Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <yyyy-mm-dd>

External identifier of this OGC® document: http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}

Internal reference number of this OGC® document: YY-nnnrx

Category: OGC® White Paper

Editor: <Name(s) of Editor or Editors>

OGC (PubSub White Paper)

Copyright notice

Copyright © <year> Open Geospatial Consortium

To obtain additional rights of use, visit http://www.opengeospatial.org/legal/

Warning

This document is not an OGC Standard. This document is an OGC White Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, an OGC White Paper should not be referenced as required or mandatory technology in procurements.

Document type:	OGC® White Paper
Document subtype:	
Document stage:	Draft
Document language:	English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Table of Contents

i. Abstract

<Insert Abstract Text here>

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, <tags separated by commas>

iii. Preface

NOTE

Insert Preface Text here. Give OGC specific commentary: describe the technical content, reason for document, history of the document and precursors, and plans for future work. > Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

iv. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

Organization name(s)

v. Submitters

All questions regarding this submission should be directed to the editor or the submitters:

Name Affiliation

Chapter 1. References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- OGC 06-121r3, OGC® Web Services Common Specification, version 1.1.0 (9 February 2007)
- OGC 13-131, OGC® Publish/Subscribe Interface Standard 1.0 - Core
- OGC 13-133, OGC® Publish/Subscribe Interface Standard 1.0 - SOAP Protocol Binding Extension
- OGC 13-132, OGC® Publish/Subscribe Interface Standard 1.0 - RESTful Protocol Binding Extension (draft, in progress)
- OGC 07-006r1, OpenGIS® Catalogue Services Specification, version 2.0.2, corrigendum 2
- OGC 07-045, OpenGIS® Catalogue Services Specification 2.0.2 - ISO Metadata Application Profile, version 1.0
- OGC 07-110r4, CSW-ebRIM Registry Service - Part 1: ebRIM profile of CSW, version 1.0.1, corrigendum 1
- OGC 07-144r4, CSW-ebRIM Registry Service - Part 2: Basic extension package, version 1.0.1, corrigendum 1
- OGC 08-103r2, CSW-ebRIM Registry Service - Part 3: Abstract Test Suite, version 1.0.1
- OGC 16-137r2, Testbed-12 PubSub / Catalog Engineering Report (12 May 2017)

Chapter 2. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r3] shall apply. In addition, the following terms and definitions apply.

2.1. Broker | Brokering Publisher

Intermediary between Subscribers and other Publishers which have been previously registered with the broker. The broker is not the original producer of messages, but only acts as a message middleman, re-publishing messages received from other Publishers and decoupling them from their Subscribers

2.2. Message

A container within which data (such as XML, binary data, or other content) is transported. Messages may include additional information beyond data, including headers or other information used for routing or security purposes

2.3. Publication

A uniquely identified aggregation of messages published by a Publisher over time. A Publisher may offer any number of publications that Subscribers may subscribe to

2.4. Publisher

An entity that offers publications to Subscribers; supports subscription management (subscribe, unsubscribe) and is responsible for filtering and matching messages of interest to active subscriptions

2.5. Receiver

An entity that receives messages from Senders; may (but need not) be the original Subscriber

2.6. Reliable Publisher

A publisher of messages that offers capabilities to detect and recover from message delivery losses, whether caused by network failures, software failures, hardware failures, or other causes

2.7. Sender

Entity that sends messages to Receivers; may (but need not) be the initial creator/producer of the data in the message payload

2.8. Subscriber

Entity that creates a subscription at a Publisher; may (but need not) be the Receiver of delivered messages

2.9. Subscription

Expression of interest in all or part of a publication offered by a Publisher. When a subscription has been created, the Publisher delivers messages that match the subscription criteria to the Receiver defined in the subscription

Chapter 3. Conventions

3.1. Abbreviated terms

- CFP Call for Participation
- DWG Domain Working Group
- HTTP Hypertext Transfer Protocol
- MEP Message Exchange Pattern
- OGC Open Geospatial Consortium
- RFQ Request for Quotation
- SSE Server-Sent Events
- SWG Standards Working Group
- UML Unified Modeling Language
- XML eXtensible Markup Language

3.2. UML notation

Most diagrams that appear in this document are presented using the UML 2 static structure diagram, as described in Subclause 5.2 of [OGC 06-121r9].

All classes in this document are extensible and may be extended with application- or domain-specific content via Extension blocks.

NOTE	The UML shown in this document is considered conceptual and abstract, and should not be interpreted as an implementation strategy for bindings that extend and implement a standard. For example, TM_Instant from ISO 19108 may be used to represent time instants for conceptual clarity, but bindings and implementations of this document may realize TM_Instant as a GML TimeInstant, an ISO 8601 date string, or any other representation that is consistent with TM_Instant.

Chapter 4. Overview

Chapter 5. Asynchronous Operations

An asynchronous operation is any interaction between two software entities where the conclusing action does not immediately follow the initiating action. There are a number of different models for Asynchronous Operations. This section attempts to describe some of the more common ones. An understanding of the scope of this space will help discuss and compare the alternatives OGC faces in asynchronous services.

Asynchronous operations have three variables:

1. The overall pattern of behavior
2. How the two entities rendezvous
3. The criteria for completing the operation

5.1. Asynchronous Behavior Models

5.1.1. Request response

Request-Response is a synchronous behavior model. A request is issued by one entity and a response is provided in return. Asynchronous behaviors can best be understood in contrast to this model.

5.1.2. Request with delayed response

This model differs from Request-Response in that the immediate response is not the final response. Rather is indicates the success or failure of the request. The final response is to be delivered latter.

5.1.3. Publish Subscribe

5.1.4. Mesh Network (Pub Sub)

5.1.5. Synchronization (geosync)

5.1.6. Broadcast

5.1.7. Standing Query

5.2. Notification and Alert

5.2.1. Callback

5.2.2. Polling

5.2.3. Stored response queue

5.2.4. Instant Message

5.2.5. e-mail

5.2.6. Phone call

5.2.7. Snail mail

5.3. Filtering

5.3.1. Event (RSS, SNMP)

5.3.2. Tags (JMS)

5.3.3. Query expression (Standing Query)

5.3.4. Check Point

Chapter 6. OGC's PubSub implementation Standard

6.1. Overview

The OGC has conducted significant work on event-based models and architectures in the past, resulting in the Web Notification Service (WNS) Best Practice [1], the proposed Sensor Alert and Sensor Event Service, as well as several ERs produced in previous OGC interoperability initiatives. Starting from 2010, these efforts have been subsumed under the scope of the PubSub SWG and resulted in the adoption of the Publish/Subscribe Interface Standard 1.0 in February 2016.

Although the current OGC Baseline still supports only synchronous web service request-response capabilities, there is wide consensus that standard solutions for push-based message exchange patterns are fundamental enablers of the OGC vision, for example to implement the ubiquitous sensor-based applications in the advanced proactive control scenarios envisioned by the Internet of Things.

Several work items in the OGC Testbed 12 RFQ/CFP [2] have addressed the means to incorporate forms of asynchronous service interaction, including Publish/Subscribe message patterns, for example in WPS, WCS, WFS, or in geospatial queries of aviation data.

In particular, the RFQ/CFP included a specific Asynchronous Service Interaction subtask, part of a set of subtasks that aimed at enhancing the OGC Baseline, by extending OGC architectural designs through efforts that cross over several individual standards and services and are applied in a much wider scope.

The subtask description in the RFQ/CFP distinguished among three different approaches to handle asynchronous interaction with OGC Web services:

1. WPS façades;
2. Specific extensions to each OGC Web Service with asynchronous request/response capabilities;
3. OGC PubSub.

The document deliverable "A067 Implementing Asynchronous Service Response Engineering Report" (OGC 16-023) elaborates on the above approaches in situations where big chunks of data require asynchronous delivery. The ER focuses on the first and the second approach, with the goal to summarize and compare the results from using a WPS facade and an extension for WFS for asynchronous service responses, as well as to provide recommendations for future activities.

The document deliverable "A074 PubSub/Catalog Engineering Report" (OGC 16-137) focuses on the third approach, OGC PubSub, in the specific case of catalogs, investigating the functional requirements of an interoperable, push-based data discovery solution.

As underlined in the RFQ/CFP, it is important to provide methods that support notification (push) of new data as opposed to search (pull), given the volume of data that will be available in catalogs in the near future.

6.2. PubSub features

The recently approved OGC PubSub standard is intended as an overarching model to extend services with Publish/Subscribe capabilities. The Publish/Subscribe model is distinguished from the request/reply and client/server models by the asynchronous delivery of messages and the ability for a Subscriber to specify an ongoing (persistent) expression of interest.

The PubSub specification is agnostic as to what constitutes a change, i.e. an event that should cause a notification by a Publisher (aka its event model). It is only required that a Publisher instance communicate what notifications it will emit by advertising them in the Publication section of its Capabilities document.

Likewise, the PubSub standard is agnostic as regards delivery methods. It defines a Publisher role that may support multiple delivery methods, such as ATOM, AMQP, or SOAP, as advertised in its Capabilities document, in the DeliveryCapabilities section. By implementing the Publisher interface, a PubSub-OWS may offer more than one method of delivery for each Publication, to be chosen by Subscribers. Publish/Subscribe would imply push-style message delivery, however some methods may actually be pull-based (e.g. polling), under the hood.

Hence, both the solutions investigated in ER OGC 16-023, the WPS façade and the specific WFS extension for asynchronous responses, are compatible with the PubSub standard and may be integrated in an application.

The application of OGC PubSub to OGC Catalogue Services allows an analyst to register and be notified when new metadata records become available. Such new OGC-compliant PubSub-enabled discovery service supports and promotes the system architecture of the Testbed 12 Initiative, as a fundamental building block to advance the current OGC standard baseline. As a consequence, such work may contribute to the enhanced availability of standards-based offerings in the marketplace.

6.3. Basic PubSub 1.0 extension for the generic OWS

This chapter summarizes the PubSub extension for the generic OWS introduced by OGC 16-137. This extension is conceived as a simple way to enable the existing request/reply OWS specifications to Publish/Subscribe, by implementing the OGC Publish/Subscribe Interface Standard 1.0.

An OWS implementing this extension is capable of accepting its usual requests as filters, and of sending notifications about data/metadata updates, based on its existing semantics and syntax expressiveness.

6.4. Conceptual model

This chapter describes how PubSub 1.0 Core operations, encodings and messages are modeled in terms of the functionalities of the generic OWS. No assumption is made on the capabilities of the target OWS, other than those defined by the OGC Web Services Common Standard. Hence this extension may apply, for example, to WFS, WCS, and other OWS interfaces.

The PubSub specification is agnostic as to what constitutes a change, i.e. an event that should cause a notification by a Publisher (aka its event model). It is only required that a Publisher instance

communicate what notifications it will emit by advertising them in the Publication section of its Capabilities document (see below).

In general, a PubSub-OWS may be able to notify about changes to any component of its information set. For example, it may notify about changes to its Capabilities document. The extension introduced in this chapter addresses the most general case, at the expenses of efficiency and semantic accuracy. The precise definition of an event model for the various OWS's is left to the relevant OGC Working Groups.

The basic PubSub-OWS MEP can be generalized as follows (see figure Figure 1):

1. The OWS client subscribes specifying a request to be used as filter for the notifications;
2. The OWS client obtains the Time-0 response via a standard Request/Reply, with the same request as above;
3. The OWS notifies the client of subsequent updates to the response, according to its existing semantics and syntax.

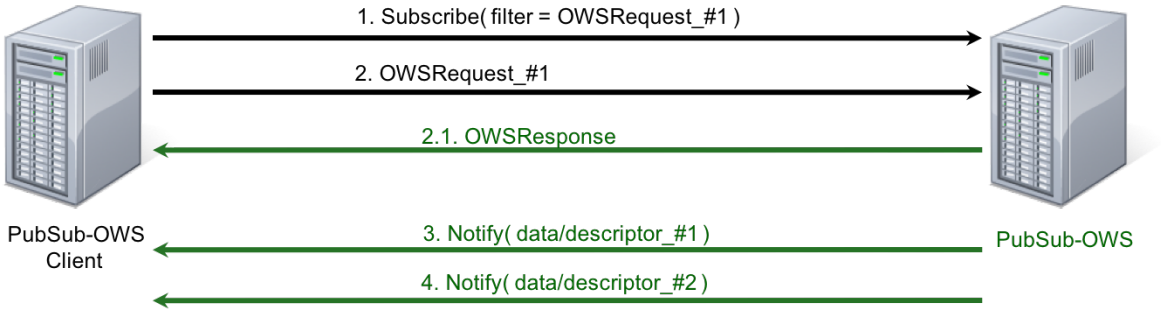

Figure 1. OWS Publish/Subscribe MEP

This may be formalized in an "OWS Request/Reply Publisher" Conformance Class that:

- Accepts OWS requests as subscription filters
  - The Publisher may constraint the filter expressions allowed in Subscriptions (e.g. by imposing OpenSearch templates)
- Sends corresponding OWS responses to notify about data/metadata updates

This MEP is a simple way to enable existing OWSs to PubSub, allowing to bind the PubSub 1.0 Core operations, encodings and messages to the standard OWS functionalities, data models, and semantics.

6.5. Required Capabilities components

PubSub Core requires that the OWS advertise the implemented Conformance Classes in its Capabilities document, namely in the Profile property of the ServiceIdentification section (as of OWS Common 1.1). Besides, it requires that the additional Capabilities components represented in figure Figure 2 are returned in the GetCapabilities response, but does not specify the specific mechanism for incorporating these additional Capabilities components into the OWS Capabilities document. These extension proposes to include these additional Capabilities components in the ExtendedCapabilities of the OWS, as detailed in the following chapters.

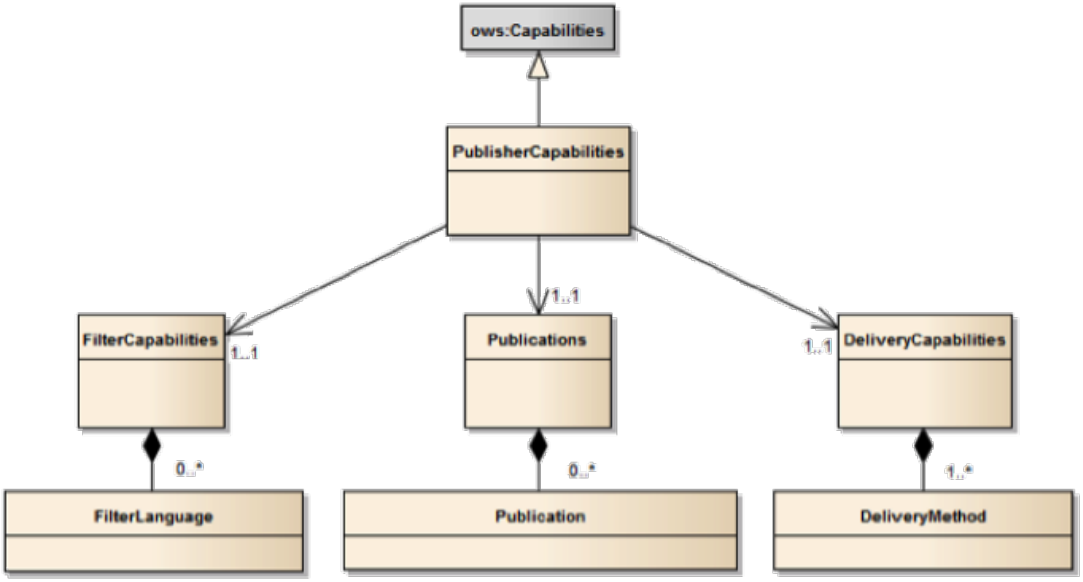

Figure 2. PubSub Capabilities components

6.5.1. FilterCapabilities

The FilterCapabilities section describes the filtering-related capabilities of a PubSub-OWS, i.e. the filter languages it supports for matching events against subscriptions (e.g., OGC Filter Encoding). This allows the pluggability of filter languages.

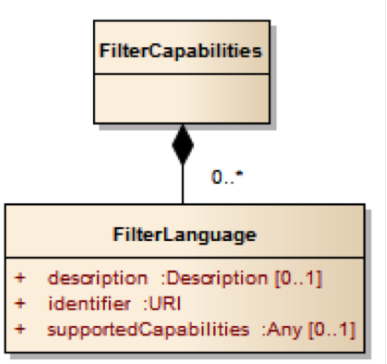

Figure 3. Filter Capabilities

The SupportedCapabilities elements allows restricting the acceptable requests, possibly providing templates. The following Capabilities snippet declares that this PubSub-OWS instance (namely, a CSW) accepts as subscription filters GetRecords requests conforming to the specified OpenSearch template. Multiple templates may be introduced, specifying multiple FilterLanguages.

FilterCapabilities

```
<FilterCapabilities>
   <FilterLanguage>
      <Abstract>This PubSub-OWS accepts requests as subscription filters, according to
the OpenSearch template specified in SupportedCapabilities.
      </Abstract>
      <Identifier>http://www.opengis.net/spec/pubsub/1.0/conf/ows/request-reply-
publisher</Identifier>
      <SupportedCapabilities>http://tb12.essi-lab.eu/pubsub-
csw/services/opensearch?ct={count?}&amp;st={searchTerms?}&amp;bbox={geo:box?}&amp;ts={
time:start?}&amp;te={time:end?}
      </SupportedCapabilities>
   </FilterLanguage>
</FilterCapabilities>
```

6.5.2. DeliveryCapabilities

The DeliveryCapabilities section describes the delivery methods supported by the PubSub-OWS, e.g. SOAP, WS-Notification, ATOM, SSE, WebSockets, OAI-PMH. This allows the pluggability of delivery methods.

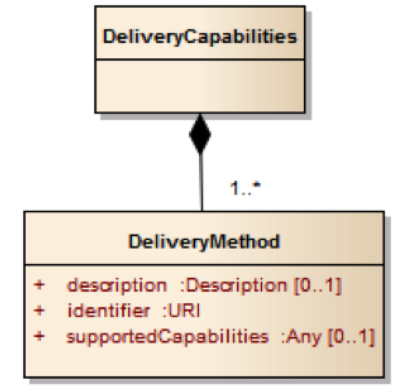

Figure 4. Delivery Capabilities

The following Capabilities snippet declares that this PubSub-OWS instance delivers notifications via SSE (see chapter Delivery methods, below).

DeliveryCapabilities

```
<DeliveryCapabilities>
   <DeliveryMethod>
      <Abstract>This PubSub-OWS supports notification delivery via SSE.
      </Abstract>
      <Identifier>http://www.w3.org/TR/eventsource/
      </Identifier>
   </DeliveryMethod>
</DeliveryCapabilities>
```

Delivery methods

The DeliveryCapabilities section describes the methods supported by the PubSub-OWS for delivering notifications. Publishers may offer more than one method of delivery for each Publication, to be chosen by Subscribers. Publish/Subscribe would imply push-style message delivery, however some methods may actually be pull-based (e.g. polling), under the hood.

Examples include: SOAP and related technologies, such as WS-Notification (used by PSSB), ATOM (polling using the “If-Modified-Since” and “start-index” parameters), PubSubHubbub, OAI-PMH (polling using the “from” parameter), e-mail, SMS, WebSockets, SSE.

Server-Sent Events (SSE) is a pure push-style communication technology based on HTTP and the SSE EventSource API standardized as part of HTML5 by the W3C. A SSE client (e.g. all modern HTML 5.0 browsers) receives automatic updates from a server via HTTP connection, simply setting the following parameters:

- ContentType: "text/event-stream;charset=UTF-8”
- Cache-Control: "no-cache”
- Connection: "keep-alive”

6.5.3. Publications

The Publications section describes the contents offered by the PubSub-OWS, i.e. the sequences of notifications that Subscribers can subscribe to.

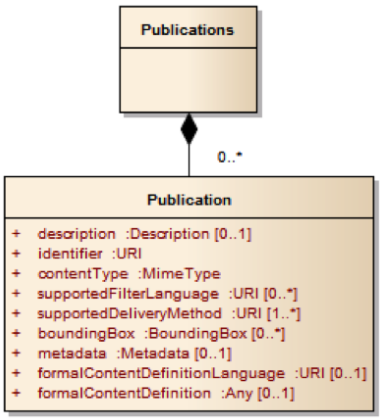

Figure 5. Publications

The following Capabilities snippet declares a publication that notifies on all the relevant events for this PubSub-OWS. Notifications can be filtered with the semantics of the requests of this OWS and are delivered via SSE, encoded in JSON (see chapter Notification encoding, below).

Publications

```
<Publications>
   <Publication>
      <Abstract>>This publication notifies on all the relevant events for this PubSub-
OWS.
      </Abstract>
      <Identifier>ALL</Identifier>
      <ContentType>application/json</ContentType>

<SupportedFilterLanguage>http://www.opengis.net/spec/pubsub/1.0/conf/ows/request-
reply-publisher</SupportedFilterLanguage>

<SupportedDeliveryMethod>http://www.w3.org/TR/eventsource/</SupportedDeliveryMethod>
   </Publication>
</Publications>
```

Notification encoding

For the generic OWS instance, no operation is defined that provides the basic semantics of “insert”, “update”, and “delete” actions on the content managed by the instance.

The most generic mechanism to notify about updates is that the Publisher re-send the whole response element corresponding to the request used as filter in the Subscription. For example, in the case of WFS, if the client subscribes with a wfs:GetFeature request as a filter, the PubSub-WFS should notify about any changes by delivering a standard wfs:FeatureCollection, in response to that request.

By receiving the new response and comparing it with the previous one, a Subscriber can figure out the changes. Future evolutions of this extension may evaluate more efficient and semantically accurate encoding of notifications. A possible option for XML-based content types is XMLdiff (e.g. XML Patch, RFC 5261), or annotations (XML attributes) to add simple CRUD semantics on top of the existing XSDs.

6.6. Support to legacy components

The integration of legacy components in an eventing architecture is desirable in a number of scenarios. However, legacy components may not be instrumented to monitor their state for the purpose of notification, nor to react upon notifications from other components (or they may, but by legacy, non-standard mechanisms).

Implementing the PubSub 1.0 Standard in a legacy component may not be feasible or practical. In some cases, the legacy component can be adapted to the Publish/Subscribe MEP by an additional functional entity that realize the Publish/Subscribe functionalities. Such mediating entity acts as a proxy/adapter, i.e. a middleman between the source and the target of the message exchange, implementing the behavior and/or the interfaces required by the PubSub specification.

This use case has been considered in the phase of requirement analysis for the PubSub 1.0 standard [3] and is supported by the Brokering Publisher Conformance Class of the PubSub 1.0 Standard.

Depending on the intended role of the legacy component, the use case is twofold:

- Proxied Subscribe – a proxy/adapter component subscribes to a Publisher on behalf of the legacy system and acts appropriately upon receiving notifications of interest.

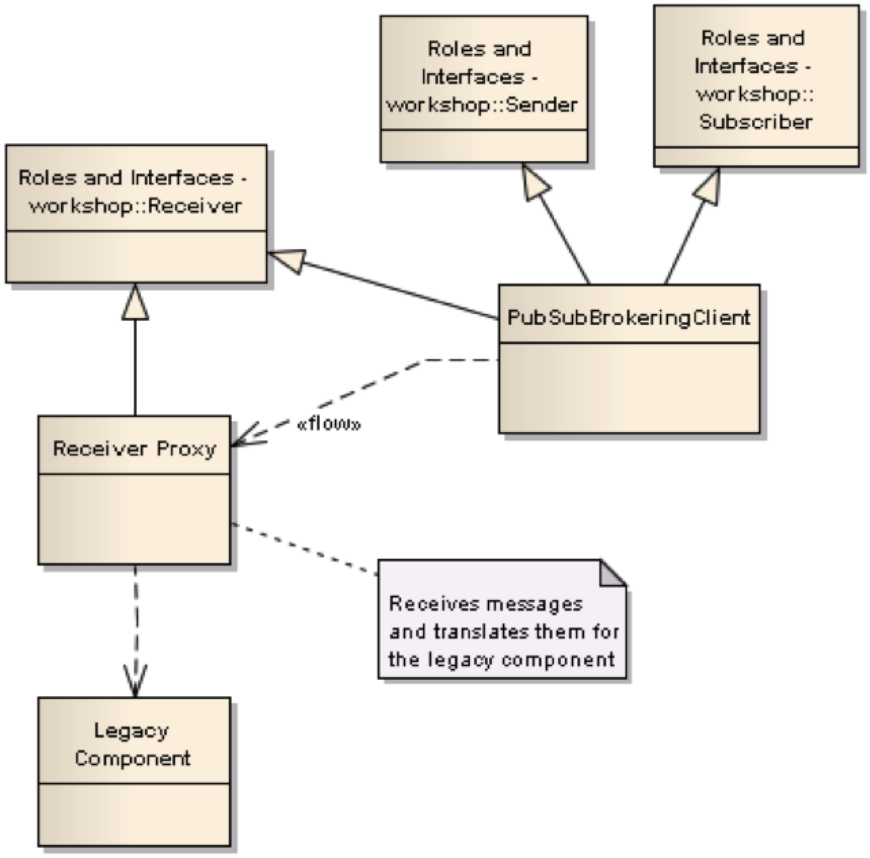

Figure 6. Proxied subscribe

- Proxied Publish – a proxy/adapter component monitors the legacy system and generates appropriate notifications upon relevant events (according to a given event model). The proxy/adapter may act as a full-fledged Publisher, accepting Subscriptions against the sequence of notifications, or just act as a pure Sender, relaying each notification to another Publisher entity (see figure Figure 7).

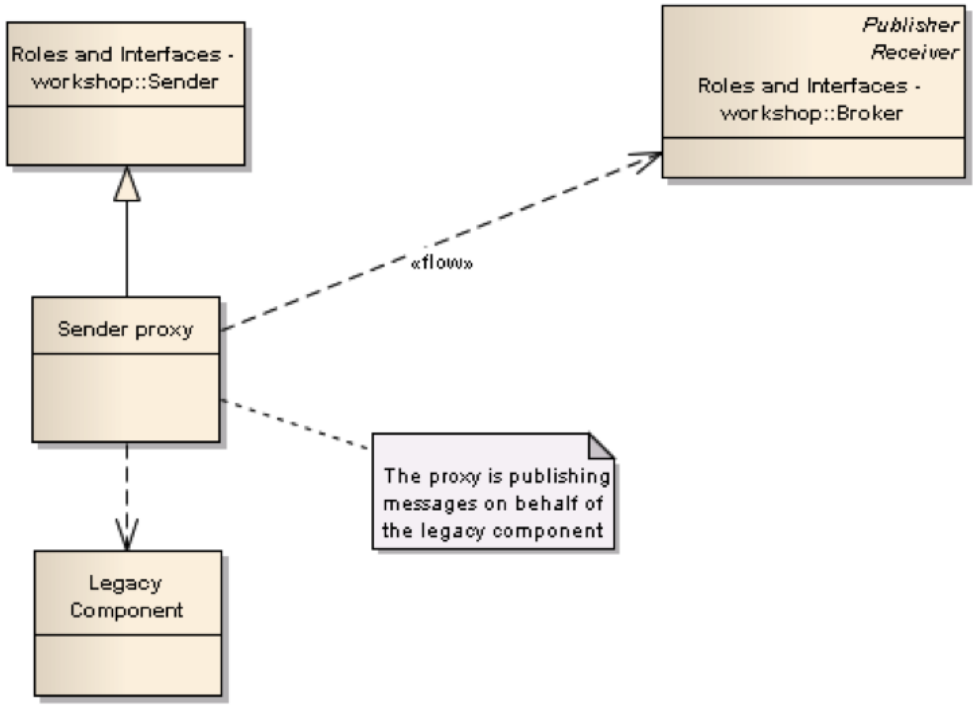

Figure 7. Proxied publish

The Brokering Publisher Conformance Class of the PubSub 1.0 Standard supports this use case. In fact, a Brokering Publisher (or, more simply, a broker), is an intermediary between Subscribers and other Publishers which have been previously registered with the broker. The broker is not the original producer of messages, but only acts as a message middleman, re-publishing messages received from other Publishers and decoupling them from their Subscribers. A broker may shuffle or aggregate messages into different publications, may offer publications with different delivery methods than the original ones, or otherwise process the messages (e.g. converting their format). A broker may also provide advanced messaging features, such as load balancing.

In general, a broker is a distinct third party that acts as a communication intermediary between the source and the target of a communication, mediating their interfaces and in some cases adding new behavior. Hence, a broker may conveniently act as a proxy/adapter for one or more legacy components, flexibly implementing any combination of the above twofold use case.

The Brokering Publisher Conformance Class does not mandate any specific behavior to be implemented, in particular as regards the support to Delivery Capabilities, Filtering Capabilities, and Publications of the brokered Publishers. Brokers are free to interact with the brokered Publishers as appropriate for their specific application. Interactions may include subscribing to the offered publications, harvesting the data, decorating the capabilities, or other behavior (future extensions of the Conformance Class may standardize the behavior of Brokering Publishers in specific application scenarios).

Examples of Brokering Publisher applications include the following:

- Publisher Aggregation – a broker subscribes to several Publishers and relays their publications (without modifications) to interested Subscribers, acting like a Proxy to multiple Publishers. Optionally, the broker may adapt the service interface (binding) of the aggregated Publishers.
- Publication Aggregation – a broker receives messages generated by several Publishers (e.g. dumb sensors) and publishes them to the interested Subscribers as a single publication at a

single endpoint, for the sake of simpler connectivity, or improved accountability, or easier management of subscriptions, etc.

- GeoSynchronization (GSS) - GSS is a mediation service that controls transactional access to one or more WFS's (e.g. to moderate updates in crowdsourcing scenarios). A GSS maintains several event channels, including one for changes applied to the WFS content. Clients can subscribe to the channels (possibly specifying a filter) and be notified by the GSS whenever new entries appear. A GSS may be used to monitor insert/update/delete operations performed on one or more WFS's and send appropriate notifications, implementing the PubSub 1.0 Brokering Publisher Conformance Class. Whenever an event (i.e. a Transaction) occurs on a WFS, the GSS will notify Subscribers of that event. In this way WFS's that do not implement the PubSub 1.0 Standard can participate in an eventing architecture. There are plans to extend GSS to other OGC access services, such as WCS.

[1] http://portal.opengeospatial.org/files/?artifact_id=18776

[2] http://www.opengeospatial.org/standards/requests/139

[3] See also the Proxied Publish/Subscribe use case (access restricted to OGC Members): https://portal.opengeospatial.org/wiki/PUBSUBswg/PubSubSwgUseCaseBrokeredPubSub

Chapter 7. PubSub in OGC SensorThings

OGC SensorThings API is a standard for interconnecting the Internet of Things (IoT) devices, data, and applications over the Web. OGC SensorThings API has two main parts, Sensing part which is focusing on heterogeneous IoT sensor systems, and the Tasking part that is focused on IoT actuators and tasking devices.

OGC SensorThings API is following ODATA for managing the sensing resources. Thus, it has a REST-like API and supports HTTP CRUD operations (GET, POST, PATCH, DELETE) as well as ODATA query options (select, expand, filter, orderby, top, skip) for data retrieval.

In addition to supporting HTTP, OGC SensorThings API has the extension for supporting MQTT for creation and real-time retrieval of sensor Observations.

MQ Telemetry Transport, known as MQTT, is an OASIS standard. It is an extremely lightweight publish/subscribe messaging protocol which is specifically designed for resource-constrained IoT devices. The main concepts are topic, and publish and subscribe functions. When a new message is published to a topic, anyone subscribing to that topic will receive a notification including that message.

OGC SensorThings API is adopting MQTT protocol and defines a topic structure for creating and also receiving notifications of sensor Observations. The topic structure for OGC SensorThings API MQTT extension is following its HTTP resource path. It means that the topic used for creating Observation through MQTT is identical to the HTTP URL that is used for POST and creating Observations. Examples of these resource paths are ***v1.0/Observations*** and ***v1.0/Datastreams(id)/Observations***. Similarly, accessing those URLs using HTTP GET would result in retrieving sensor Observations, while subscribing to those as topics means receiving notifications for those Observations in real-time.

Here is the summary of lessons learnt from MQTT adoption is OGC SensorThings API that might be applicable to any RESTful API:

- Each RESTful API has a potential for MQTT binding to receive updates for a resource collection
  - The topic would be the resource GET URL
  - The payload would be the same as content of HTTP GET
  - Whenever there is a new resource, it will be published to the resource GET URL topic
- For any RESTful API, to create a resource, MQTT can be an option just like HTTP POST
  - The topic will be same as the POST topic
  - The payload will be the same as POST payload
  - The service would subscribe to the topics
  - When it receives the payload, it uses the same process as POST for creating the resource

We also did a pilot on a simple rules engine for Institute for the Management of Information Systems (IMIS) pilot project. In that pilot, what we did was to define topics that contain the resource path together with $filter query options. Then the Observation that are published to that topic would be filtered by the criteria defined in the topic. This way we can define rules that notify

subscribers about certain condition that happened, for example when the CO reading are higher than a threshold and considered dangerous. Example of the topic would be ***v1.0/Datastreams(id)/Observations?$filter=result gt 100***. This was a pilot implementation and could be a starting point for OGC SensorThings API potential part 3, rules engine.

Chapter 8. W3C Pub Sub Recommendation

Description of the standard and recommendationsfor OGC PubSub.

Presentation from Stuttgart would be good to draw from.

Chapter 9. Asynchronous Messaging for Aviation from OGC testing

Asynchronous Messaging for Aviation from OGC testing

Chapter 10. WMO OpenWIS use of AMQP

Description of the standard and recommendations for OGC PubSub

Chapter 11. AsyncAPI description

Description of the AsyncAPI spec in general.

The role of AsyncAPI is to provide a language for designing the API.

Chapter 12. Ideas for AsyncAPI in OGC

Application of Async in the context of OGC API work

Chapter 13. Ideas for an OGC Abstract Spec for PubSub

Concepts to be included in an OGC AS for PubSub - draw from the previous sections

Annex A: Title

NOTE	Place other Annex material in sequential annexes beginning with "A" and leave final two annexes for the Revision History and Bibliography <<<

Annex B: Revision History

Date	Release	Editor	Primary clauses modified	Description
2016-04-28	0.1	G. Editor	all	initial version

Annex C: Bibliography

NOTE

Example Bibliography (Delete this note).

The TC has approved Springer LNCS as the official document citation type.

Springer LNCS is widely used in technical and computer science journals and other publications

- For citations in the text please use square brackets and consecutive numbers: [1], [2], [3]

– Actual References:

[n] Journal: Author Surname, A.: Title. Publication Title. Volume number, Issue number, Pages Used (Year Published)

[n] Web: Author Surname, A.: Title, http://Website-Url

[1] OGC: OGC Testbed 12 Annex B: Architecture. (2015). [2] L. Bigagli, M. Rieke. The new OGC Publish/Subscribe Standard - applications in the Sensor Web and the Aviation domain. Open Geospatial Data, Software and Standards, ISSN: 2363-7501, doi: 10.1186/s40965- 017-0030-7, vol. 2, issue no. 1, article 18, Springer, Heidelberg, Germany, December 2017 (electronic 3 August 2017). [3] M. Rieke, L. Bigagli, S. Herle, S. Jirka, A. Kotsev, T. Liebig, C. Malewski, T. Paschke, C. Stasch. Geospatial IoT—The Need for Event-Driven Architectures in Contemporary Spatial Data Infrastructures. ISPRS International Journal of Geo-Information (IJGI), ISSN 2220-9964, Volume 7, Issue 10, article 385, doi: 10.3390/ijgi7100385, MDPI, Basel, Switzerland, 25 September 2018.