Network Working Group	E. Hammer-Lahav
Internet-Draft	Yahoo!
Intended status: Informational	January 09, 2009
Expires: July 13, 2009	

TOC

HTTP-based Resource Descriptor Discovery
draft-hammer-discovery-00

Status of this Memo

Copyright Notice

Abstract

This memo describes an HTTP-based process for obtaining information about a resource identified by a URI. The 'information about a resource' - a resource descriptor - typically provides machine-readable information that aims to assist and enhance the interaction with the resource. This memo only defines the process for locating and obtaining the descriptor, but leaves the descriptor format and its interpretation out of scope.

Table of Contents

1. Introduction

TOC

This memo aims to provide a uniform and easily implementable process for locating resource descriptors. With the development of interoperability specifications comes the need to enable compliant services and resources to declare their conformance to these

specifications. There is a growing need to describe resources in a way that does not depend on their internal structure, or even the availability of an HTTP-accessible representation of these resources. For example, while an end-user is reading a web page such as a blog article, the user-agent can discover whether the content of this page has generated from an Atom feed or Atom entry and whether that feed supports Atom authoring. It can discover whether there is an iCalendar-formatted or CalDAV calendar associated with the page, or where other content by the same page author might be found.
In an example related to the identity space, an end-user can use a URI as an identifier for signing into web services, and in turn, the web service can discover more information about the user's resources and preferences such as who did the user delegate their identity management to, where they keep their address book or list of social network friends, where their profile information is stored to reduce signup registration requirements, and what other services they use which may enhance their interaction with the web service.

---

2. Notational Conventions

TOC

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] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

---

3. Scope

TOC

The scope of this memo is intentionally restricted to locating resource descriptors, leaving out their format. Given the wide range of use cases and information that can be provided 'about a resource', no single descriptor format can adequately accommodate all needs. However, the process in which the desired descriptor is located should be consistent across use cases and formats.

---

4. Resource Discovery and Service Discovery

TOC

Resource discovery provides a process for obtaining information about a resource identified with a URI. It allows resource-providers to describe their resources in a machine-readable format, enabling automatic interoperability by user-agents and resource-consuming

applications. Discovery enables applications to utilize a wide range of web services and resources across multiple providers without the need to know about their capabilities in advance, reducing the need for manual configuration and resource-specific software.
When discussing discovery, it is important to differentiate between resource discovery and service discovery. Both types attempts to associate capabilities with resources, but they approach it from opposite ends.
Service discovery centers around identifying the location of qualified resources, typically finding an endpoint capable of certain protocols and capabilities. In contrast, resource discovery begins with a resource, trying to find which capabilities it supports.
A simple way to distinguish between the two types of discovery is to define the questions they are each trying to answer:

> **Resource-Discovery:** Given a resource, what are its attributes: capabilities, characteristics, and relationships to other resources?
>
> **Service-Discovery:** Given a set of attributes, which available resources match the desired set and what is their location?

While this memo deals exclusively with resource discovery, it is important to note that the two discovery types are closely related and are usually used in tandem. In fact, a typical use case will switch between service discovery and resource discovery multiple times in a single workflow, and can start with either one.
One reason for this dependency between the two discovery types is that resource descriptors usually contain not only a list of capabilities, but also relationships to other resources. Since those relationships are usually typed, the process in which an application chooses which links to use is in fact service discovery.
Applications use resource discovery to obtain the list of links, and service discovery to choose the relevant links. In another common example, the application uses service discovery to find a resource with a given capability, then uses resource discovery to find out what other capabilities it supports.
Unless otherwise noted, the term 'discovery' is used in this memo to mean resource discovery.

---

5. Discovery Workflow

TOC

Discovery can be performed before or after a resource is obtained. Performing discovery ahead of accessing the resource allows a resource-consumer to learn more about the properties of the resource. For example, a consumer can learn about the protocols supported by the resource and if understood, utilize them to interact with it.

In many cases, discovery is performed after the resource has been obtained, based on the content of the resource and the way in which the user-agent interacts with it (or based on human interactions). Most web applications make strong assumptions about the resources they interact with, mostly due to lack of a standard discovery protocol for web resources. Such assumptions are not likely to disappear even with the introduction of a discovery workflow. In many cases, discovery will be used as a secondary step for enhancing the interaction with a resource rather than the first step of determining how to interact with it at all.
The focus of this memo is on the first step in discovery: identifying the location of the resource descriptors. The overall discovery workflow includes two additional steps:

1. The location of the resource descriptor document is obtained using the resource URI. It does not matter how the resource URI has been obtained, just that a URI is known. Once the descriptor location has been identified, the descriptor document is retrieved.

2. The resource descriptor document is parsed based on the descriptor document format used. For example, two such formats are POWDER (Protocol for Web Description Resources c) and XRD (Extensible Resource Descriptor [XRD] (Hammer-Lahav, E., Ed., "XRD 1.0," .) [[replace with new XRD specification reference]]).

3. The information about the resource contained within the descriptor document is processed to find out its capabilities, characteristics, and relationships to other resources. Capabilities are usually described with identifiers or description languages that the consuming application can match to a database of known capabilities or process via an interpreter.

While the process described in this memo utilizes the HTTP protocol [RFC2616] (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," June 1999.) for locating descriptors, it can be used with any URI scheme and is not limited to just the 'http' and 'https' URI schemes. HTTP is an ideal framework for performing discovery activities on web resources, but it does not clearly define a mechanism for attaching a descriptor or metadata to a resource identified with a URI.

TOC

6. 'describedby' Link Relationship

The first step when performing discovery is to identify the location of the resource descriptor document for the desired resource. This can be simply described as a link between the URI of the resource and the URI of the descriptor. Links are one of the most fundamental building blocks of the web, and provide all that is necessary to define the relationship between a resource and its descriptors.
The purpose of this memo is to define a consistent set of methods using HTTP through which this link information is obtained when performing discovery. The web provides a large number of methods for defining links between resources, but in order to achieve interoperability, the selection has to be narrowed down to a much smaller set of options. Since a single resource can have many descriptors, the descriptor link relationship has a one-to-many structure. In the case of multiple descriptors, selecting which descriptor to use is application-specific. It can involve factors such as the descriptor document format, accessibility, and other typed relationships, and as such is beyond the scope of this memo.
All the methods described in this memo build directly on the typed-relationships framework defined in [I-D.nottingham-http-link-header] (Nottingham, M., "Link Relations and HTTP Header Linking," November 2008.). The relationship type between a resource and its descriptor used for discovery is 'describedby' which was originally defined by [POWDER] (Archer, P., Ed., Smith, K., Ed., and A. Perego, Ed., "POWDER: Protocol for Web Description Resources," .) as a generic relationship type as follows:

> Relationship type: describedby
>
> Purpose: To link a resource to a description that applies to that resource
>
> Documentation: http://www.w3.org/TR/powder-dr/#assoc-linking
>
> Note: The relationship A 'describedby' B does not imply that B is a POWDER file (the Media Type does that), simply that B provides a description of A. This is the only constraint placed on A and B by asserting the describedby relationship.

[[NOTE: the link type 'describedby' Link Relationship has been submitted to IANA for review, approval, and inclusion in the Atom Link Relations registry. The Atom Link Relations registry is expected to be replaced by a generic Link Relations registry as defined in [I-D.nottingham-http-link-header] (Nottingham, M., "Link Relations and HTTP Header Linking," November 2008.) section 4.2.]].
For example, the following HTTP response header (fragment) returned with the HTTP representation of the resource http://example.com/resource/1:

```
HEAD /resource/1 HTTP/1.1
Server: example.com

Link: <http://example.com/resource/1;about>;
          rel="describedby"; type="application/xrd+xml"
```

defines a link between the resource http://example.com/resource/1 and its descriptor located at http://example.com/resource/1;about and is hinted to be using the XRD [XRD] (Hammer-Lahav, E., Ed., "XRD 1.0," .) document format.
The methods described in this memo all result in one or more link relationships with type 'describedby'. Two out of the three methods use existing link mechanisms as-is, by simply specifying the relationship type used. The third defines a new mechanism for dynamically constructing links using templates.

---

7. Method Selection

TOC

Due to the wide range of use cases requiring resource descriptors, and the desire to reuse as much as possible, no single solution has been found to sufficiently cover the requirements for linking between the resource URI and the descriptor URI. An analysis of the potential methods considered and the reason for their inclusion or rejection can be found in Appendix A (Method Suitability Analysis). A discussion regarding the architectural issues around discovery can be found in [Uniform Access] (Rees, J., "Uniform Access to Information About," .).
Obtaining the link information between the resource URI and the descriptor URI is accomplished using one of three methods. The criteria used to determine which methods a resource-provider SHOULD support and resource-consumer SHOULD attempt to use are based on a combination of factors:

- The document type of the available resource representation (text/html, application/atom+xml, image/png, unknown, etc.).
- The URI scheme (http, https, mailto, xmpp, etc.).
- The availability of an HTTP-accessible representation for the resource (a representation of the resource that can be retrieved using an HTTP GET request).
- The ability, desire, or applicability of the resource-consumer to directly interact and retrieve a resource representation (which might be unknown to it).

When selecting a method to use, the following requirement of each method are considered (each method is described in details in Section 8 (Obtaining Descriptor Location)):

*<LINK> Element: Limited to resources with an accessible markup representation with direct support for typed-relationships using the <LINK> element, such as HTML [W3C.REC-html401-19991224] (Hors, A., Jacobs, I., and D. Raggett, "HTML 4.01 Specification," December 1999.) and Atom [RFC4287] (Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format," December 2005.). Other document types are allowed as long as the semantics of their <LINK> element are fully compatible with the link framework defined in [I-D.nottingham-http-link-header] (Nottingham, M., "Link Relations and HTTP Header Linking," November 2008.). This method requires full retrieval of the resource representation before any discovery information about it is available. While HTTP is the most common transport for HTML and Atom documents, this method is transport independent.

*HTTP Link Header: Limited to resources with an accessible representation using the HTTP protocol [RFC2616] (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," June 1999.), or resources for which an HTTP GET or HEAD request returns a valid HTTP response header. This method uses the Link header defined in [I-D.nottingham-http-link-header] (Nottingham, M., "Link Relations and HTTP Header Linking," November 2008.). This method requires the retrieval of the resource representation header (using an HTTP GET or HEAD request).

*Site-Meta Document: A known-location based solution used for any resources identified by a URI with a DNS-resolvable authority component (i.e. an authority that can be directly mapped to an IP address). This method uses the Site-Meta document defined in [I-D.nottingham-site-meta] (Nottingham, M. and E. Hammer-Lahav, "draft-nottingham-site-meta-00," October 2008.). This method does not require any direct interaction with the resource.

The order in which the methods are listed is based on their applicability specialization, from the most restrictive method to the most generic method. This ordering however does not imply the order in which multiple applicable methods should be attempted (which is application specific). Because different methods are more appropriate in different circumstances, all three methods described are considered equal and can be attempted in any order. To ensure interoperability, the following rules MUST be observed:

*Resource-providers MUST support at least one of the three methods for each resource for which discovery information is to be made

available. If more than one method is supported, all methods MUST produce the same resource descriptor location (either by returning the same descriptor URI or a different descriptor URI that leads to the same descriptor URI after following HTTP redirections).

*Resource-consumers SHOULD support all three methods and attempt each in their preferred order until a descriptor URI is obtained successfully. Resource-consumers SHOULD NOT attempt additional methods after a previous method has concluded successfully.

---

8. Obtaining Descriptor Location

TOC

To obtain the location of the resource descriptor using the resource URI, the resource-consumer SHALL proceed as follows:

1. Select one of the three methods as defined in Section 7 (Method Selection). In many cases, only some of the methods will be applicable. If more than one method is available, the resource-consumer SHOULD pick the method most efficient for its needs.

2. Perform the steps described below for the selected method. If successful, the method will produce the descriptor location. If the method fails, repeat the process from the previous step by selecting another method. If no method is left, the discovery process fails.

3. Once the desired descriptor URI has been obtained, the descriptor document is obtained via an HTTP GET request to the identified URI. The resource-consumer MUST obey all HTTP 301 and 302 redirects and the descriptor document is considered valid only if contained within an HTTP response with the HTTP 200 response code.

8.1. <LINK> Element

TOC

Resources with an HTML [W3C.REC-html401-19991224] (Hors, A., Jacobs, I., and D. Raggett, “HTML 4.01 Specification,” December 1999.) or an Atom [RFC4287] (Nottingham, M., Ed. and R. Sayre, Ed., “The Atom Syndication Format,” December 2005.) representations MAY include a <LINK> element with the 'describedby' relationship type to link between the resource and its descriptor.

For example:

```
<LINK href="http://example.com/resource;about"
        rel="describedby" type="application/powder+xml">
```

A resource-consumer trying to obtain the location of the resource's descriptor using this method SHALL:

1. Retrieve a valid representation of the resource using the applicable transport for that resource URI. If the resource representation is obtained using HTTP, the resource-consumer MUST only use it if the HTTP response containing the representation carries a valid HTTP 200 response code. If any other response code is returned, the method fails. [[This is written specifically about the request producing the representation, ignoring any potential redirects that might have occurred prior. Should redirects be explicitly mentioned here?]]

2. Parse the document as defined by the document specification and look for <LINK> elements with a 'rel' attribute value containing the 'describedby' relationship (a multiple relationship 'rel' attribute value is allowed and MUST be handled by the consumer, for example 'rel="describedby copyright"').

3. The resource-consumer SHOULD examine any available 'type' attributes as hints for the document format used by the descriptor document. If more than one link is found, the descriptor mime-type SHOULD be used to narrow down the selection.

4. The descriptor location is obtained from the value of the 'href' attribute on the selected <LINK> element.

8.2. HTTP Link Header

TOC

Resources with an accessible HTTP representation MAY include a Link header in the HTTP response header as defined by [I-D.nottingham-http-link-header] (Nottingham, M., “Link Relations and HTTP Header Linking,” November 2008.) with a 'rel' parameter value set to 'describedby'.
For example:

```
Link: <http://example.com/resource;about>; rel="describedby";
          type="application/powder+xml"
```

A resource-consumer trying to obtain the location of the resource's descriptor using this method SHALL:

1. Retrieve a valid HTTP response header for the representation of the resource using an HTTP GET or HEAD request. The resource-consumer MUST follow HTTP redirections 301 and 302. The resulting header MUST only be used for the purpose of discovery if the HTTP response containing the header has one of the following HTTP response codes: 200, 303, and 401. If any other response code is returned, the method fails.

2. Parse the HTTP response header and look for a Link header with a 'rel' parameter value containing the 'describedby' relationship (a multiple relationship 'rel' parameter value is allowed and MUST be handled by the consumer, for example 'rel="describedby copyright";').

3. The resource-consumer SHOULD examine any available 'type' parameters as hints for the document format used by the descriptor document. If more than one link is found, the descriptor mime-type SHOULD be used to narrow down the selection.

4. The descriptor location is obtained from the URI-reference locating between the '<>' characters of the selected Link header.

If the HTTP response code is 303, any descriptor location is defined to be between the requested resource and the descriptor and not between the 'See other' resource indicated by a Location header. If the response code is 401, any descriptor location MUST only be used in association with obtaining access to the resource, which once obtained, must be queried again for its descriptor location which MAY be different from the unauthorized response.

8.3. Site-Meta Document

TOC

The descriptor location of resources identified with a URI which contains a DNS-resolvable authority component MAY be obtained from the Site-Meta document [I-D.nottingham-site-meta] (Nottingham, M. and E. Hammer-Lahav, “draft-nottingham-site-meta-00,” October 2008.) via a templatized map used to transform the resource URI to the descriptor URI. The templatized link is provided by an extension (defined by this

memo) to the Site-Meta schema for describing link templates. Using a template provided by Site-Meta for URIs under its authority, a resource URI can be deconstructed and then reconstructed to form the URI of the descriptor location.
For example, given the resource identified by http://example.com/r/1, the Site-Meta document for its authority example.com is obtained from http://example.com/site-meta. The Site-Meta document defines a template in which the resource URI is converted to the descriptor URI by appending ';about' to the URI:

```
<metadata>
  <link-template template="{uri};about"
          rel="describedby"
          type="application/powder+xml" />
</metadata>
```

resulting in the descriptor location URI http://example.com/r/1;about.

8.3.1. Site-Wide Links

TOC

Site-Meta defines a method for locating site-wide metadata for web sites. Its primary objective is to avoid the need of further known-location solutions by creating one last such resource which can point to other resources. It can be considered a registry for "known-location" resources to avoid further intrusion into the site's naming authority, hopefully the last such resource.
In the context of discovery, Site-Meta offers a convenient location for storing information about how to map between resource URIs and their descriptor URIs at an authority level. Site-Meta provides a method for obtaining descriptor locations that does not depend on the availability of an HTTP representation (or 303 See Other response) for resources. It can also, with an additional authority verification step (described in Section 8.3.3 (DNS Verification for Non-HTTP(S) URIs)), provide descriptor locations for URIs with schemes other than 'http' and 'https'.
The elements defined by Site-Meta are meant to contain site-wide information. Unlike Link headers included in the HTTP response to requests for the domain root resource (obtained via 'GET / HTTP/1.1' or 'HEAD / HTTP/1.1') which are specific to the root resource, links in Site-Meta are between the abstract 'web site' entity and the linked resources. It is critical not to confuse the root resource of a domain authority with the abstract 'web site' entity described by Site-Meta. For this reason, any <meta> elements containing linked resources with relationship type 'describedby', identify the location of the abstract 'web site' entity description which by itself cannot be described using

a URI. While this is a valid application of the 'describedby' relationship type, it is beyond the scope of this memo.

8.3.2. <link-template> Element

TOC

The <link-template> element is defined as a child of the root <metadata> element and identical to the Site-Meta <meta> element with the following differences:

- It cannot contain any value or child elements and must be self-closing.
- The 'href' attribute in the <meta> element is replaced by the 'template' attribute.
- The OPTIONAL 'scheme' attribute is added.
- The resulting relationship is defined as between the individual resource used as an input to the template and the resulting descriptor URI, and do not in relation to the abstract 'web site' entity.

The 'scheme' attribute serves as a filter indicating which URI scheme are meant to be transformed using the provided template. This OPTIONAL attribute is meant to allow different handling of different URI schemes. The attribute value is a space separated list of lowercase scheme names. If omitted, the template is meant to be applied to any URI schemes.
The 'template' attribute defines a URI template with a very simple syntax. The attribute value is used to construct a valid URI by substituting the variable enclosed in '{}' with the value of the variable.
In the example above, the 'uri' variable is replaced with the actual resource URI (the resource URI http://example.com replaces the '{uri}' string which results in http://example.com;about). If the variable name is prefixed by a '%' character, any character other than unreserved in variable value MUST be percent-encoded per [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” January 2005.).

```
unreserved  = ALPHA / DIGIT / "-" / "." / "_" / "~"
```

For example, the following template when used with the resource URI http://example.com:

```
<metadata>
  <link-template template="http://example.com?describe={%uri}"
          rel="describedby"
          type="application/xrd+xml"
          scheme="mailto http https" />
</metadata>
```

produces the descriptor URI: http://example.com?
describe=http%3A%2F%2Fexample.com.
[[This initial draft only defines a single 'uri' variable. However, it is expected that future revision will define a larger template vocabulary which will be based on the URI structure definition and include: uri, scheme, authority, domain, port, path, query, fragment, and username (for 'mailto' URIs).]]
[[Site-Meta is pending a major revision of its document format which is likely to replace its XML structure with a simpler text based structure. This memo will be revised as soon as the new Site-Meta draft is published to reflect these changes. However, the changes are expected to only change the document formatting.]]

8.3.3. DNS Verification for Non-HTTP(S) URIs

TOC

Site-Meta uses the HTTP protocol for providing metadata about the abstract 'web site' entity. This raises the issue whether an HTTP server can speak authoritatively for a non-HTTP resource, namely, a resource identified by a URI with a scheme other than 'http' or 'https'.
From a deployment perspective, many organizations separate the administration responsibilities of their HTTP resources from other resources such as email (SMTP) or instant messaging (XMPP). It would be an unexpected behavior in such organizations if the HTTP server provides authoritative information about identifiers belonging to other departments.
The process defined in this memo was design with ease of deployment as one of its top priorities. Since it is unlikely that protocols such as SMTP will introduce their own discovery extensions (which will realize any significant deployment in the foreseeable future), Site-Meta must be able to provide authoritative information regarding the descriptor location of non-HTTP(S) resources.
To obtain such authority, the owner of the domain (as represented by the administrator of its DNS records) MUST declare that Site-Meta is indeed allowed to provide such information. To do so, the domain DNS records MUST include, for each domain or sub-domain for which the HTTP server has such authority, a TXT record with the exact value of '/site-meta non-http delegation enabled'. [[The DNS record type and value are

included in this draft only as a straw man proposal and are likely to change based on feedback received from the DNS community. Proposals for such record are requested.]]
Resource-consumers MUST verify the existence of such DNS record before obtaining and utilizing the Site-Meta document for the discovery of non-HTTP(S) resources. If no such record is found, the method fails.

8.3.4. Method Workflow

TOC

A resource-consumer trying to obtain the location of the resource's descriptor using this method SHALL:

1. Examine the resource URI and extract its authority component as defined by [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," January 2005.) section 3:

   ```
   foo://example.com:8042/over/there?name=ferret#nose
   \_/   \______________/\_________/ \_________/ \__/
    |           |            |            |        |
   scheme    authority       path        query   fragment
   ```

   If the authority contains an '@' character, the '@' character and everything to its left is removed. For example, in the URI mailto:username@example.com the authority component username@example.com is stripped of the '@' character and the characters to its left, leaving example.com as the extracted authority value used in follow-up steps.

2. If the URI scheme being discovered is not 'http' or 'https', the resource-consumer MUST perform DNS verification as described in Section 8.3.3 (DNS Verification for Non-HTTP(S) URIs) to ensure that the HTTP protocol service for that domain has the authority to relay discovery location for other schemes.

3. Retrieve the Site-Meta document for the extracted authority as defined by [I-D.nottingham-site-meta] (Nottingham, M. and E. Hammer-Lahav, "draft-nottingham-site-meta-00," October 2008.) section 4, by making an HTTP GET request:

   ```
   GET /site-meta HTTP/1.1
   Host: example.com
   ```

   If the request fails to retrieve a valid Site-Meta document, the method fails. [[Should the method require the use of HTTPS

when retrieving the Site-Meta document when performing discovery on 'https' scheme URIs?]]

4. Parse Site-Meta document and look for <link-template> elements with a 'rel' attribute value containing the 'describedby' relationship (a multiple relationship 'rel' attribute value is allowed and MUST be handled by the consumer, for example 'rel="describedby copyright"').

5. The resource-consumer SHOULD examine any available 'type' attributes as hints for the document format used by the descriptor document. If more than one link template is found, the descriptor mime-type SHOULD be used to narrow down the selection.

6. The descriptor location is constructed by applying the template obtained from the 'template' attribute of the selected <link-template> element on the resource URI.

---

9. Caching

TOC

Resource-consumers MUST obey all HTTP caching headers and directives and discard any cached descriptor location as defined by the resource-provider. The ability to cache descriptor locations was a key requirement in selecting which methods to include in the discovery workflow. It is critical that such information is cached as defined by HTTP.

---

10. Security Considerations

TOC

The methods used to perform discovery are not secure, private or integrity-guaranteed, and due caution should be exercised when using them. Applications that perform discovery should consider the attack vectors opened by automatically following, trusting, or otherwise using links gathered from <LINK> elements, HTTP Link headers, or Site-Meta documents.

---

11. IANA Considerations

TOC

This memo includes no request to IANA. The relationship type 'describedby' used by this memo is pending approval by the IANA and

must be fully registered before this memo can become final. If for any reason the 'describedby' relationship type fails to register with the IANA, it is expected that this memo will define a new relationship type.

Appendix A. Method Suitability Analysis

TOC

The following analysis attempts to list all the method proposed for addressing resource discovery. It has been previously published as an article at [Discovery and HTTP] (Hammer-Lahav, E., "Discovery and HTTP," .) and is included here to provide background information as to why certain methods have been selected while others rejected from the discovery process. It has been updated to match the terms used in this memo and its structure.

Appendix A.1. Requirements

TOC

Getting from a resource URI to its descriptor document can be implemented in many ways. The problem is that none of the current methods address all of the requirements presented by the common use cases. The requirements are simple, but the more we try to address, the less elegant and accessible the process becomes. While working on the now defunct XRDS-Simple specification [XRDS-Simple] (Hammer-Lahav, E., "XRDS-Simple 1.0," .) and talking to companies and individual about it, the following requirements emerged for any proposed process:

Self Declaration:
: Allow resources to declare the availability of descriptor information and its location. When a resource is accessed, it needs to have a way to communicate to the resource-consumer that it supports the discovery protocol and to indicates the location of such descriptor.

  This is useful when the consumer is able or is already interacting with the resource but can enhance its interaction with additional information. For example, accessing a blog page enhanced if it was generated from an Atom feed or Atom entry and that feed supports Atom authoring.

Direct Descriptor Access:
: Enable direct retrieval of the resource descriptor without interacting with the resource itself. Before a resource is accessed, the resource-consumer should have a way to obtain the resource descriptor without accessing the resource.

This is important for two reasons.

First, accessing an unknown resource may have undesirable consequences. After all, the information contained in the descriptor is supposed to inform the consumer how to interact with the resource. The second is efficiency - removing the need to first obtain the resource in order to get its descriptor (reducing HTTP round-trips, network bandwidth, and application latency).

Web Architecture Compliant:
Work with well-established web infrastructure. This may sound obvious but it is in fact the most complex requirement. Deploying new extensions to the HTTP protocol is a complicated endeavor. Beside getting applications to support a new header, method, or content negotiation, existing caches and proxies must be enhanced to properly handle these requests, and they must not fail performing their normal duties without such enhancements.

For example, a new content negotiation method may cause an existing cache to serve the wrong data to a non-discovery consumer due to its inability to distinguish the metadata request from the resource representation request.

Scale and Technology Agnostic:
Support large and small web providers regardless of the size of operations and deployment. Any solution must work for a small hosted web site as well as the world largest search engine. It must be flexible enough to allow developers with restricted access to the full HTTP protocol (such as limited access to request or response headers) to be able to both provide and consume resource descriptors. Any solution should also support caching as much as possible and allow reuse of source code and data.

Extensible:
Accommodate future enhancements and unknown descriptor formats. It should support the existing set of descriptor formats such as XRD and POWDER, as well as new descriptor relationships that might emerge in the future. In addition, the solution should not depend on the descriptor format itself and work equally well with any document format - it should aim to keep the road and destination separate.

TOC

Appendix A.2. Analysis

The following is a list of proposed and implemented methods trying to address resource discovery. Each method is reviewed for its compliance with the requirements identified previously. The [-], [+], or [+-] symbols next to each requirement indicate how well the method complies with the requirement.

Appendix A.2.1. HTTP Response Header

TOC

When a resource representation is retrieved using and HTTP GET request, the server includes in the response a header pointing to the location of the descriptor document. For example, POWDER uses the 'Link' response header to create an association between the resource and its descriptor. XRDS [XRDS] (Wachob, G., Reed, D., Chasen, L., Tan, W., and S. Churchill, "Extensible Resource Identifier (XRI) Resolution V2.0," .) (based on the Yadis protocol [Yadis] (Miller, J., "Yadis Specification 1.0," .)) uses a similar approach, but since the Link header was not available when Yadis was first drafted, it defines a custom header X-XRDS-Location which serves a similar but less generic purpose.

> **[+] Self Declaration -** using the Link header, any resource can point to its descriptor documents.
>
> **[-] Direct Descriptor Access -** the header is only accessible when requesting the resource itself via an HTTP GET request. While HTTP GET is meant to be a safe operation, it is still possible for some resource to have side-effects.
>
> **[+] Web Architecture Compliant -** uses the Link header which is an IETF Internet Standard [[Currently a standard-track draft]], and is consistent with HTTP protocol design.
>
> **[-] Scale and Technology Agnostic -** since discovery accounts for a small percent of resource requests, the extra Link header is wasteful. For some hosted servers, access to HTTP headers is limited and will prevent implementation.
>
> **[+] Extensible -** the Link header provides built-in extensibility by allowing new link relationships, mime-types, and other extensions.

Minimum roundtrips to retrieve the resource descriptor: 2

Appendix A.2.2. HTTP Response Header Via HEAD

TOC

Same as the HTTP Response Header method but used with an HTTP HEAD request. The idea of using the HEAD method is to solve the wasteful overhead of including the Link header in every reply. By limiting the appearance of the Link header only to HEAD responses, typical GET requests are not encumbered by the extra bytes.

> **[+] Self Declaration -** Same as the HTTP Response Header method.
>
> **[-] Direct Descriptor Access -** Same as the HTTP Response Header method.
>
> **[-] Web Architecture Compliant -** HTTP HEAD should return the exact same response as HTTP GET with the sole exception that the response body is omitted. By adding headers only to the HEAD response, this solution violates the HTTP protocol and might not work properly with proxies as they can return the header of the cached GET request.
>
> **[+] Scale and Technology Agnostic -** solves the wasted bandwidth associated with the HTTP Response Header method, but still suffers from the limitation imposed by requiring access to HTTP headers.
>
> **[+] Extensible -** Same as the HTTP Response Header method.

Minimum roundtrips to retrieve the resource descriptor: 2

Appendix A.2.3. HTTP Content Negotiation

TOC

Using the Accept request header, the consumer informs the server it is interested in the descriptor and not the resource itself, to which the server responds with the descriptor document or its location. In Yadis, the consumer sends an HTTP GET (or HEAD) request to the resource URI with an Accept header and content-type application/xrds+xml. This informs the server of the consumer's discovery interest, which in turn may reply with the descriptor document itself, redirect to it, or return its location via the X-XRDS-Location response header.

> **[-] Self Declaration -** does not address as it focuses on the consumer declaring its intentions.
>
> **[+] Direct Descriptor Access -** provides a simple method for directly requesting the descriptor document.

[-] Web Architecture Compliant - while it can be argued that the descriptor can be considered another representation of the resource, it is very much external to it. Using the Accept header to request a separate resource (as opposed to a different representation of the same resource) violates web architecture. It also prevents using the discovery content-type as a valid (self-standing) web resource having its own descriptor.

[-] Scale and Technology Agnostic - requires access to HTTP request and response headers, as well as the registration of multiple handlers for the same resource URI based on the Accept header. In addition, improper use or implementation of the Vary header in conjunction with the Accept header will cause caches to serve the descriptor document instead of the resource itself - a great concern to large providers with frequently visited front-pages.

[-] Extensible - applies an implicit relationship type to the descriptor mime-type, limiting descriptor formats to a single purpose. It also prevents using existing mime-types from being used as a descriptor format.

Minimum roundtrips to retrieve the resource descriptor: 1

Appendix A.2.4. HTTP Header Negotiation

TOC

Similar to the HTTP Content Negotiation method, this solution uses a custom HTTP request header to inform the server of the consumer's discovery intentions. The server responds by serving the same resource representation (via an HTTP GET or HEAD requests) with the relevant Link headers. It attempts to solve the HTTP Response Header waste issue by allowing the consumer to explicitly request the inclusion of Link headers. One such header can be called 'Request-links' to inform the server the consumer would like it to include certain Link headers of a given 'rel' type in its reply.

[+] Self Declaration - same as HTTP Response Header with the option of selective inclusion.

[-] Direct Descriptor Access - does not address.

[-] Web Architecture Compliant - HTTP does not include any mechanism for header negotiation and any custom solution will break existing caches.

[+-] Scale and Technology Agnostic - Requires advance access to HTTP headers on both the consumer and provider sides, but solves the bandwidth waste issue of the HTTP Response Header method.

[+] Extensible - builds on top of Link header extensibility.

Minimum roundtrips to retrieve the resource descriptor: 2

Appendix A.2.5. <Link> Element

TOC

Embeds the location of the descriptor document within the resource representation by leveraging the HTML <Link> header element (as opposed to the HTTP header). Applies to HTML resource representations or similar markup-based formats with support for 'Link'-like elements such as Atom. POWDER uses the <Link> element in this manner, while XRDS uses the HTML <meta> element with an 'http-equiv' attribute equals to X-XRDS-Location (to create an embedded version of the X-XRDS-Location custom header).

[+] Self Declaration - similar to HTTP Response Header method but limited to HTML resources.

[-] Direct Descriptor Access - the method requires fetching the entire resource representation in order to obtain the descriptor location. In addition, it requires changing the resource HTML representation which makes discovery an intrusive process.

[+] Web Architecture Compliant - uses the <Link> element as designed.

[+] Scale and Technology Agnostic - while this solution requires direct retrieval of the resource and manipulation of its content, it is extremely accessible in many platforms.

[-] Extensible - extensibility is restricted to HTML representations or similar markup formats with support for a similar element.

Minimum roundtrips to retrieve the resource descriptor: 2

Appendix A.2.6. HTTP OPTIONS Method

TOC

The HTTP OPTIONS method is used to interact with the HTTP server with regard to its capabilities and communication-related information about its resources. The OPTIONS method, together with an optional request header, can be used to request both the descriptor location and descriptor content itself.

> **[-] Self Declaration -** does not address.
>
> **[+] Direct Descriptor Access -** provides a clean mechanism for requesting descriptor information about a resource without interacting with it.
>
> **[+] Web Architecture Compliant -** uses an existing HTTP featured.
>
> **[-] Scale and Technology Agnostic -** requires consumer and provider access to the OPTIONS HTTP method. Also does not support caching which makes this solution inefficient.
>
> **[+] Extensible -** built-into the OPTIONS method.

Minimum roundtrips to retrieve the resource descriptor: 1

Appendix A.2.7. WebDAV PROPFIND Method

TOC

Similar to the HTTP OPTIONS method, the WebDAV PROPFIND method defined in [RFC4918] (Dusseault, L., “HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV),” June 2007.) can be used to request resource specific properties, one of which can hold the location of the descriptor document. PROPFIND, unlike OPTIONS, cannot return the descriptor itself, unless it is returned in the required PROPFIND schema (a multi-status XML element). Other alternatives include URIQA [URIQA] (Nokia, “The URI Query Agent Model,” .), an HTTP extension which defines a method called MGET, and ARK (Archival Resource Key) [ARK] (Kunze, J. and R. Rodgers, “The ARK Identifier Scheme,” .) - a method similar to PROPFIND that allows the retrieval of resource attributes using keys (which describe the resource).

[-] Self Declaration - does not address.

[+-] Direct Descriptor Access - does not require interaction with the resource, but does require at least two requests to get the descriptor (get location, get document).

[+] Web Architecture Compliant - uses an HTTP extension with less support than core HTTP, but still based on published standards.

[-] Scale and Technology Agnostic - same as the HTTP OPTIONS Method.

[+-] Extensible - uses extensible protocols but at the same time depends on solutions that have already gone beyond the standard HTTP protocol, which makes further extensions more complex and unsupported.

Minimum roundtrips to retrieve the resource descriptor: 2

Appendix A.2.8. Custom HTTP Method

TOC

Similar to the HTTP OPTIONS Method, a new method can be defined (such as DISCOVER) to return (or redirect to) the descriptor document. The new method can allow caching.

[-] Self Declaration - does not address.

[+] Direct Descriptor Access - same as the HTTP OPTIONS Method.

[-] Web Architecture Compliant - depends heavily on extending every platform to support the extension. Unlikely to be supported by existing proxy services and caches.

[-] Scale and Technology Agnostic - same as HTTP OPTIONS Method with the additional burden on smaller sites requiring access to the new protocol.

[+] Extensible - new protocol that can extend as needed.

Minimum roundtrips to retrieve the resource descriptor: 1

TOC

Appendix A.2.9. Static Resource URI Transformation

Instead of using HTTP facilities to access the descriptor location, this method defines a template to transform any resource URI to the descriptor document URI. This can be done by adding a prefix or suffix to the resource URI, which turns it into a new resource URI. The new URI points to the descriptor document. For example, to fetch the descriptor document for http://example.com/resource, the consumer makes an HTTP GET request to http://example.com/resource;about using a static template that adds the ';about' suffix.

> **[-] Self Declaration -** does not address.
>
> **[+] Direct Descriptor Access -** creates a unique URI for the descriptor document.
>
> **[+-] Web Architecture Compliant -** uses basic HTTP facilities but intrudes on the domain authority namespace as it defines a static template for URI transformation that is not likely to be compatible with many existing URI naming conventions.
>
> **[+-] Scale and Technology Agnostic -** depending on the static mapping chosen. Some hosted environment will have a problem gaining access to the mapped URI based on the URI format chosen.
>
> **[-] Extensible -** provides a very specific and limited method to map between resources and their descriptor, since each relationship type must mint its own static template.

Minimum roundtrips to retrieve the resource descriptor: 1

Appendix A.2.10. Dynamic Resource URI Transformation

TOC

Same as the Static Resource URI Transformation method but with the ability for each domain authority to specify its own discovery transformation template. This can done by placing a configuration file at a known location (such as robots.txt) which contains the template needed to perform the URL mapping. The consumer first obtains the configuration document (which may be cached using normal HTTP facilities), parses it, then uses that information to transform the resource URI and access the descriptor document.

[+-] Self Declaration - does not address individual resources, but allows entire domains to declare their support (and how to use it).

[+-] Direct Descriptor Access - once the mapping template has been obtained, descriptors can be accessed directly.

[+-] Web Architecture Compliant - uses an existing known-location design pattern (such as robots.txt) and standard HTTP facilities. The use of a known-location if not ideal and is considered a violation of web architecture but if it serves as the last of its kind, can be tolerated. An alternative to the known-location approach can be using DNS to store either the location of the mapping or the map template itself, but DNS adds a layer of complexity not always available.

[+-] Scale and Technology Agnostic - works well at the URI authority level (domain) but is inefficient at the URI path level (resource path) and harder to implement when different paths within the same domain need to use different templates. With the decreasing cost of custom domains and sub-domains hosting, this will not be an issue for most services, but it does require sharing configuration at the domain/sub-domain level.

[+-] Extensible - can be, depending on the schema used to format the known-location configuration document.

Minimum roundtrips to retrieve the resource descriptor: initially 2, 1 after caching

---

Appendix B. Acknowledgments

TOC

With the exception of the Site-Meta template extension, very little of this memo is original work. Many communities and individuals have been working on solving discovery for many years and this work is a direct result of their hard and dedicated efforts.
Inspiration for this memo derived from previous work on a descriptor format called XRDS-Simple, which in turn derived from another descriptor format, XRDS. Previous discovery workflows include Yadis which is currently used by the OpenID community. While suffering from significant shortcomings, Yadis was a breakthrough approach to performing discovery using extremely restricted hosting environments, and this memo has strived to preserve as much of that spirit as possible.
The use of Link elements and headers and the introduction of the 'describedby' relationship type in this memo is a direct result of the

dedicated work and contribution of Phil Archer to the W3C POWDER specification and Jonathan Rees to the W3C review of Uniform Access to Information About. The Site-Meta approach was first proposed by Mark Nottingham as an alternative to attaching links directly to resource representations.
The author wishes to thanks the OASIS XRI community for their support, encouragement, and enthusiasm for this work. Special thanks go to Lisa Dusseault, Mark Nottingham, Drummond Reed, John Panzer, and Joseph Holsten for their invaluable feedback.
The author takes all responsibility for errors and omissions.

12. References

TOC

12.1. Normative References

TOC

[I-D.nottingham-http-link-header]	Nottingham, M., “Link Relations and HTTP Header Linking,” draft-nottingham-http-link-header-03 (work in progress), November 2008 (TXT).
[I-D.nottingham-site-meta]	Nottingham, M. and E. Hammer-Lahav, “draft-nottingham-site-meta-00,” draft-nottingham-site-meta-00 (work in progress), October 2008 (TXT).
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2616]	Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” RFC 2616, June 1999 (TXT, PS, PDF, HTML, XML).
[RFC3986]	Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” STD 66, RFC 3986, January 2005 (TXT, HTML, XML).
[RFC4287]	Nottingham, M., Ed. and R. Sayre, Ed., “The Atom Syndication Format,” RFC 4287, December 2005 (TXT, HTML, XML).
[RFC4918]	Dusseault, L., “HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV),” RFC 4918, June 2007 (TXT).
[W3C.REC-html401-19991224]	Hors, A., Jacobs, I., and D. Raggett, “HTML 4.01 Specification,” World Wide Web Consortium Recommendation REC-html401-19991224, December 1999 (HTML).

12.2. Informative References

TOC

[ARK]	Kunze, J. and R. Rodgers, “The ARK Identifier Scheme” (HTML).
[Discovery and HTTP]	Hammer-Lahav, E., “Discovery and HTTP” (HTML).
[POWDER]	Archer, P., Ed., Smith, K., Ed., and A. Perego, Ed., “POWDER: Protocol for Web Description Resources” (HTML).
[URIQA]	Nokia, “The URI Query Agent Model” (HTML).
[Uniform Access]	Rees, J., “Uniform Access to Information About” (HTML).
[XRD]	Hammer-Lahav, E., Ed., “XRD 1.0.”
[XRDS]	Wachob, G., Reed, D., Chasen, L., Tan, W., and S. Churchill, “Extensible Resource Identifier (XRI) Resolution V2.0” (HTML, PDF).
[XRDS-Simple]	Hammer-Lahav, E., “XRDS-Simple 1.0” (HTML).
[Yadis]	Miller, J., “Yadis Specification 1.0” (PDF, ODT).

Author's Address

TOC

	Eran Hammer-Lahav
	Yahoo!
Email:	eran@hueniverse.com
URI:	http://hueniverse.com