is, these options are built into the core syntax, making extension verbs and object types significantly more usable, particularly when combined with the new language context features.

C.2.1. Publishing Extension objectType and verb Libraries

By treating extension objectTypes and verbs as objects in their own right, it becomes trivially possible to use the Activity Streams format as a means of publishing metadata about extension verbs.

For example:

```
{
  "displayName": "My object types and verbs",
  "items": [
    {
      "objectType": "verb",
      "id": "urn:example:verbs:create",
      "alias": "post",
      "displayName": "Create"
    },
    {
      "objectType": "objectType",
      "id": "urn:example:types:article",
      "displayName": "Article"
    }
  ]
```

```
}
```

Implementations could use such documents to dynamically learn about new verbs and objectTypes.

C.3. First Class Links

Linking in the 1.0 syntax is largely undefined and inconsistent. There is a general notion of Media Link objects that are used for some things like images and videos, along with a "url" property that in some cases is used to always point to HTML represenations while in other cases might point to JSON documents or image files, and there is no reusable concept of a generic link provided for extensions to leverage which has led to inconsistent implementation. The 2.0 syntax introduced here deals with these issues by introducing a clear, consistent, reusable first class linking model.

For instance, using the 2.0 syntax, an "image" object type can be represented simply as:

```
{
  "objectType": "image",
  "url": "http://example.org/cats.jpg",
  "mediaType": "image/jpeg",
  "displayName": "A picture of my cats",
  "alternate": {
    "url": "http://example.org/gallery?i=cats.jpg",
    "mediaType": "text/html"
  }
  "preview": "http://example.org/thumbnails/cats.jpg"
}
```

Essentially, any 2.0 object that contains a "url" property can be interpreted as a link. That "url" property points to a representation of the object, while the "mediaType" property identifies the content type of that linked resource. RFC 5988 Link Relations can be used directly within the 2.0 syntax to provide additional data -- in this case, an alternative HTML representation of the image as well as a thumbnail preview.

Another case that the more flexible linking approach allows us to address is providing multiple links for a single property. For instance, it is not uncommon for there to be several alternative versions of an image resource offered at various resolutions to support multiple types of devices. With the 2.0 syntax, multiple choices for a single link can be easily provided.

```
{
  "objectType": "application",
  "displayName": "My application",
  "icon": [
    {
      "url": "http://example.org/sd/icon.png",
      "width": 57,
      "height": 57
    },
    {
      "url": "http://example.org/hd/icon.png",
      "width": 114,
      "height": 114
    }
  ],
  "preview": [
    "http://www.example.org/screenshots/1.jpg",
    "http://www.example.org/screenshots/2.jpg",
    "http://www.example.org/screenshots/3.jpg"
  ]
}
```

C.4. Use of External Vocabularies

Use of an "external vocabulary" within Activity Streams means using object types, verbs and properties that are not defined by the core Activity Streams specification. An example would be using concepts defined within a microdata vocabulary such as that defined by Schema.org (http://schema.org).

Implementations that wish to use a Activity Streams with such external vocabularies face the challenge that, often times, identical or overlapping concepts can be expressed in a multitude of ways depending on which vocabulary is selected. This can make it difficult to map abstract data models into a specific JSON serialization.

For instance, suppose an application uses the Schema.org model to represent an article, described here: http://schema.org/Article. Within this model, there are several properties defined that directly overlap properties defined by the core Activity Streams syntax. Such properties include "name", "contentLocation", and "articleBody". In order to encode the abstact model of a Schema.org/Article into the JSON Activity Streams model, the application needs to determine precisely how to map the abstract properties to the serialized format. In Activity Streams 1.0, no guidance was given on how to achieve such a mapping, within the 2.0 syntax, the JSON Serialization for Linked Data (JSON-LD) provides a foundation.

Using JSON-LD I can maintain basic Activity Streams 2.0 syntax while mapping the physical serialization to the abstract model inline.

```
{
  "objectType": "article",
  "displayName": "My article about things",
  "content": "This is my article",
  "@context": {
    "objectType": "@type",
    "article": "http://schema.org/Article",
    "displayName": "http://schema.org/name",
    "content": "http://schema.org/Article/articleBody"
  }
}
```

For any non-JSON-LD aware implementation, this can be processed just as if it were an ordinary Activity Streams object, without any additional consideration given. For a JSON-Ld aware implementation, however, the addition of the "@context" property allows the serialized JSON to be unambiguously mapped to the Schema.org concept of an "Article". The fact that we can support such a mapping allows the Activity Streams format to extend to a broader range of scenarios without requiring alternative, incompatible vocabulary specific models of "actions" or "activities" to be developed.

C.5. Embedded Actions

Every Activity Streams object represents a discreet modular component that can be distributed, shared, or acted upon in a variety of ways. The 2.0 syntax allows these components to not only express information about the content but also about the specific types of actions that can be performed with the object.

For example, an email that is automatically generated by an expense reporting system could embed a structured Activity Stream object that

contains a listing of the possible actions the recipient of the email can take. An intelligent email agent can interpret this embedded metadata and provide a tailored, in-context UI experience:

```
<html>
  <body>
    <script type="application/activity+json">
      {
        "objectType": "expense-report",
        "url": "url": "https://example.org/view?id=abc123",
        "description": "John's trip to San Francisco",
        "actions": {
          "approve": "https://example.org/approve?id=abc123",
          "reject": "https://example.org/reject?id=abc123"
        }
      }
</script>
  <p>
    Your employee, John Doe, has submitted a new expense
    report for "John's trip to San Francisco"....
  </body>
</html>
```

Author's Address

James M Snell (editor)
IBM

Email: jasnell@gmail.com