Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 17, 2014

I. Young, Ed.
Independent

August 16, 2013

Metadata Query Protocol
draft-young-md-query-00

Abstract

This document defines a simple protocol for retrieving metadata about entities. The goal of the protocol is to profile various aspects of HTTP to allow requesters to rely on certain, rigorously defined, behaviour.

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 February 17, 2014.

Table of Contents

1. Introduction

Many clients of web-based services are capable of consuming descriptive metadata about a service in order to customize or information the client's connection parameters. While the form of the metadata (e.g.. JSON, XML) and content varies between services this document attempts to specifies a set of semantics for HTTP [RFC2616] that allow clients to rely on certain behavior. The defined behavior is meant to make it easy for clients to perform queries, to be efficient for both requesters and responders, and allow the responder to scale in various ways.

1.1. Notation and Convention

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 [RFC2119].

1.2. Terminology

entity - A single logical construct for which metadata may be asserted. Generally this is a network accessible service.

metadata - A machine readable description of certain entity characteristics. Generally metadata provides information such as end point references, service contact information, etc.

2. Protocol Transport

The metadata retrieval protocol seeks to fully employ the features of the HTTP protocol. Additionally this specification makes mandatory some optional HTTP features.

2.1. HTTP Version

Metadata retrieval protocol responders MUST use HTTP, version 1.1 [RFC2616]

2.2. HTTP Method

All metadata retrieval requests MUST use the GET method.

2.3. Request Headers

All metadata retrieval requests MUST include the following HTTP headers:

- Accept - this header MUST contain the content-type identifying the type, or form, of metadata to be retrieved

All metadata retrieval requests SHOULD include the following HTTP headers:

- Accept-Charset
- Accept-Encoding

A metadata request to the same URL, after an initial request, MUST include the following header per section 13.3.4 of RFC2616 [RFC2616]:

- If-None-Match

2.4. Response Headers

All successful metadata retrieval responses (even those that return no results) MUST include the following headers:

- Content-Encoding - required if, and only if, content is compressed
- Content-Type
- ETag

All metadata retrieval responses SHOULD include the following headers:

Cache-Control

Content-Length

Last-Modified

2.5. Status Codes

This protocol uses the following HTTP status codes:

200 - standard response code when returning requested metadata

304 - response code indicating requested metadata has not been updated since the last request

400 - response code indicating that the requester's request was malformed in some fashion

401 - response code indicating the request must be authenticated before requesting metadata

404 - indicates that the requested metadata could not be found; this MUST NOT be used in order to indicate a general service error.

405 - response code indicating that a non-GET method was used

406 - response code indicating that metadata is not available in the request content-type

500 - standard response code when something goes wrong within the responder

501 - response code indicating that a given identifier transformation is not supported

505 - response code indicating that HTTP/1.1 was not used

2.6. Base URL

Requests defined in this document are performed by issuing an HTTP GET request to a particular URL. The final component of the path to which requests are issued is defined by the requests specified within this document. A base URL precedes such paths. Such a base URL MUST contain at least the scheme and host name components. It MAY also include a port as well as a path. It MUST NOT include URL fragments. If a path is included the path required by the particular defined request is appended to the path in the base URL.

2.7. Content Negotiation

As there may be many representations for a given piece of metadata, agent-driven content negotiation is used to ensure the proper representation is delivered to the requester. In addition to the required usage of the Accept header a response SHOULD also support the use of the Accept-Charset header.

3. Metadata Query Protocol

The metadata query protocol retrieves metadata based on one or more "tag" or "keyword" identifiers. A request may return information for none, one, or a collection of entities.

3.1. Identifiers

The query protocol uses identifiers to "tag" metadata for single- and multi-entity metadata collections. An identifier MAY contain any URL-encodable character but MUST NOT start with '{' (ASCII 0x7B) as this character has a special meaning in the first position (see below). The assignment of such identifiers to a particular metadata document is the responsibility of the query responder. If a metadata collection already contains a well known identifier it is RECOMMENDED that such a natural identifier is used when possible. Any given metadata collection MAY have more than one identifier associated with it.

3.1.1. Transforms

In some cases it may be advantageous to query for metadata using a transformed identifier. For example, some protocols will transmit hashed entity identifiers. This may be done to reduce the overall size of the identifier, escape special characters, obfuscate the identifier, etc.

A transformed identifier is represented by pre-pending the identifier with '{' + transformation indicator + '}'. The transformation indicator MUST be composed exclusively of printable ASCII characters (0x21-0x7E) excluding '{' (0x7B) and '}' (0x7D). Such an identifier need only make sense in the context within which it is used. Responders MUST support the MD5 (transformation indicator 'md5') and SHA1 (transformation indicator 'sha1') hashing algorithms as identifier transformations. The responder MAY support other transformation indicators.

For example, the identifier
http://example.org/service
transformed by means of MD5 hashing would become
{md5}f3678248a29ab8e8e5b1b00bee4060e0

3.2. Protocol

3.2.1. Request

A Metadata Query request is performed by issuing an HTTP GET request. All Metadata Query requests MUST use the URL format:

<base_url>/entities/{ID}+{ID}+...
The request MUST contain at least one identifier but MAY contain more than one. Each identifier must be properly URL encoded. If more than one identifier is used the returned metadata MUST have been labelled with each identifier. That is to say a search with multiple identifiers results in the intersection of the metadata that would be retrieved by searching for each identifier individually.

3.2.2. Response

The response to a Metadata Query request MUST be a document that provides metadata for the given request identifiers in the format described by the request's Content-Type header. Note, in the event that multiple identifiers were used in the request, it is the responder's responsibility to ensure that the metadata returned is valid. If the responder can not create a valid document it MUST respond with a 500 status code. An example of such an error would be the case where the result of the query is metadata for multiple entities but the request content type does not support returning multiple results in a single document.

3.2.3. Example Request and Response

```
GET /service/entities/http%3A%2F%2Fexample.org%2Fidp HTTP/1.1
Host: metadata.example.org
Accept: application/samlmetadata+xml
```

Example Metadata Query Request

```
HTTP/1.x 200 OK
Content-Type: application/samlmetadata+xml
ETag: abcdefg
Last-Modified: Thu, 15 Apr 2010 12:45:26 GMT
Content-Length: 1234

<?xml version="1.0" encoding="UTF-8"?>
<EntityDescriptor entityID="http://example.org/idp"
    xmlns="urn:oasis:names:tc:SAML:2.0:metadata">
....
```

Example Metadata Query Response

4. Efficient Retrieval and Caching

4.1. Conditional Retrieval

Upon a successful response the responder is required to return an ETag header and may return a Last-Modified header as well. Requesters SHOULD user either or both, with the ETag being preferred, in any subsequent requests for the same resource. In the event that a resource has not changed since the previous request, the requester will receive a 304 (Not Modified) status code as a response.

4.2. Content Caching

Responders SHOULD include cache control information with successful (200 status code) responses, assuming the responder knows when retrieved metadata is meant to expire. The responder should also include cache control information with 404 Not Found responses. This allows the requester to create and maintain a negative-response cache. When cache controls are used only the 'max-age' directive SHOULD be used.

4.3. Content Compression

As should be apparent from the required request and response headers this protocol encourages the use of content compression. This is in recognition that some metadata documents can be quite large or fetched with relatively high frequency.

Requesters SHOULD support, and advertise support for, gzip compression unless such usage would put exceptional demands on constrained environments. Responders MUST support gzip compression. Requesters and responders MAY support other compression algorithms.

5. Security Considerations

5.1. Integrity

As metadata often contains information necessary for the secure operation of interacting services it is RECOMMENDED that some form of content integrity checking be performed. This may include the use of SSL/TLS at the transport layer, digital signatures present within the metadata document, or any other such mechanism.

5.2. Confidentiality

In many cases service metadata is public information and therefore confidentiality is not required. In the cases where such functionality is required, it is RECOMMENDED that both the requester and responder support SSL/TLS. Other mechanisms, such as XML encryption, MAY also be supported.

5.3. Authentication

All responders which require client authentication to view retrieved information MUST support the use of HTTP basic authentication over SSL/TLS. Responders SHOULD also support the use of X.509 client certificate authentication.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

The editor would like to acknowledge the following individuals for their contributions to this document:

Scott Cantor (The Ohio State University)

Leif Johansson (SUNET)

Thomas Lenggenhager (SWITCH)

Special acknowledgement is due to Chad LaJoie (Covisint) for his work in editing previous versions of this specification.

8. Normative References

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

[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.

Author's Address

Ian A. Young (editor)
Independent

Email: ian@iay.org.uk