Protocol for the Syndication of Semantic Descriptions

Graham Moore[1], Marc Wilhelm Küster[2]

[1]Networked Planet
graham.moore@networkedplanet.com

[2]FH Worms – University of Applied Sciences
kuester@fh-worms.de

Abstract. This document describes a protocol to be used for the exchange of semantic descriptions. The protocol defines how a web service can publish a series of web accessible feeds that describe snapshots and changes to a collection of semantic descriptions. This protocol also defines how a client should process the feeds provided by the service such that a local store is in sync. A client can synchronize with more than one server to act as an aggregator for semantic descriptions.

Overview

This document describes a protocol to be used for the exchange of semantic descriptions. The protocol defines how a web service can publish a series of web accessible feeds that describe snapshots and changes to a collection of semantic descriptions. This protocol also defines how a client should process the feeds provided by the service such that a local store is in sync. A client can synchronize with more than one server to act as an aggregator for semantic descriptions.

The current version of the specification, which is still subject to revision in the standardization process, is available at http://www.egovpt.org/fg/CWA_Part_1b

Scope

This document specifies the underlying syndication protocol for the exchange of information about semantic descriptions. The protocol conforms to the Atom Syndication Format and the TMDM syntax and model. It defines several layers

Maicher, L.; Garshol, L. M. (eds.): *Subject-centric computing. Fourth International Conference on Topic Maps Research and Applications, TMRA 2008, Leipzig, Germany, October 16-17, 2008, Revised Selected Papers.* (Leipziger Beiträge zur Informatik: XII) - ISBN 978-3-941152-05-2

of syndication feeds that must be provided by a conforming application. Finally it defines algorithms for the providion and processing of the different feeds on the server and on the client.

Concepts

This protocol defines how a server can produce a number of Atom feeds that describe either a list of topic maps that the server manages, a list of snapshots of a given topic map or a list of topic map fragments. Each fragment is created because it is the new representation of a topic that has changed.

A client that wishes to maintain a topic map in sync with one held on the server first fetches the most recent snapshot for the required map and stores it locally. It then subscribes to the feed of 'changes' for that map. This feed lists topicmap fragments. A fragment is created and an entry added to the feed when a topic is updated, added or deleted from the topic map. The client then updates its local topicmap with the new topic representation.

This protocol defines the structure of the feeds published by the server and how a client should interpret and process these feeds.

Note: the notion of a client and server is solely defined by the responsibilities of each. Thus a given machine can act both as client and server (peer-to-peer scenario) or restrict itself to exactly one of the roles (publish-subscribe scenario).

Protocol

Terminology

Server Node: A node hosting both feed and data services that allow a client to understand the state of the topic map being managed over time.

Client Node: A node that subscribes to one or more server nodes and implements the update semantics defined in this protocol.

Server Contract

A server node is responsible for providing information about the state of the topic map(s) it is managing. It provides a number of feeds that allow clients to see which aspects of the map has changed over time and data services that allow a client to fetch representations of the topic map or individual topic instances in order to update a client environment.

Feeds & Data Services

A compliant server will provide the following Atom 1.0 feeds, fragment data services and snapshot data services.

Topic Maps Feed **-** a list of all the topic maps being managed by the server.

Example Service URL: `[server]/topicmaps`

Example Invocation:

```
Server: http://tmshare.networkedplanet.com
GET /topicmaps
```

The Atom payload of the topic maps feed contains an entry for each topic map. Each entry has a link to an Atom feed for the specified topic map.

Example response body:

```
<?xml version="1.0" encoding="utf-8"?>
  <feed xmlns="http://www.w3.org/2005/Atom">
  <title>Topic Maps managed by
         tmshare.networkedplanet.com</title>
  <link href="http://tmshare.networkedplanet.com/"/>
  <updated>2008-12-13T18:30:02Z</updated>

  <author>
    <name>TMShare Server</name>
  </author>

  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
  <!-- topic map entry -->
  <entry>
    <title>eGovernment Resources Topic Map</title>
    <!-- a link to the atom feed that has entries linking to
         the snapshots and fragments feed for this map -->
    <link rel="topicmapfeed" type="application/atom+xml"
          href="http://tmshare.networkedplanet.com/topicmaps/
                egov"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2008-12-13T18:30:02Z</updated>
    <summary>A topic map that contains the classification of
             eGovernment services.</summary>
  </entry>
```

```
    <!-- an entry follows for each topic map being exposed.
                 ...
    -->
  </feed>
```

Topic Map Feed **-** a feed for a given topic map that provides exactly two entries, one linking to a snapshots feed and one to a fragments feed.

Example Service URL: `[server]/topicmaps/egov`

Example Invocation:

```
  Server: http://psi.egovpt.org
  GET /topicmaps/egov
```

The Atom payload of the response contains the two entries, one that links to a feed with all snapshots of the topic map (`rel`-attribute of the link is `snapshotfeed`), and another one that links to aan Atom feed that lists changes to the topic map (`rel`-attribute of the link is `fragmentfeed`). Optionally, both links can be duplicated in the entries with `rel`-attributes that have the value `alternate`. This helps Atom feed readers to correctly display the links.

```
  <a:feed xmlns:a="http://www.w3.org/2005/Atom">
    <a:title>eGov TM</a:title>
    <a:updated>2008-09-26T11:13:40-01:00</a:updated>
    <a:subtitle>Feeds around the eGov TM</a:subtitle>
    <a:id>http://http://psi.egovpt.org/feeds/testtm/</a:id>
    <a:author>
      <a:name>Isidor</a:name>
    </a:author>
    <a:link href="http://http://psi.egovpt.org/feeds/testtm/"
  rel="self"/>
    <a:entry>
      <a:title>eGov TM: Fragments</a:title>
      <a:id>http://http://psi.egovpt.org/testtm/fragments/
            </a:id>
      <a:updated>2008-09-11T17:58:39-01:00</a:updated>
      <a:author>
        <a:name>Isidor</a:name>
      </a:author>
      <a:link
          href="http://http://psi.egovpt.org/testtm/fragments/"
          rel="alternate" type="application/atom+xml"/>
      <a:link
          href="http://http://psi.egovpt.org/testtm/fragments/"
          rel="fragmentfeed" type="application/atom+xml"/>
    </a:entry>
    <a:entry>
      <a:title>eGov TM: Snapshots</a:title>
    <a:id>http://http://psi.egovpt.org/testtm/snapshots/
          </a:id>
    <a:updated>2008-09-11T17:58:39-01:00</a:updated>
    <a:author>
      <a:name>Isidor</a:name>
    </a:author>
    <a:link
       href="http://http://psi.egovpt.org/testtm/snapshots/"
       rel="alternate" type="application/atom+xml"/>
    <a:link
       href="http://http://psi.egovpt.org/testtm/snapshots/"
       rel="snapshotfeed" type="application/atom+xml"/>
  </a:entry>
</a:feed>
```

Snapshot Feed - a list of all the representations of a given topic map over time. At present, XTM 1.0 representations are supported.

Example Service URL: `[server]/topicmaps/egov/snapshots`

Example Invocation:

```
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom"
 xmlns:tmshare="http://www.egovpt.org/tmshare">
  <title>The Snapshots of the eGovernment
         Resources Topic Map</title>
  <subtitle>A list of all XTM representations of
            this map</subtitle>
  <author>
    <name>TMShare Server</name>
  </author>
  <updated>2008-07-17T12:15:07.020071Z</updated>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>

  <tmshare:ServerSrcLocatorPrefix>http://psi.networkedplanet.
  com/</tmshare:ServerSrcLocatorPrefix>

  <!-- a link to the feed -->
  <link rel="self"
        href="http://tmshare.networkedplanet.com/topicmaps/
               egov/snapshots"/>
  <entry>
    <title>Snapshot 2008-07-17</title>
    <updated>2008-07-17T14:04:42.205299Z</updated>
    <!-- a link to the XTM 1.0 snapshot -->
    <link rel="topicmapdata" type="application/xtm1+xml"
          href="http://tmshare.networkedplanet.com/topicmaps
                 /egov/shapshots/60a76c80-d300-11d9-b93C-
                 0003939e0af6"/>
    <id>60a76c80-d300-11d9-b93C-0003939e0af6</id>
    </entry>
    <!-- an entry follows for each XTM snapshot being exposed.
                 ...
    -->
  </feed>
```

Topic Map Fragments Feed **-** a list of topic map fragments that indicate changes for a given topic map over a period of time.

Example Service URL: `[server]/topicmaps/egov/fragments`

Example Invocation:

```
  Server: http://tmshare.networkedplanet.com
  GET /topicmaps/egov/fragments
```

Example response body:

The Atom payload contains an entry for each fragment. Each entry contains one link to the fragment and the updated element contains the time at which the fragment was created. In addition to the standard Atom elements this protocol introduces two new elements.

The new elements are:

`<ServerSourceLocatorPrefix>`: Indicates to a client the prefix to use to locate topic properties that should be removed when updating a topic. This element should occur once as a child element of the `<feed>` and before the first entry. (See the fragment update algorithm below for more information).

And

`<TopicSI>`: Indicates to a client which topic is being updated from all those present in the fragment. This element MUST occur once as a child element of each `<entry>`.(See the fragment update algorithm below for more information).

```
  <?xml version="1.0"?>
  <feed xmlns="http://www.w3.org/2005/Atom"
  xmlns:tmshare="http://www.egovpt.org/tmshare">
    <title>Change fragments from the eGovernment Resources
           Topic Map</title>
    <author>
      <name>TMShare Server</name>
    </author>
    <updated>2008-07-17T15:47:17.062211Z</updated>
    <id>28C5DBD8-652A-4617-8C4A-C0FFC49B4475</id>
    <!-- The serversrclocatorprefix is used by a client
     to know the providence of topic map constructs. -->

  <tmshare:ServerSrcLocatorPrefix>http://psi.networkedplanet.
```

```
com/</tmshare:ServerSrcLocatorPrefix>
  <link rel="self"
        href="http://tmshare.networkedplanet.com/
              topicmaps/egov/fragments"/>
  <entry>
  <!-- Best practice: the topic display name or the PSI
should be used for the entry title -->
    <title>ISO 19115:2003 Geographic Information -
           Metadata</title>
    <!-- the published date and time of the fragment -->
    <updated>2008-07-17T15:55:21.971145Z</updated>
    <!-- the id value is some unique value -->
    <id>69CD5264-DB78-49c1-A7E4-04EECFA0AA85</id>
    <link rel="topicmapdata" type="text/xtm1+xml"
          href="http://tmshare.networkedplanet.com/topicmaps/
                egov/fragments/69CD5264-DB78-49c1-A7E4-
                04EECFA0AA85"/>

    <tmshare:TopicSI>http://psi.egovpt.org/standard/ISO+19115
   %3A+Geographic+Information+-+Metadata</tmshare:TopicSI>
  </entry>
  <!-- an entry follows for each fragment being exposed
                 ...
  -->
</feed>
```

Topic Map Fragment Data Service **-** a service that returns a specified topic map fragment.

Example Service URL:
`[server]/topicmaps/egov/fragments/fragment-for-topic-1`

Example Invocation:

```
Server: http://tmshare.networkedplanet.com
GET /topicmaps/egov/fragments/fragment-for-topic-1
```

Response structure:

A topic map fragment representation is a valid XTM 1.0 XML document. A fragment is created in the context of exactly ONE topic. The following algorithm should be applied when generating a fragment for given topic:

- Let 'export' mean to create an XTM representation of the TMDM construct
- Let T be the topic being exported.
- export T including ALL topicnames, identifiers, and occurrences.

- for each topicname in T export a topic stub for each name type (if it exists)
- for each topicname in T export a topic stub for each scope topic (if it exists)
- for each occurrence in T export a topic stub for the occurrence type (if it exists)
- for each occurrence in T export a topic stub for each scope topic (if it exists)
- for each association A in which T plays a role export the association
- for each association A export a topic stub for the association type
- for each association A export a topic stub for each topic scope topic
- for each role R in A export a topic stub for the role type and one for the role player UNLESS the role player is T

For each stub topic exported (the following minimum must be exported)

- export ALL of the topic's identifiers

ALL topics (stub or not) MUST have at least one Subject Identifier.

An server may choose to export more information in the fragment, what is described here is the minimum required.

Client Responsibilities

There are two aspects to client behaviour. The first is consumption of the feeds provided by the service the second is the updating of the local map based on the fragments it retrieves.

A Clean start

When a client first wants to sync with a server it can use the feeds provided to locate the topic map of interest, retrieve the full XTM topic map representation and merge it into the local topic map it is managing.

A Clean Replacement

If a client has a local topic map that contains topic map data from more than one server and wants to fetch and update the latest full topic map from ONE source then it MUST do the following. Apply the delete topic algorithm from below, but apply it to the entire topic map. Then proceed in terms of 'A Clean Start', by fetching the topic map and merging it in.

A partial update

Clients wishing to update their local topic map as new changes occur on the server, should process the changes feed for the appropriate topic map. The client MUST record the date and time that it last updated its local copy and then find all Atom entries that have an updated value after that time. For each of these, in time order of most distant to most recent it should apply the following update algorithm.

The Topic Map Fragment Update Algorithm

- Let SP be the value of the `ServerSourceLocatorPrefix` element in the Atom feed F
- Let SI be the value of TopicSI element in Atom entry E
- feed F contains E
- entry E references topic fragment TF
- Let LTM be the local topic map
- Let T be the topic in LTM that has a subjectidentifier that matches SI
- For all names, occurrences and associations in which T plays a role, TMC
    - Delete all `SrcLocators` of TMC that begin with SP
    - If the count of srclocators on TMC = 0 then delete TMC
- Merge in the fragment TF using SP as the base all generated source locators.

Note: To delete a topic an empty topic is published.

Note: The understanding is that each name, occurrence and association created or modified during the update will in its internal, TMDM-conformant representation have or get item identifiers that act as source locators and start with the `ServerSourceLocatorPrefix.`

References

1. XTM 1.0
   `http://www.topicmaps.org/xtm/index.html`
2. XTM 2.0
   `http://www.isotopicmaps.org/sam/sam-xtm/`
3. RFC 2616 HTTP 1.1
   `http://www.w3.org/Protocols/rfc2616/rfc2616.html`
4. XML 1.0
   `http://www.w3.org/TR/REC-xml/`
5. RFC 4287 Atom Syndication Format 1.0
   `http://www.atompub.org/rfc4287.html`