OGC Testbed-13

Asynchronous Services ER

Table of Contents

1. Summary . . . 4
   1.1. Requirements . . . 4
   1.2. Key Findings and Prior-After Comparison . . . 5
   1.3. What does this ER mean for the Working Group and OGC in general . . . 5
   1.4. Document contributor contact points . . . 5
   1.5. Future Work . . . 6
   1.6. Foreword . . . 6
2. References . . . 7
3. Terms and definitions . . . 8
   3.1. Asynchronous Messaging . . . 8
   3.2. Message . . . 8
4. Abbreviated terms . . . 9
5. Overview . . . 10
6. Previous Work . . . 11
   6.1. Additional Request Parameters/Polling . . . 11
   6.2. Publish/Subscribe . . . 13
   6.3. MQTT Extension . . . 14
7. NG119 Asynchronous WFS-1 . . . 16
   7.1. Demonstration Scenario . . . 16
   7.2. Implementation . . . 19
   7.3. NSG TDS OSM as JSON . . . 24
8. NG120 Asynchronous WFS-2 . . . 27
   8.1. Demonstration Scenario . . . 27
   8.2. Implementation . . . 29
      8.2.1. Data source . . . 30
      8.2.2. GetCapabilities operation . . . 31
      8.2.3. GetFeature operation . . . 32
      8.2.4. GetStatus operation . . . 33
      8.2.5. Cancel operation . . . 33
      8.2.6. View the result . . . 33
      8.2.7. Scenario UI . . . 34
9. Recommendations & Future Work . . . 35
   9.1. Exploring the role of Asynchronous Web Feature Services in Geospatial Enterprise Architectures 35
   9.2. Asynchronous Messaging for other OGC Service Types . . . 35
   9.3. PubSub Extension of Web Feature Service . . . 35
   9.4. Additional Response Formats for WFS responses . . . 36
Appendix A: Revision History . . . 37

Appendix B: Bibliography 39

Publication Date: 2018-01-08

Approval Date: 2017-12-07

Posted Date: 2017-10-27

Reference number of this document: OGC 17-028

Reference URL for this document: http://www.opengis.net/doc/PER/t13-NG007

Category: Public Engineering Report

Editor: Benjamin Pross, Christoph Stasch

Title: OGC Testbed-13: Asynchronous Services ER

OGC Engineering Report

COPYRIGHT

WARNING

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is 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, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

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.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Summary

While most current OGC specifications are based on synchronous communication patterns, there are several applications that need asynchronous client-server interaction patterns. This way, no immediate response is required to continue processing on the client side. Clients can pick up messages directly or at a later point in time.

The OGC Publish/Subscribe (PubSub) standard (OGC 13-131r1) released in 2016 is now providing a standardized interface for adding asynchronous messaging to OGC web services following the publish/subscribe pattern, i.e. clients can subscribe to certain information and then receive the information once it has been published by an information provider. The OGC Web Processing Service (WPS) standard (OGC 14-065) also defines asynchronous communication, but instead of subscribing to processing results, the status information needs to be queried by the client ("polling"). Another way of providing asynchronous messaging is defined in the OGC SensorThings API (OGC 15-078r6), where utilizing Message Queue Telemetry Transport (MQTT) is proposed for this purpose.

In recent testbeds, there were several activities investigating how to enable asynchronous communication for OGC web services. The latest results of these activities are three engineering reports written in Testbed 12:

- OGC Testbed 12 Asynchronous Services Response Engineering Report (OGC 16-023r3)
- OGC Testbed 12 Asynchronous Messaging for Aviation (OGC 16-017), and
- OGC Testbed 12 PubSub / Catalog ER (OGC 16-137).

OGC 16-023r3 describes two approaches for adding asynchronous functionality to the OGC Web Feature Service (WFS) standard, i.e. using a WPS as facade and the polling mechanism of WPS or adding additional query parameters in the WFS requests. OGC 16-017 describes an approach for asynchronous retrieval of aviation data (i.e. Aeronautical Information Exchange Model (AIXM) and Flight Information Exchange Model (FIXM)) information using geospatial queries and the Advanced Message Queuing Protocol (AMQP) 1.0 as the underlying messaging protocol. OGC 16-137 provides information on a PubSub binding for the OGC Catalogue Service (CS-W) standard and demonstrated how a Representational State Transfer (REST) binding for PubSub can be defined.

The goal of this ER is to summarize and compare the results from the activities dealing with asynchronous WFS responses in Testbed 13. Special focus will be given to the specific requirement for automatic notification of users if new or updated information becomes available and to the software components addressing these requirements, i.e. two asynchronous Web Feature Services (NG119 and NG120).

Parts of this work have been funded by the COLABIS and Mudak-WRM projects funded by the German Federal Ministry of Education and Research under grant agreement numbers 03G0852C and 02WGR1431C.

1.1. Requirements

Compared to previous testbeds, new classes of use cases are addressed in Testbed 13:

- Communication in situations with denied, degraded, intermittent, or limited bandwidth: Users have an unreliable connection to the network and need to synchronize their local data with servers as soon as a connection become available.
- Automatic notification of users if new or updated data sets become available that fulfill their query criteria. In this case, specific focus needs to be given to the way how to define such query criteria.

1.2. Key Findings and Prior-After Comparison

Extensions for adding asynchronous communication to WFSs are not yet available. The approaches for Asynchronous Web Feature Services implemented in this Testbed implement and extend an approach developed in the previous Testbed 12 (see Section 7, OGC 16-023r3) using additional query parameters indicating that the operation should be executed in an asynchronous communication mode. The approaches described in this engineering report extend the previous work as follows:

- **Profile support**: The Asynchronous WFS-1 implements the NSG WFS 2.0 Profile. The Asynchronous WFS-2 implements the Defense Geospatial Information Working Group (DGIWG) WFS 2.0 profile.
- **Management of asynchronous operations**: The Asynchronous WFS-2 implements additional operations for retrieving the status (GetStatus) of or canceling (Cancel) operations running in asynchronous communication mode. These are aligned with the OGC WPS specification.
- **Novel use cases**: The scenarios where the two Asynchronous WFSs have been tested in two scenarios with denied, degraded, intermittent, or limited bandwidth: one scenario where an Asynchronous WFS supports simulated users in a Mass Migration Scenario over Syria and Jordan and another scenario where a large OpenStreetMap (OSM) dataset needs to be downloaded from a WFS.

1.3. What does this ER mean for the Working Group and OGC in general

While most current OGC standards are based on synchronous communication patterns, there are several applications, which need asynchronous client-server interaction patterns. This way clients do not have to keep the connection to the server continuously open in order to wait for responses. This ER summarizes the results from the different activities in Testbed 13 dealing with asynchronous service responses and discusses and relates them to previous related activities. Special focus was given to the specific requirements for asynchronous communication in the two planned use case types, i.e. communication in situations with denied, degraded, intermittent or limited bandwidth and automatic notification of users if new or updated data sets become available. Solutions have been provided for the OGC Web Feature Service that reuse and extend the concepts developed in Testbed 12. The ER documents the relevant decisions, implementations, and findings in order to help to enable asynchronous communication in OGC based infrastructures.

1.4. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization
Benjamin Pross (Editor)	52°North GmbH
Christoph Stasch (Editor)	52°North GmbH
Jeff Harrison	Carbon Project
Chen-Yu (How) Hao	Feng Chia University
Chih-Wei (Will) Khuan	Feng Chia University

1.5. Future Work

The following issues are considered as future work:

- Exploring the role of Asynchronous Web Feature Services in Geospatial Enterprise Architectures
- Asynchronous Messaging for other OGC Service Types
- PubSub Extension of Web Feature Service
- Additional Response Formats for WFS responses

A more detailed description of these issues is given in Section 9.

1.6. Foreword

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.

Chapter 2. References

The following normative documents are referenced in this document.

- OGC 06-121r9, OGC® Web Services Common Standard [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- OGC 14-065, OGC® WPS 2.0 Interface Standard Corrigendum 1 [http://docs.opengeospatial.org/is/14-065/14-065.html]
- OGC 13-131r1, OGC® Publish/Subscribe Interface Standard 1.0 - Core [http://docs.opengeospatial.org/is/13-131r1/13-131r1.html]
- OGC 13-133r1, OGC® Publish/Subscribe Interface Standard 1.0 - SOAP Protocol Binding Extension [http://docs.opengeospatial.org/is/13-133r1/13-133r1.html]
- OGC 07-006r1, OpenGIS Catalogue Service Implementation Specification [http://portal.opengeospatial.org/files/?artifact_id=20555]
- OGC 09-025r2, OGC® Web Feature Service 2.0 Interface Standard - With Corrigendum [http://docs.opengeospatial.org/is/09-025r2/09-025r2.html]
- OGC 15-078r6, OGC SensorThings API Part 1: Sensing [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html]
- OGC 16-023r3, OGC Testbed 12 Asynchronous Services Response Engineering Report [http://docs.opengeospatial.org/per/16-023r3.html]
- OGC 16-017, OGC Testbed 12 Asynchronous Messaging for Aviation [http://docs.opengeospatial.org/per/16-017.html]
- OGC 16-137, OGC Testbed 12 PubSub / Catalog ER [http://docs.opengeospatial.org/per/16-137r2.html]
- OGC 17-037, OGC Testbed-13: SWAP Engineering Report [https://portal.opengeospatial.org/wiki/pub/Testbed13/ConvertDocsT13Output/T13/NG005-SWAP.html]
- OGC 17-031, OGC Geo-Synchronization Standard [https://portal.opengeospatial.org/wiki/pub/Testbed13/ConvertDocsT13Output/T13/NG011-GeoSynchronization.html]

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

3.1. Asynchronous Messaging

```
Describes a communication pattern in which sending entities can deliver in an
asynchronous way. No immediate response from the receiving entity is required to
continue processing. Receiving entities can pick up messages directly or at a later
point in time (as defined previously in OGC 16-017).
```

3.2. 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.(source: OGC 13-
131r1)
```

Chapter 4. Abbreviated terms

- API Application Program Interface
- Async WFS Asynchronous Web Feature Service
- AIXM Aeronautical Information Exchange Model
- CS-W Catalogue Service
- DGIWG Defense Geospatial Information Working Group
- ER Engineering Report
- FIXM Flight Information Exchange Model
- GML Geography Markup Language
- JSON JavaScript Object Notation
- NSG National System for Geospatial Intelligence
- MQTT Message Queue Telemetry Transport
- OGC Open Geospatial Consortium
- OSM OpenStreetmap
- PubSub Publish/Subscribe
- REST Representational State Transfer
- SOS Sensor Observation Service
- TDS Topographic Data Store
- UI User Interface
- WCS Web Coverage Service
- WFS Web Feature Service
- WPS Web Processing Service
- WS-N Web Service Notification

Chapter 5. Overview

While most current OGC specifications are based on synchronous communication patterns, there are several applications, which need asynchronous client-server interaction patterns. This way, clients do not have to keep the connection to the server continuously open in order to wait for responses, but can rather subscribe to certain information and then receive the information once it has been published by an information provider (Publish/Subscribe pattern).

The two major use cases that require such an asynchronous communication and that have been addressed in Testbed 13 are:

- Communication in situations with denied, degraded, intermittent, or limited bandwidth: Users have an unreliable connection to the network and need to synchronize their local data with servers as soon as a connection becomes available.
- Automatic notification of users if new or updated data sets become available that fulfill their query criteria. In this case, specific focus needs to be given to the way how to define such query criteria.

The goal of this ER is to summarize and compare the results from the different activities dealing with asynchronous service responses in Testbed 13. An overview on these activities is given in Figure 1.

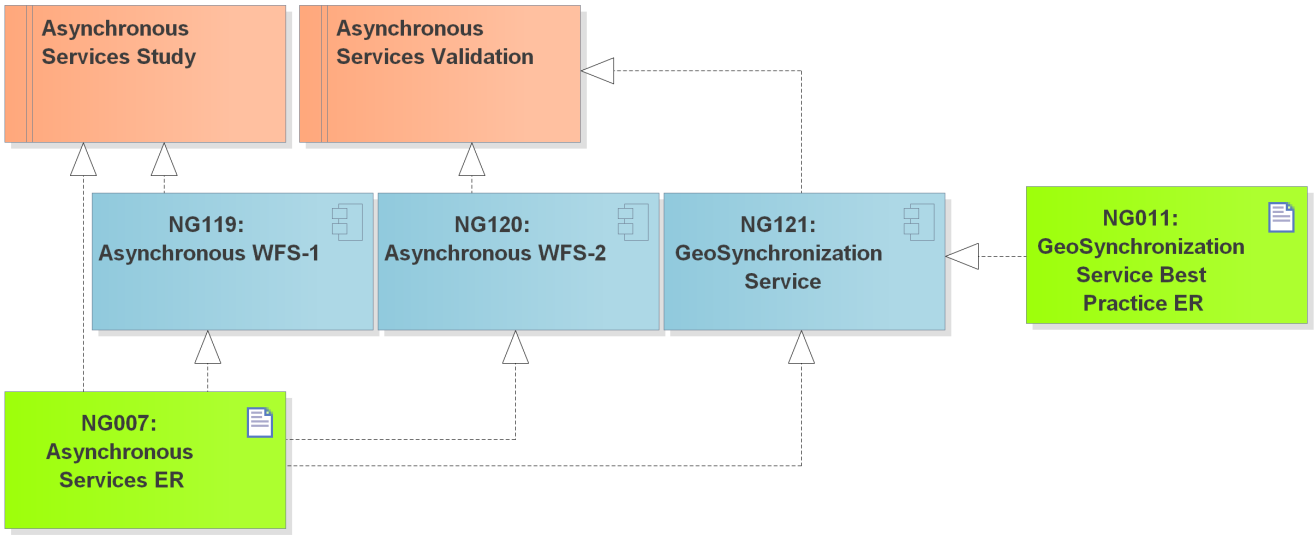

Figure 1. Overview on activities in the Asynchronous Services thread in Testbed 13.

Special focus is given to the specific requirement for automatic notification of users, if new or updated information becomes available, and to the software components addressing these requirements, i.e. the two asynchronous Web Feature Services (NG119 and NG120). The work regarding the GeoSynchronization Service (NG121) is described in a separate standard document developed in Testbed 13, the OGC Geo-Synchronization Standard (OGC 17-031) [https://portal.opengeospatial.org/wiki/pub/Testbed13/ConvertDocsT13Output/T13/NG011-GeoSynchronization.html].

The remainder of this ER is as follows: Section 6 provides an overview on previous activities and related technologies. Sections 7 and 8 describe the scenarios and the actual implementations of the two Asynchronous WFSs including results of integration testing. Finally, Section 9 describes recommendations and future work items derived from the work described in this ER.

Chapter 6. Previous Work

Several discussion papers and best practices have been developed previously for supporting asynchronous communication for OGC services. In the Sensor Web community, the Sensor Alert Service and Sensor Event Service were specified to enable publish/subscribe patterns for sensor data sources and data consumers. For delivering the information to subscribed clients, the Web Notification Service has been defined as a complementary service. These concepts have been used in previous Testbeds 6 to 10 to implement asynchronous communication. However, the Sensor Web approaches did not reach the level of OGC implementations standards and there was no general approach for asynchronous communication with OGC services.

Thus, the OGC initiated the PubSub working group in 2010 that ended in 2016 with the release of the OGC Publish/Subscribe Interface Standard 1.0. In addition, the OGC Web Processing Service Implementation Standard also defines support for asynchronous communication since the first version.

Currently, three approaches exist for extending OGC Web Services by asynchronous communication. These are described in the following two subsections. The work described in this ER consists of extensions of the previous approaches, especially the approach for additional request parameters in WFS requests. A special focus was on supporting the two use cases mentioned in Section 5 and delivering the feature data in NSG and DGIWG WFS 2.0 profiles.

6.1. Additional Request Parameters/Polling

In the polling approach, the client sends a request to the service indicating that a service operation should be run in asynchronous mode. The server responds with a status message indicating that the operation has been started. Subsequently, the client can request status information about the operation's execution. Once the operation is finished, the server will respond with the actual result of the operation. In this approach, the client needs to actively query the status of the operation running on the server.

The OGC WPS (OGC 14-065) supports this approach for asynchronous communication by defining an additional service parameter `mode` parameter for its `Execute` operation. The sequence of operations is shown in Figure 2.

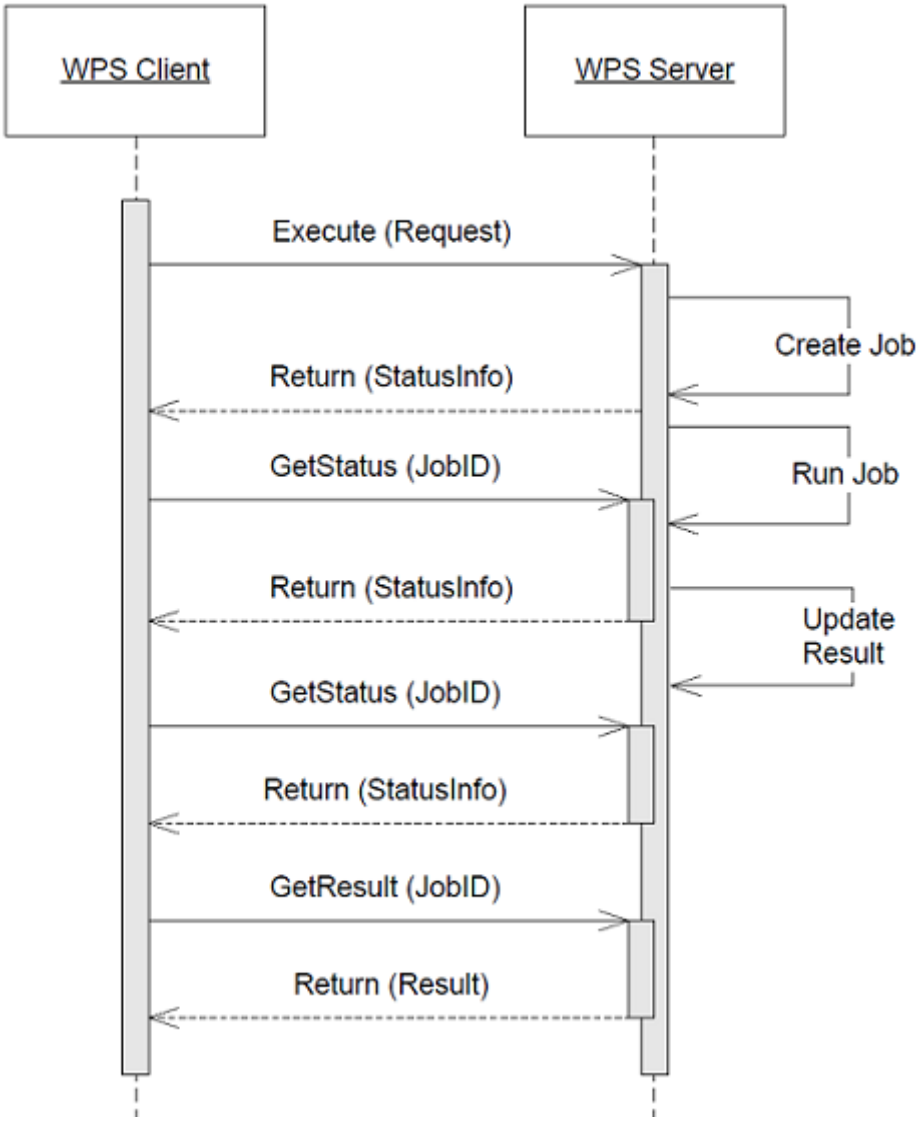

Figure 2. Sequence of asynchronous execution of Execute operation in OGC WPS (Source: OGC 14-065).

In Testbed 12, the WPS and its polling approach have been evaluated to support asynchronous retrieval of feature data sets from a WFS and WCS. In comparison, an additional WFS query parameter `responseFormat` was specified to directly support asynchronous communication for WFSs. As an example, the following URL can be used to indicate that a GetFeature operation should be executed in asynchronous mode and the result should be mailed to a certain email address:

```
http://tb12.cubewerx.com/a007/cubeserv?datastore=USGS&
   service=WFS&
   version=2.0.2&
   request=GetFeature&
   typeNames=NHDFlowline&
   count=100&
   outputFormat=application%2Fgml%2Bxml&
   responseHandler=mailto:tb12@pvretano.com&
   bbox=37.709077,-122.513476,37.839064,-122.351771,urn:ogc:def:crs:EPSG::4326
```

The concepts for both approaches and the evaluation results are described in detail in the OGC Testbed 12 Implementing Asynchronous Services Response Engineering Report (OGC 16-023) [http://docs.opengeospatial.org/per/16-023r3.html]. In a nutshell, asynchronous messaging was successfully

implemented using both approaches. Both provide a way to add asynchronous messaging to OGC Web Services in a more lightweight fashion compared to the OGC Publish/Subscribe specification. One potential drawback of the method with WPS facade is that the usual service requests need to be wrapped in a WPS request and response which causes a little overhead. However, the bigger the data sets, the less overhead occurs. An advantage is that the WPS facade approach can be used with the different OGC data services (WFS, WCS, SOS). Besides the advantage of less communication overhead, direct support of asynchronous communication in WFS using additional query parameters was also implemented without polling, but with a notification endpoint, where the operation's result should be sent to.

6.2. Publish/Subscribe

This approach is following the publish/subscribe messaging pattern that is commonly implemented by message brokers and specified for SOAP Web Services by OASIS in the Web Service Notification (WS-N) framework. Message providers, called publishers, publish their messages to a server. Clients can subscribe for these messages, but do not need to continuously query information. Instead, the clients provide an interface for receiving messages. The publish/subscribe operations may be either directly provided by an OGC Web Service or provided in an external component, the message (or notification) broker.

As mentioned above, the previously defined Sensor Event Service and Sensor Alert Service were following the publish/subscribe pattern and were evaluated in Testbeds 6 to 10. With the release of the OGC Publish/Subscribe Interface Standard in 1.0 (OGC 13-131r1) in 2016, a general standard is now available that can be used to extend existing OGC Web Services with asynchronous messaging.

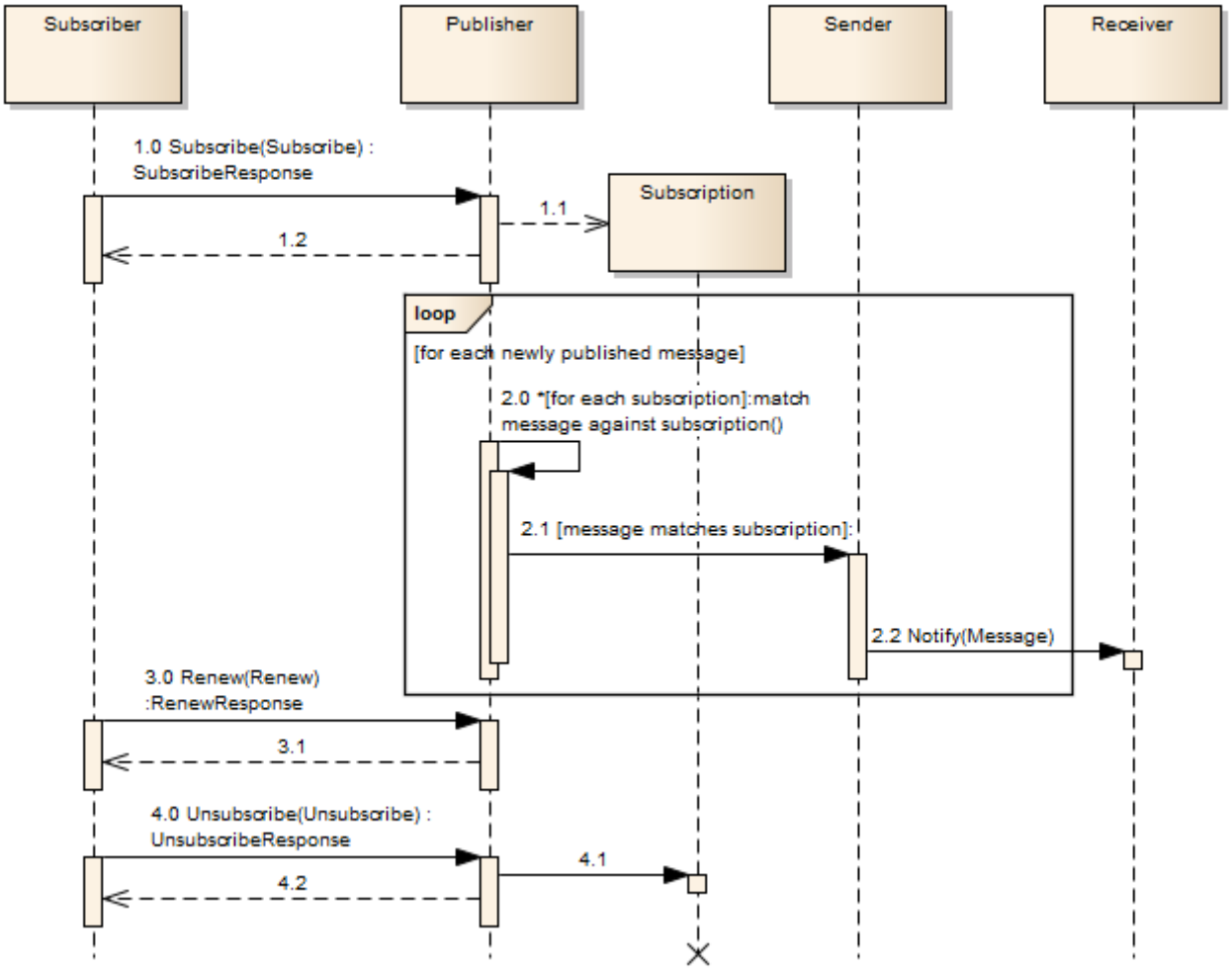

Figure 3. Basic Publish/Subscribe workflow (Source: OGC 13-131r1).

Figure 3 shows the general interaction between subscribers and publishers. OGC Web Services would usually act as a `Publisher`, where clients (`Subscriber`) can subscribe for certain information, e.g. in case a feature set in a WFS is updated or new features are available. The publisher is responsible for checking all subscriptions and sending messages to subscribers using `Sender` and `Receiver` components. Subscribers can also renew subscriptions or unsubscribe.

In Testbed 12, a publish/subscribe extension for the OGC Catalogue Service, version 2.02 (OGC 07-006r1) has been specified and evaluated. The results are described in the OGC Testbed 12 PubSub/Catalog ER (OGC 16-137) [http://docs.opengeospatial.org/per/16-137r2.html]. Furthermore, the application of the OGC PubSub standard in the aviation domain has also been evaluated and reported in the corresponding OGC Testbed 12 Asynchronous Messaging for Aviation ER (OGC 16-017) [http://docs.opengeospatial.org/per/16-017.html].

6.3. MQTT Extension

The OGC SensorThings API (OGC 15-078r6) takes a slightly different approach than utilizing the OGC Publish/Subscribe standard. Instead, an MQTT extension is specified that describes how to apply the Message Queue Telemetry Transport (MQTT) protocol for asynchronous communication in the SensorThings API. The interactions for creating and subscribing to observations in the SensorThings MQTT extension are shown in Figure 4.

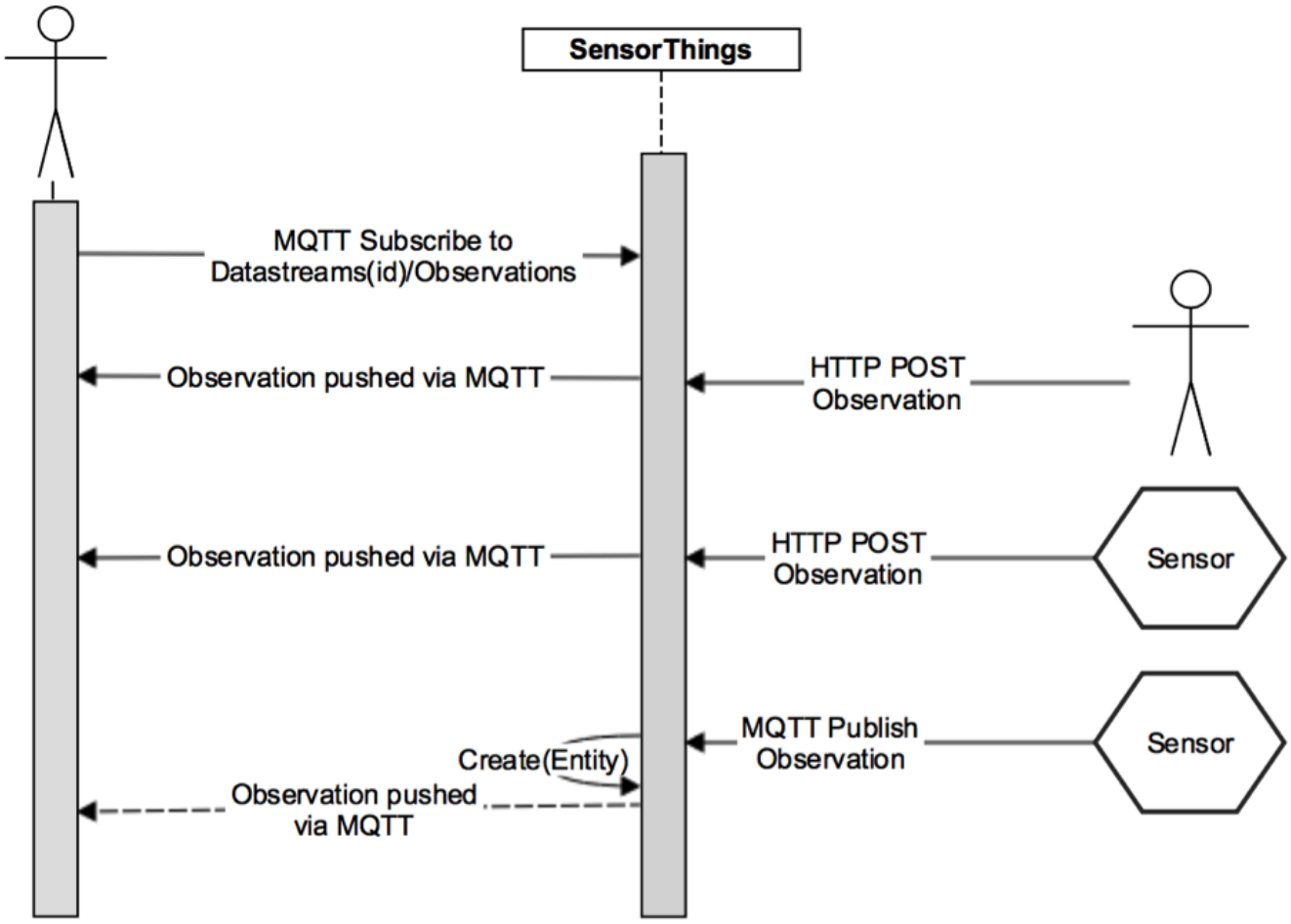

Figure 4. Interactions for creating and subscribing to observations in the SensorThings MQTT extension (Source: OGC 15-078r6).

Clients can subscribe to data streams/observations. Different sensor sources can push new observations to the SensorThings API. If clients are subscribed, they will receive the new observations via MQTT. MQTT has been originally designed for machine-to-machine communication and thus for messages with high frequency and small payload. As such, it is well-suited for publish/subscribe in sensor applications.

Chapter 7. NG119 Asynchronous WFS-1

7.1. Demonstration Scenario

In OGC Testbed 13, participants assessed the ability of an Asynchronous WFS to support simulated users in a Mass Migration Scenario over Syria and Jordan. In this scenario, large numbers of people have been displaced from the Daraa region of Syria to the Zaatari refugee camp in Jordan due to ongoing conflict. As the conflict ends 'de-escalation zones' are established by major powers and plans are made to return displaced people from refugee camps to the region. Understanding the situation on the ground and the infrastructure, as well as transporting these people from refugee camps into a former conflict zone is a major challenge for relief agencies. To accomplish this task, they must understand the environment and infrastructure in the region between Zaatari refugee camp and the Daraa region.

The following examples provide a brief sample of the scenario involved in testing the Asynchronous WFS approach.

Years of war have displaced hundreds of thousands from Darra city and region. Thousands have sought shelter at the sprawling Zaatari refugee camp (Figure 5).

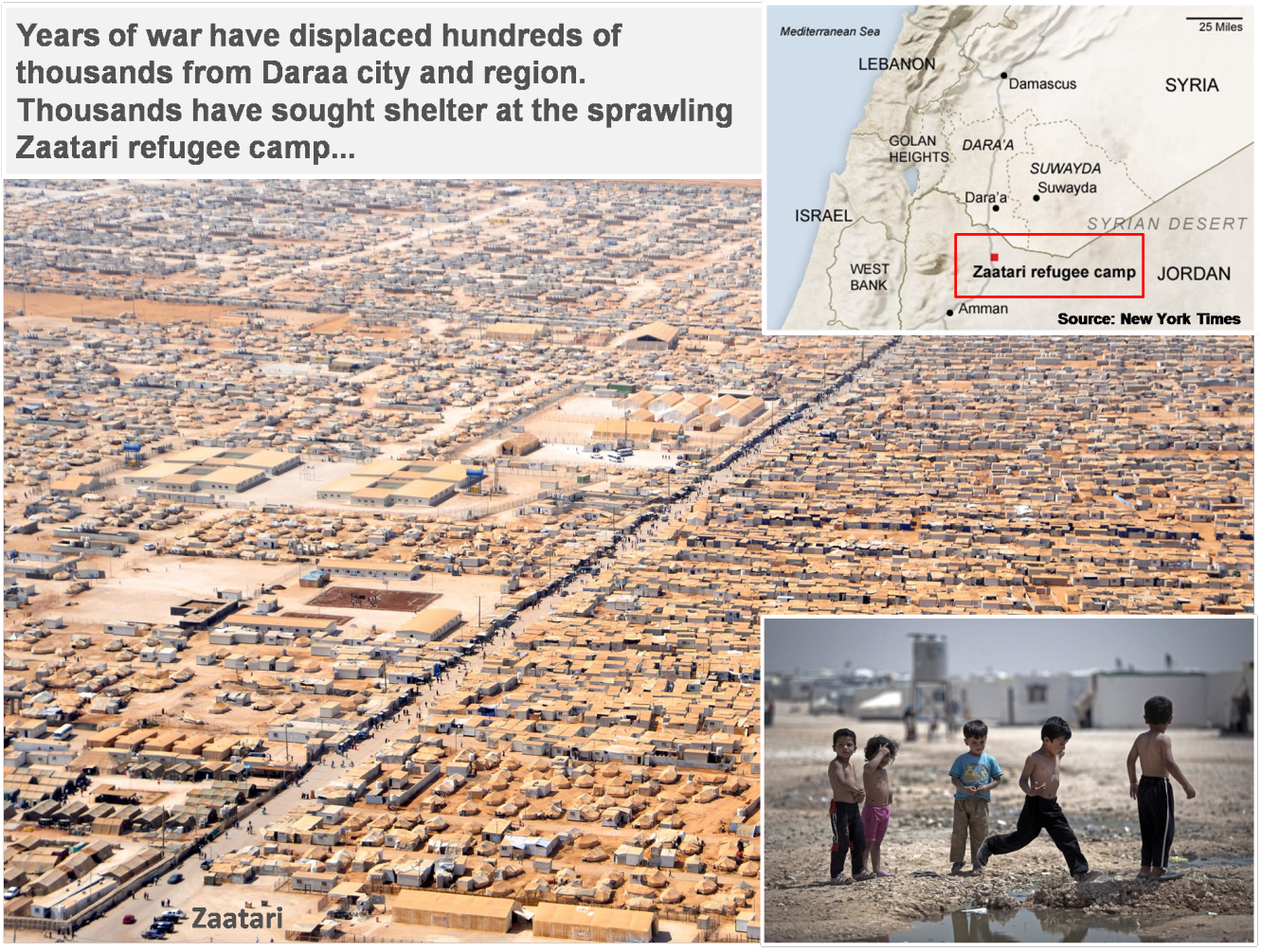

Figure 5. Illustration of Zaatari.

A ceasefire has been declared this year. Government and non-government organizations seek to

assist displaced populations at Zaatari in returning to Daraa, as illustrated in Figure 6.

Figure 6. Illustration of Zaatari.

The demonstration focuses on one type of user, the Analyst User.

Analyst User needs to execute the following tasks, as illustrated in Figure 7:

- Submits a request for NSG TDS OSM data over Daraa.
- Gets notified when ready.
- Accesses and moves out into the relief zone.

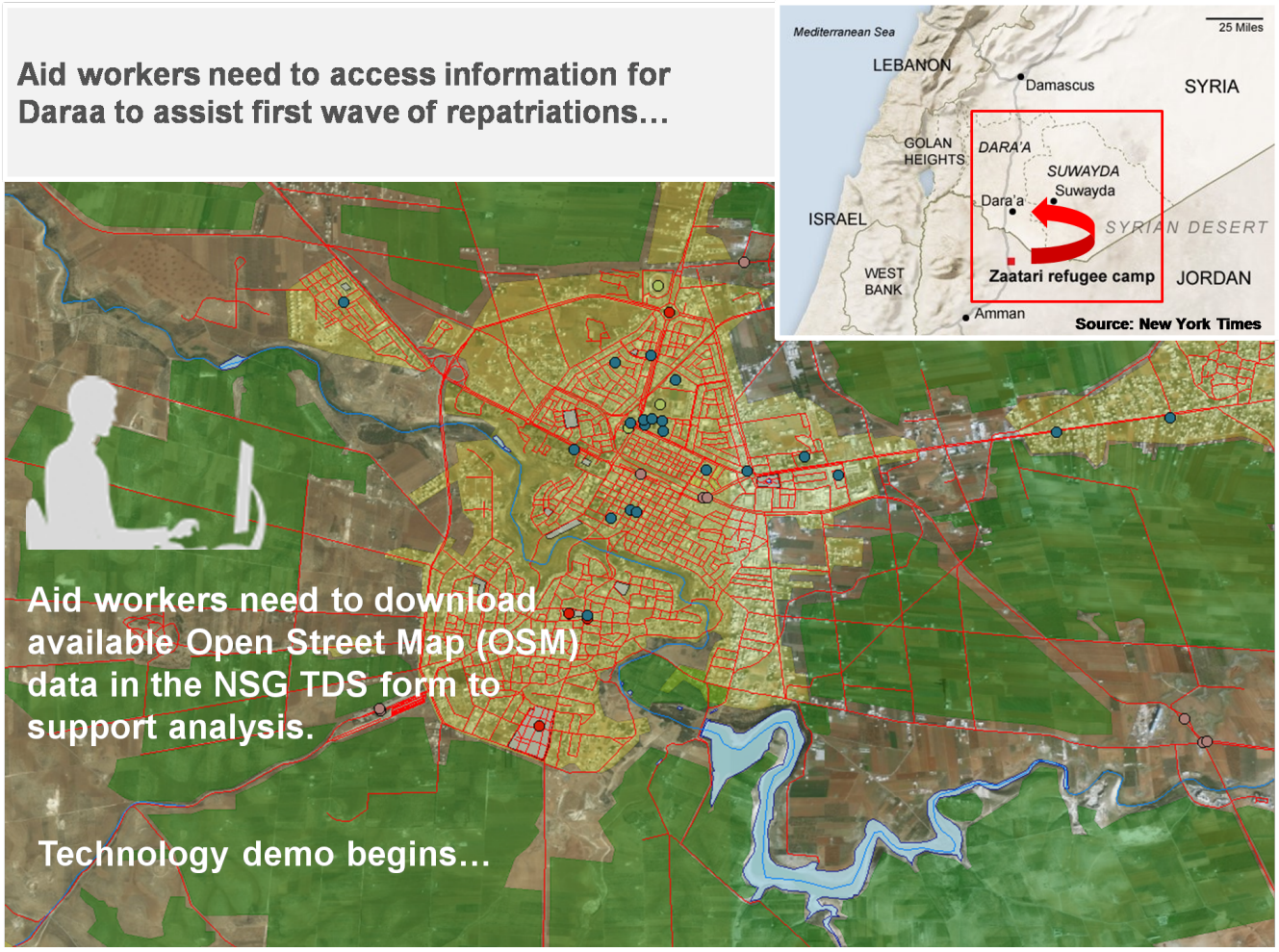

Figure 7. Illustration of use case for the analyst user.

Hence, the following User Stories are considered for the Analyst User (Figure 8):

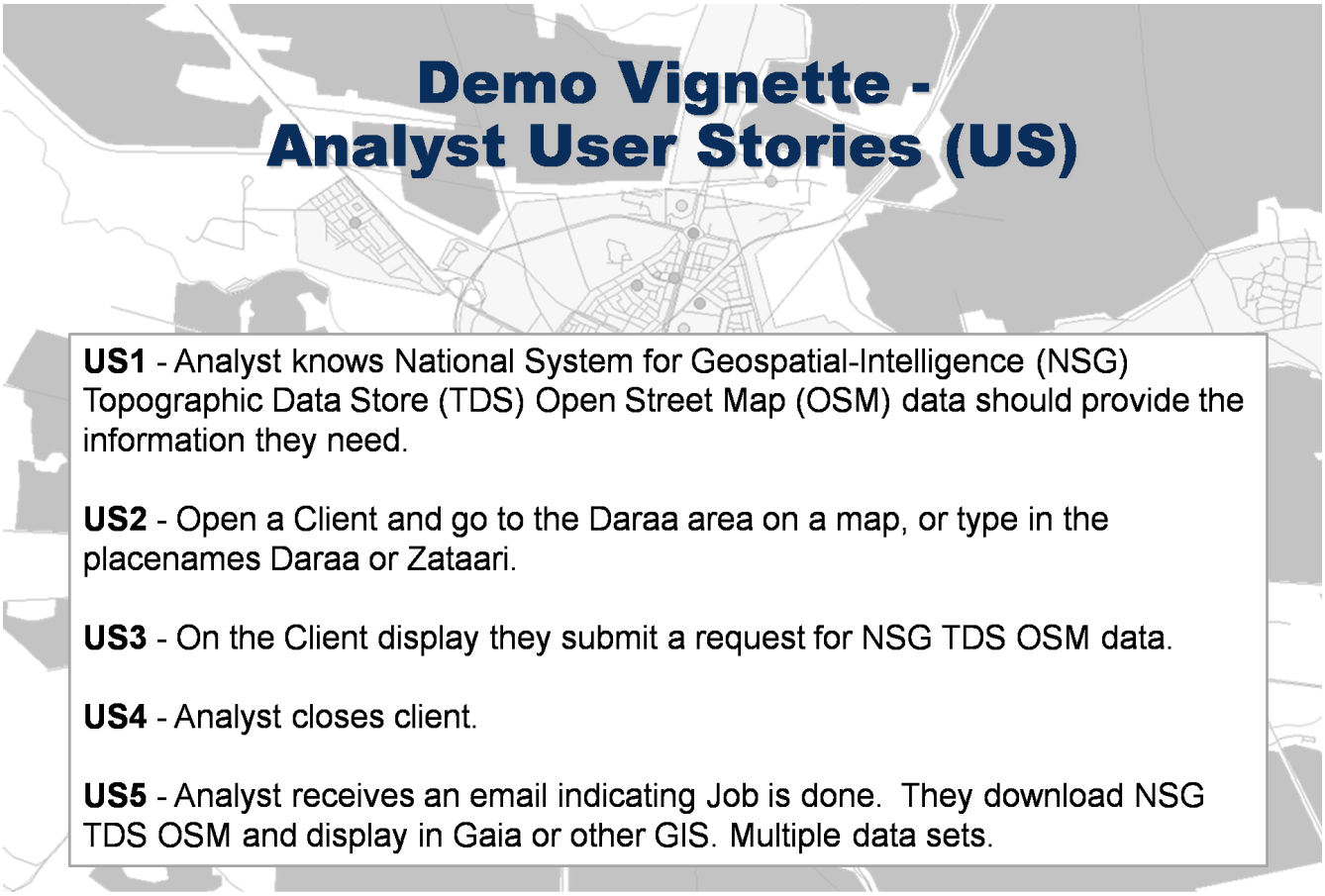

Figure 8. User stories for the analyst user.

7.2. Implementation

For integration testing, The Carbon Project implemented an Asynchronous Web Feature Server (Async WFS) according to the architecture outlined in section 5 of this ER. The Async WFS system components were hosted using cloud computing services technology based on The Carbon Project's CarbonCloud® geospatial data platform.

CarbonCloud® uses a cloud-based operating system called Windows Azure to run its 'fabric layer' - a cluster hosted at Microsoft data centers that manages computing and storage resources of the computers and provisions the resources to applications like Async WFS. Scaling, reliability, memory resources and load balancing are controlled by the Azure cloud - so developers can focus on deploying applications like Async WFS on the CarbonCloud platform.

Key Windows Azure technical features used by the Async WFS components include:

- **Virtual Machine Instances**: The most basic function that the cloud platform performs is to execute applications and virtual machines. Virtual machines make it possible to deploy applications and infrastructure to the cloud without changing existing code.
- **Data Management**: For storing data the system uses cloud-based Database Servers and Storage (Database, Tables and Blobs).
- **Web Applications**: Web Application Servers within the Virtual Machine allow The Carbon Project developers to control WFS Services management web applications.

For Testbed 13, the Async WFS system components implemented OSM data in a schema based on

the National System for Geospatial-Intelligence (NSG) Topographic Data Store (TDS) model, as illustrated in Figure 9. For the Async WFS-1 work item OSM was deployed as Geography Markup Language (GML) in the TDS Content Specification. Systems participating within the NSG may use the NSG TDS Content Specification to help ensure consistent geospatial data semantics, adopt common conditions for geospatial intelligence collection/exchange, support net-centric geospatial services, and achieve geospatial data interoperability. GML provides NSG TDS OSM data in XML form and JavaScript Object Notation (JSON) provides it in a lightweight exchange format.

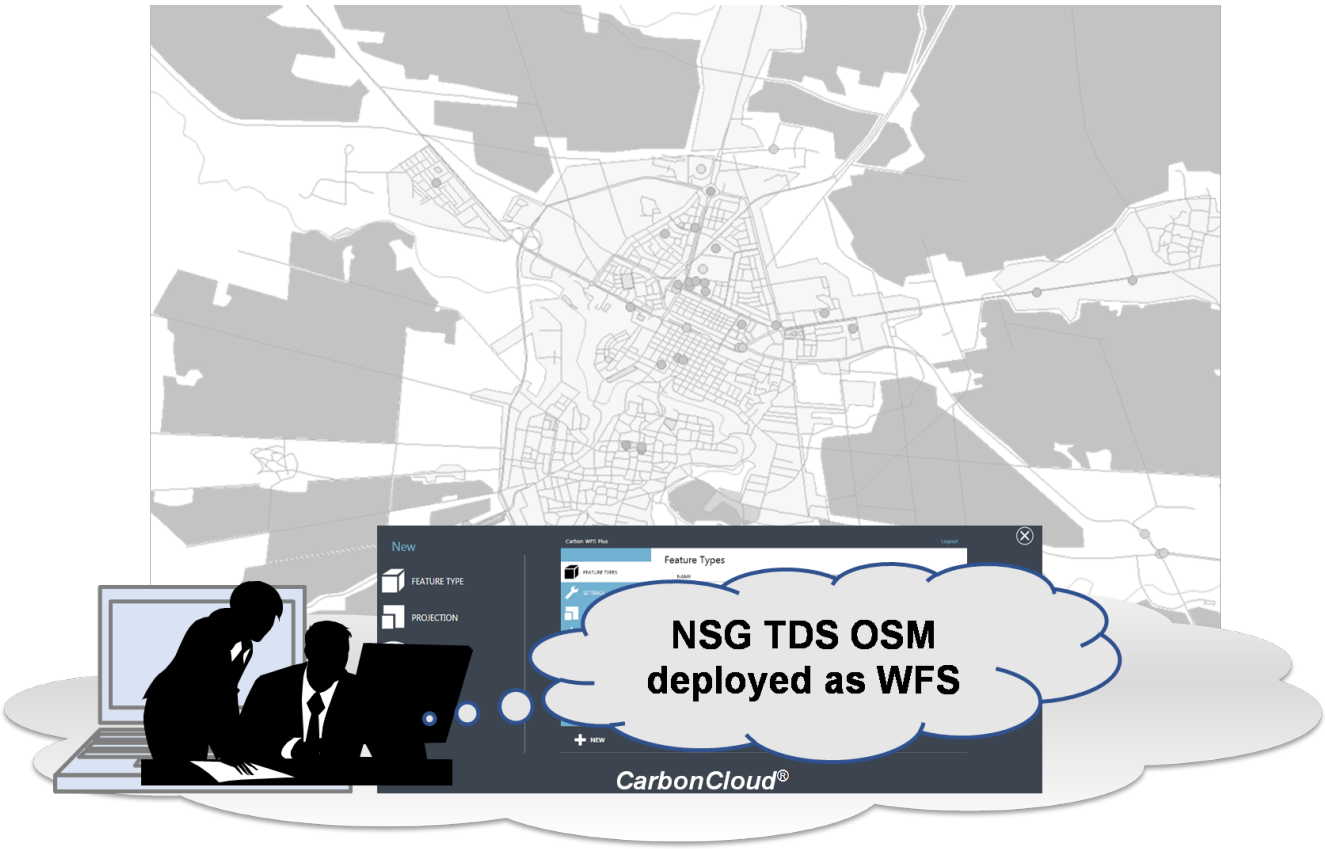

Figure 9. Overview on Asynchronous WFS-1 serving OSM data in NSG TDS format.

In Testbed 13, The Carbon Project's Async WFS-1 was tested in a humanitarian aid and mass migration scenario described in the Demonstration Scenario section. In particular, testing showed that it is feasible to request feature data to support relevant needs and have it delivered to users asynchronously to users via email and simple data download (see Figure 10 below).

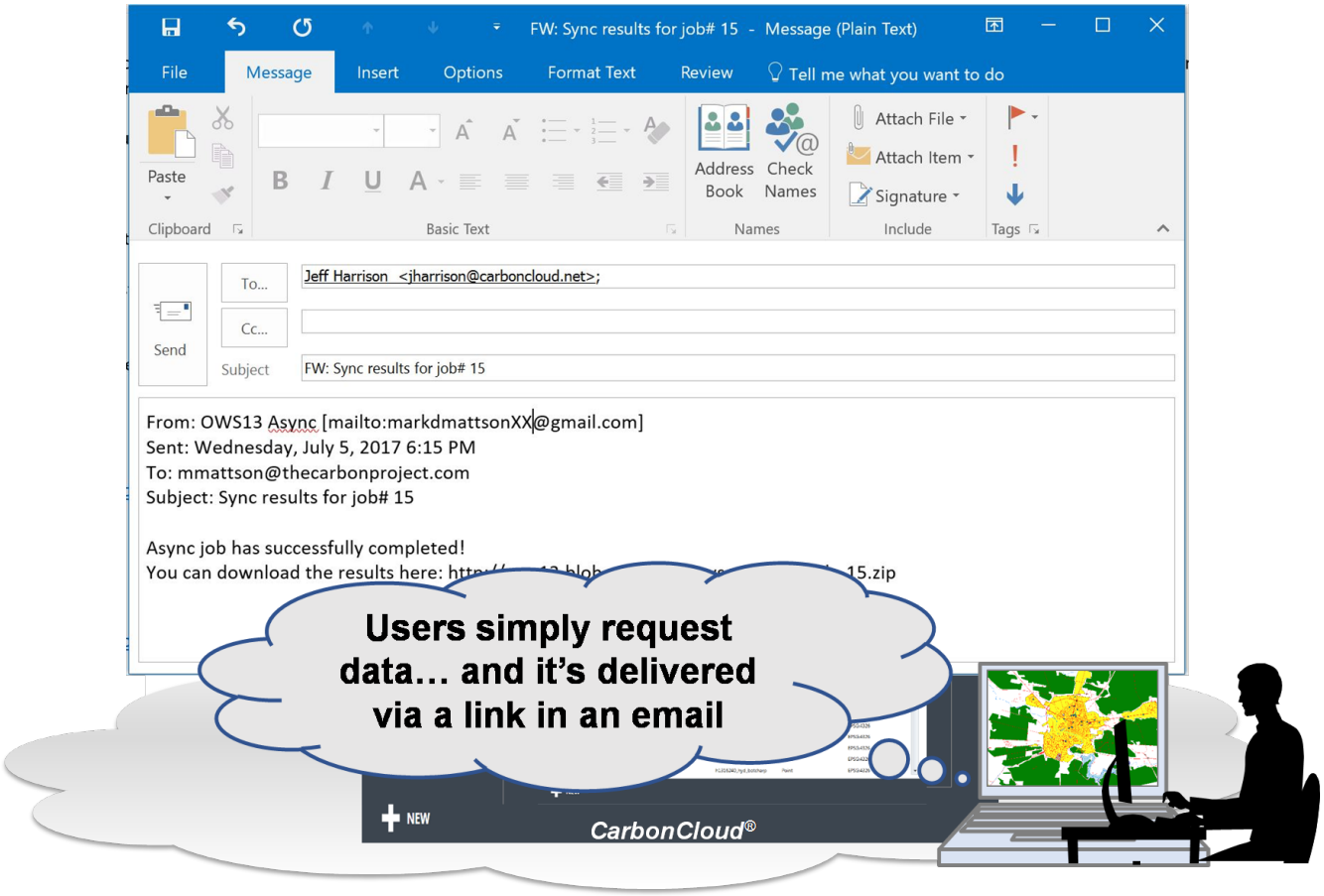

Figure 10. Asynchronous delivery of feature data.

The data delivered to users asynchronously implemented the National System for Geospatial-Intelligence (NSG) Topographic Data Store (TDS) OpenStreetMap (OSM) model using The Carbon Project's free Gaia application. The Structure Points (structpts) shapefile data from the NSG TDS OSM was used for testing and delivered as GML for the Asynchronous WFS in the example below (Figure 11).

Figure 11. NSG TDS OSM data delivered as GML.

The following example shows the use of the `mailto: response handler` in the Asynchronous WFS request:

```
http://ows13web.azurewebsites.net/wfs/featuretypes/structpts?responseHandler=jharrison
@thecarbonproject.com&outputFormat=gml
```

In this example, the Asynchronous WFS request is sent to a REST WFS endpoint with a GML output format. Other formats such as JSON may also be specified. An example of TDS OSM JSON output is included in that section that follows.

The following is an example of the acknowledgment message that the server generates in response to an Asynchronous WFS request:

```
<acknowledgment xmlns="http://www.opengis.net/ows">
  <link rel="monitor" href="http://ows13web.azurewebsites.net/wfs/jobs/45" xmlns=
"http://www.w3.org/2005/Atom" />
  <link rel="cancel" href="http://ows13web.azurewebsites.net/wfs/jobs/cancel/45"
xmlns="http://www.w3.org/2005/Atom" />
  <Status>pending</Status>
</acknowledgment>
```

Please note that the future acknowledgment message that the server generates in response to an Asynchronous WFS request may be in JSON format.

The following is an example of the notification message that the server generates in response to an

Asynchronous WFS request and sends to the `mailto:` response handler:

```
-----Original Message-----
From: markdmattsonXX@gmail.com [mailto:markdmattsonXX@gmail.com]
Sent: Wednesday, September 20, 2017 2:34 PM
To: jharrison@thecarbonproject.com
Subject: Sync results for job# 45

Async job has successfully completed!
You can download the results here: http://ows13.blob.core.windows.net/jobs/job_45.zip
```

The notification message from the server includes the following link to download the results of the Asynchronous WFS request:

```
http://ows13.blob.core.windows.net/jobs/job_45.zip
```

When the user clicks the link in the notification email the data from `Job 45` is downloaded. The GML data may be rendered in the geospatial application online or offline. For reference purposes it is shown in the Gaia application below (Figure 12) on top of satellite imagery tiles for the Daraa region.

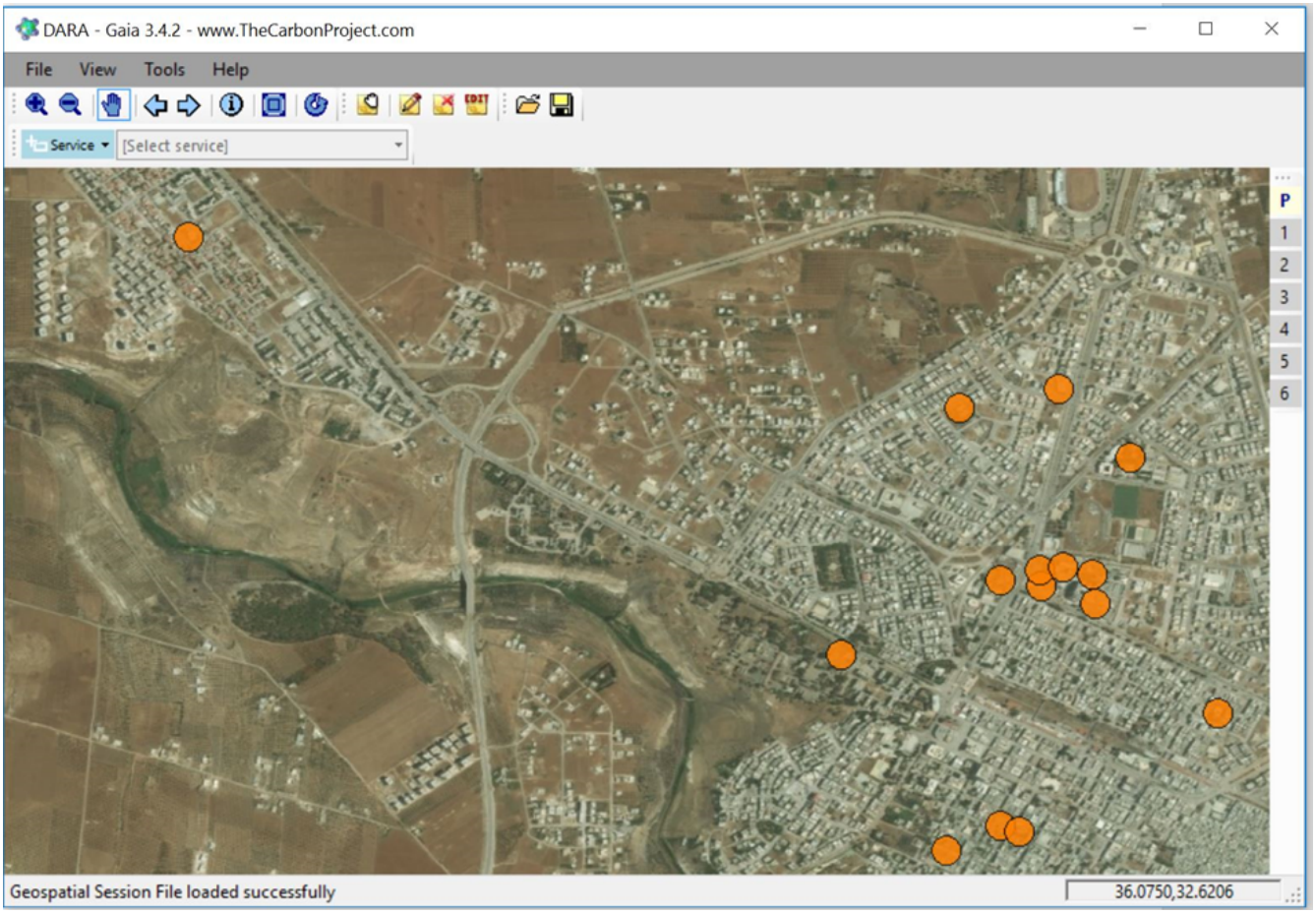

Figure 12. NSG TDS OSM data delivered as GML.

7.3. NSG TDS OSM as JSON

For the Async WFS-1 work item OSM was also deployed as JSON in the TDS Content Specification. JSON provides a lightweight exchange format and GeoJSON is a geospatial data interchange format based JSON. It defines several types of JSON objects and the manner in which they are combined to represent data about geographic features, their properties, and their spatial extents. GeoJSON uses a geographic coordinate reference system, World Geodetic System 1984, and units of decimal degrees.

GeoJSON objects may represent a region of space (a Geometry), a spatially-bounded entity (a Feature), or a list of features (a Feature Collection). Features in GeoJSON contain a geometry object and additional properties, and a Feature Collection that contains a list of features. An example of a prototype TDS OSM JSON is shown below. It should be noted that the TDS data model is extensive and it may be simplified since many of the properties are not able to be populated with the information provided in OSM.


```
{ "type": "FeatureCollection", "features": [{"type":"Feature","_id":"1","geometry":{
"type": "Point", "coordinates": [36.1009302267023, 32.6123398833619] },"properties":{
  "cp:ADR": "No Information",
  "cp:AOO": -999999.0,
  "cp:ARA": -999999.0,
  "cp:AWP": -999999,
  "cp:BEN": "noInformation",
  "cp:CAA": -999999,
  "cp:CCN": "No Information",
  "cp:CDR": "No Information",
  "cp:FFN": 931,
  "cp:FFN2": -999999,
  "cp:FFN3": -999999,
  "cp:F_CODE": "AL013",
  "cp:HGT": -999999.0,
  "cp:LMC": -999999,
  "cp:LZN": -999999.0,
  "cp:OTH": "noInformation",
  "cp:PCF": -999999,
  "cp:SAX_RS1": "No Information",
  "cp:SAX_RS2": "No Information",
  "cp:SAX_RS3": "noInformation",
  "cp:SAX_RS4": "noInformation",
  "cp:SAX_RS5": "No Information",
  "cp:SAX_RS6": "noInformation",
  "cp:SAX_RS8": "No Information",
  "cp:SAX_RS9": "No Information",
  "cp:SAX_RX1": "noInformation",
  "cp:SAX_RX2": "noInformation",
  "cp:SAX_RX5": "noInformation",
  "cp:SAX_RX6": "noInformation",
  "cp:SAX_RX7": "noInformation",
  "cp:SAX_RX8": "noInformation",
```

```
  "cp:SAX_RX9": "No Information",
  "cp:SAX_RY0": "No Information",
  "cp:SAX_RY1": "noInformation",
  "cp:SAX_RY2": "noInformation",
  "cp:SSR": -999999,
  "cp:SSR2": -999999,
  "cp:SSR3": -999999,
  "cp:TEL": 0,
  "cp:UFI": "c7d5c699-6557-4f37-8174-15a75fa1fe17",
  "cp:VOI": "noInformation",
  "cp:WID": -999999.0,
  "cp:ZI001_SDP": "No Information",
  "cp:ZI001_SDV": "noInformation",
  "cp:ZI001_SPS": -999999,
  "cp:ZI001_SRT": "noInformation",
  "cp:ZI001_VSC": "noInformation",
  "cp:ZI001_VSD": "noInformation",
  "cp:ZI001_VSN": "No Information",
  "cp:ZI004_RCG": "noInformation",
  "cp:ZI005_FNA": "Ø§Ù„Ø¬Ø§Ù…Ø¹ Ø§Ù„Ù‡Ø±Ø±ÙŠ",
  "cp:ZI005_NFN": "No Information",
  "cp:ZI006_MEM": "<OSM>{\"osm_id\":\"1080136521\",\"hoot:layername\":
\"planet_osm_point\",\"tags\":\"\\\"name\\\"=>\\\"Ø§Ù„Ø¬Ø§Ù…Ø¹ Ø§Ù„Ù‡Ø±Ø±ÙŠ
\\\", \\\"amenity\\\"=>\\\"place_of_worship\\\", \\\"religion\\\"=>\\\"muslim\\\"
\"}</OSM>",
  "cp:ZI020_GE4": "noInformation",
  "cp:ZI026_CTUC": 5,
  "cp:ZI026_CTUL": -999999,
  "cp:ZI026_CTUU": -999999,
  "cp:ZSAX_RS0": "noInformation",
  "cp:ZSAX_RX0": "noInformation",
  "cp:ZSAX_RX3": "noInformation",
  "cp:ZSAX_RX4": "noInformation",
  "cp:ZVH": -999999.0,
  "cp:FCSUBTYPE": 100083,
  "cp:HEI": 0.0,
  "cp:ATB": -999999,
  "cp:BNF": -999999,
  "cp:HST": -999999,
  "cp:HST2": -999999,
  "cp:HST3": -999999,
  "cp:MFB": -999999,
  "cp:MUB": -999999,
  "cp:RLE": -999999,
  "cp:VCM": -999999,
  "cp:VCM2": -999999,
  "cp:VCM3": -999999,
  "cp:ZI014_PBY": -999999,
  "cp:ZI014_PBY2": -999999,
  "cp:ZI014_PBY3": -999999,
  "cp:ZI014_PPO": -999999,
```

```
  "cp:ZI014_PPO2": -999999,
  "cp:ZI014_PPO3": -999999,
  "cp:ZI014_PRW": -999999,
  "cp:ZI014_PRW2": -999999,
  "cp:ZI014_PRW3": -999999,
  "cp:ZI018_WIT": -999999,
  "cp:ZI037_REL": 2,
  "cp:ZI037_RFA": -999999,
  "cp:BSU": 0,
  "cp:ETY": 0,
  "cp:SRL": 0,
  "cp:CRM": 0,
  "cp:PYC": 0,
  "cp:PYM": 0,
  "cp:TOS": 0,
  "cp:COS": 0,
  "cp:GUG": 0,
  "cp:SPT": 0,
  "cp:TTC": 0,
  "cp:TTC2": 0,
  "cp:TTC3": 0
}}
```

Chapter 8. NG120 Asynchronous WFS-2

8.1. Demonstration Scenario

The scenario for the Async WFS-2 is also to download a large data set from a WFS. Instead of providing OSM data in the TDS format, data from the Ordnance Survey of Great Britain is provided by the WFS implementing the DGIWG WFS profile.

In December 2015, Northwest of England suffered a severe flood, many places were flooded, including the historic city of Yorkshire. On December 26th, the Yorkshire flood control system suddenly opened the flood gates, so that the disaster became more serious and made the local people dissatisfied. The authorities explained that they were concerned that the electronic pumps inside were flooded by water and decided to open the flood gates [1]. The situation in the Yorkshire area worsened and thousands of residents needed to be evacuated, causing local resident's indignation. Illustrations of the floods are shown in Figure 13 and Figure 14.

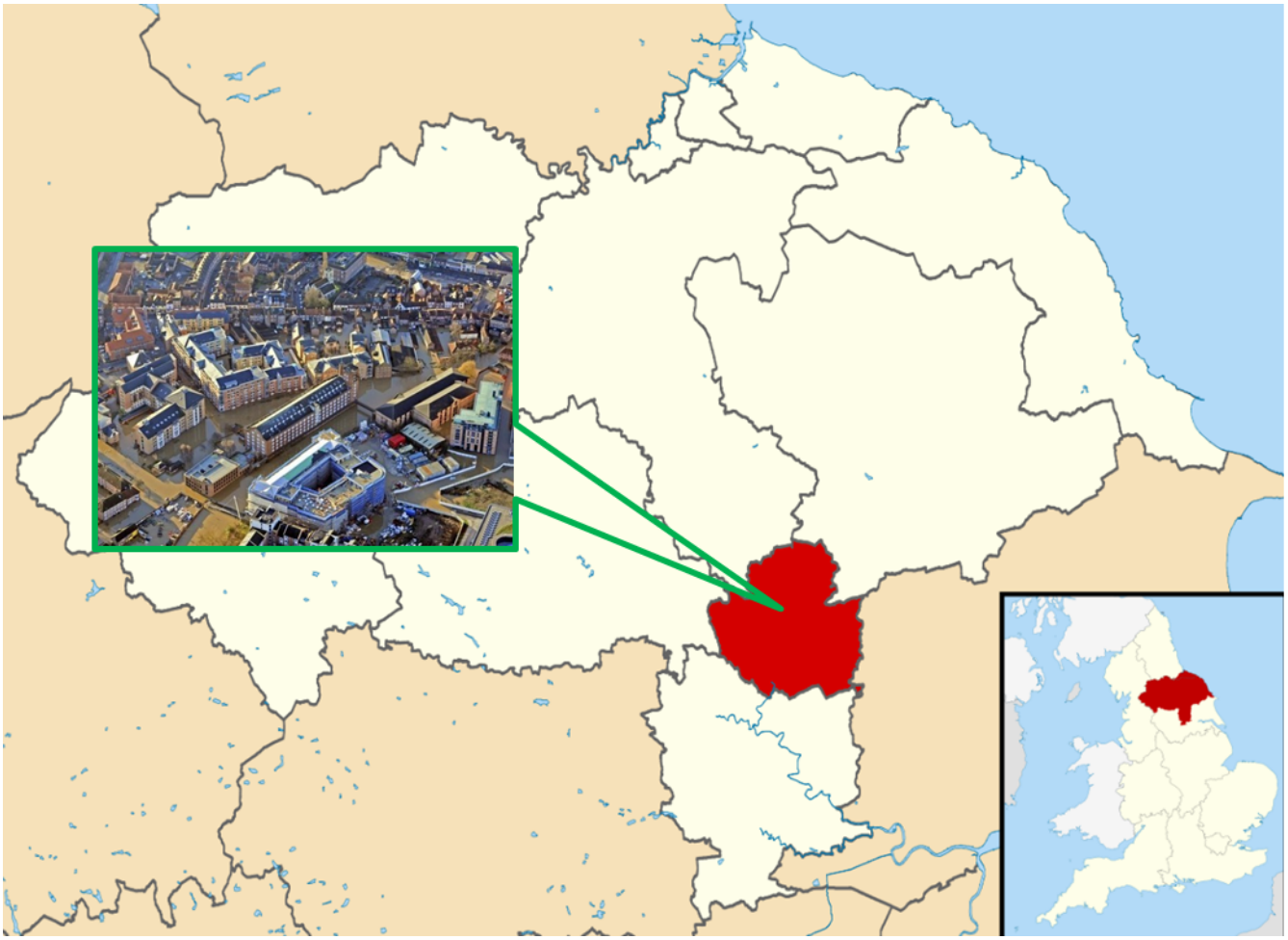

Figure 13. Flooding in city of Yorkshire.

Figure 14. Flooding events.

In this case, if the authorities could pre-simulate the degree of flooding with real-time WFS service, they could avoid a worsening disaster. In Figure 15, Government units can also provide real-time WFS services for hydro-node and flooding map, so that citizens can understand the current flooding situation through the Application. The scientist or application providers can calculate and simulate the degree of flooding with overlap map from the asynchronous WFS services, so they could send a warning to the citizens. The raw OSM map is shown in Figure 15, an overlay with flooded areas provided by the Async WFS is shown in Figure 16.

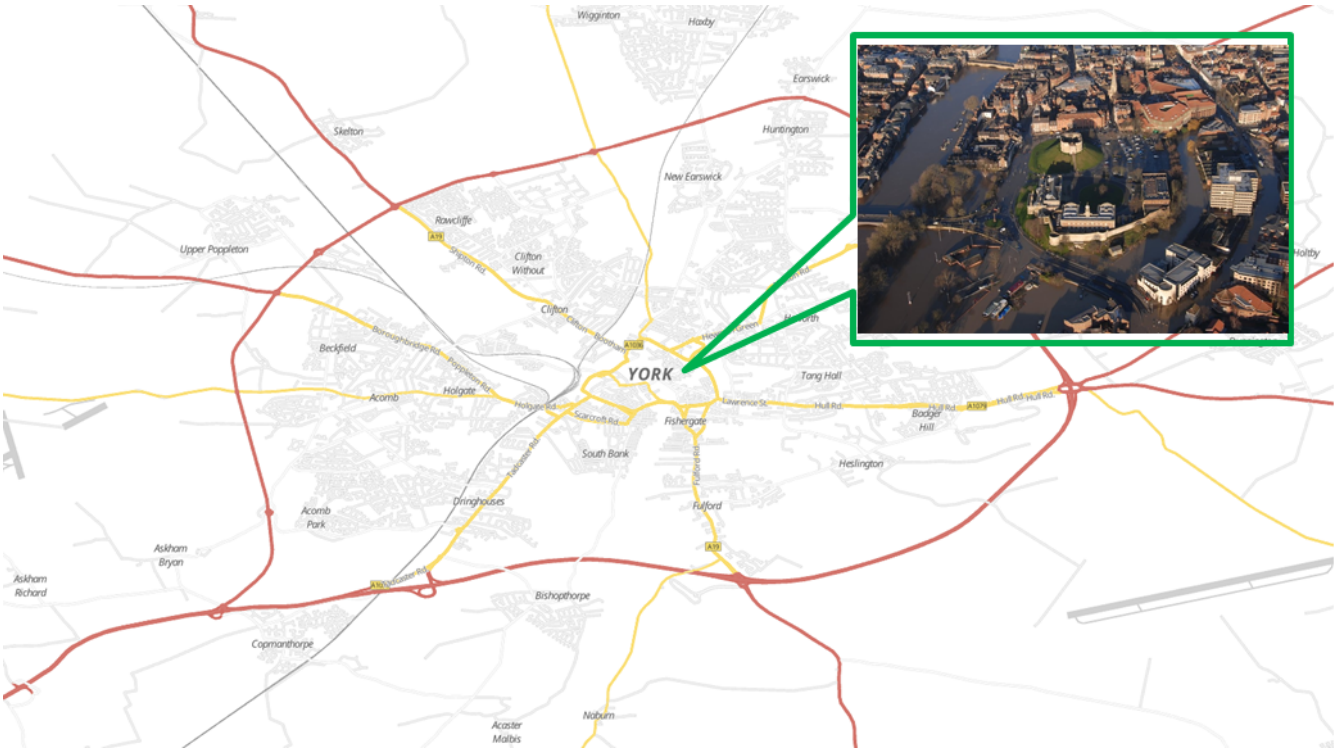

Figure 15. Displayed York city on OSM.

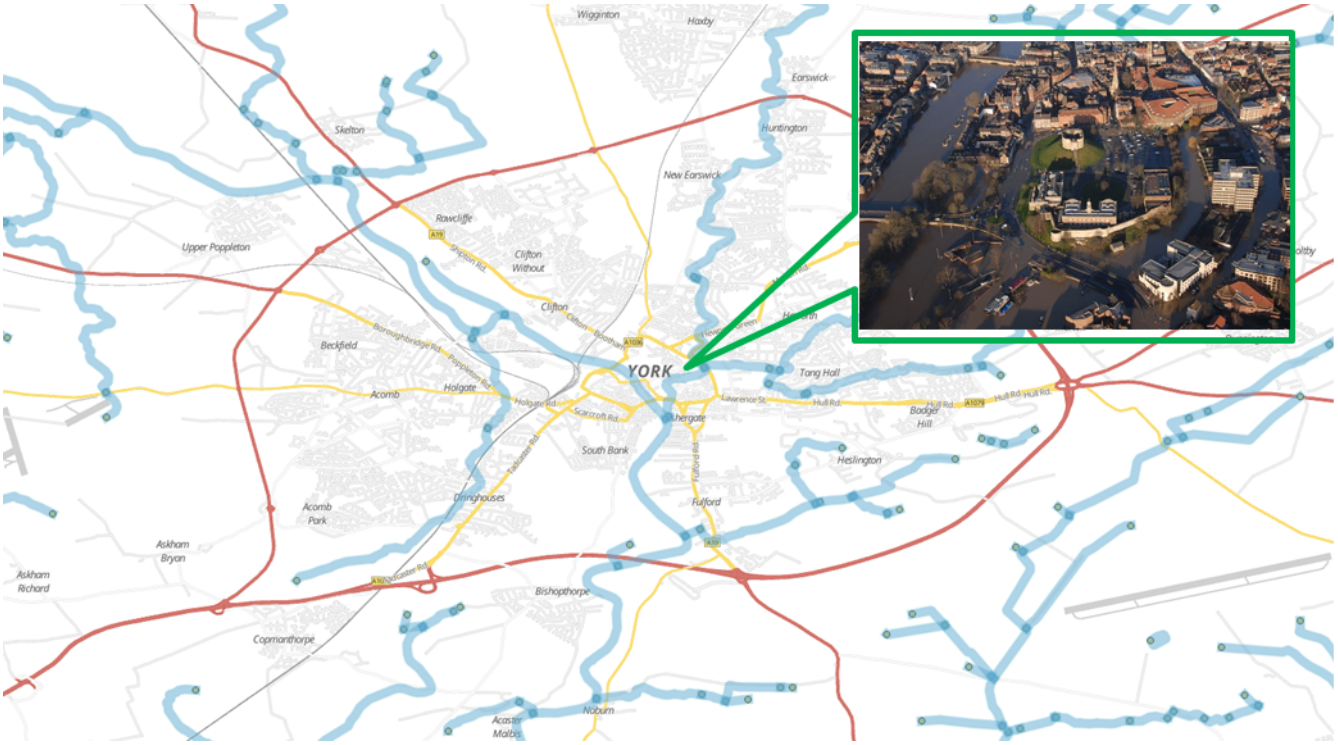

Figure 16. Overlay asynchronous WFS services on OSM.

8.2. Implementation

The Asynchronous WFS-2 provides four service operations: GetCapabilities, GetFeature, GetStatus, and Cancel. These four operations demonstrate the capability to deal with Big Data query scenarios as illustrated in Figure 17. GetStatus and Cancel are optional and not required. In the scenario, a user can operate client components (procedure 1.1). The client would send a GetCapabilities request to the WFS-2 (procedure 1.2 and 1.3). The Async WFS-2 will respond with the Capabilities document immediately and send it back to the Client (procedure 1.4). For the asynchronous scenario, a customized GetFeature operation with an extra `mailto` parameter, containing the subscription email address, needs to be sent to the request handler (procedures 2.1-2.3). The WFS-2 will then return a message with operation ID information (procedure 2.4). The client can trace the progress via the operation ID (procedures 3.1 and 3.2) or it can cancel the operation using the operation ID (procedures 3.3-3.5). If the operation is not canceled, the WFS-2 will generate the download URL (procedures 2.5 and 2.6) after all backend query processing has finished, and deliver an e-mail with URL to the subscribed user (procedures 2.7).

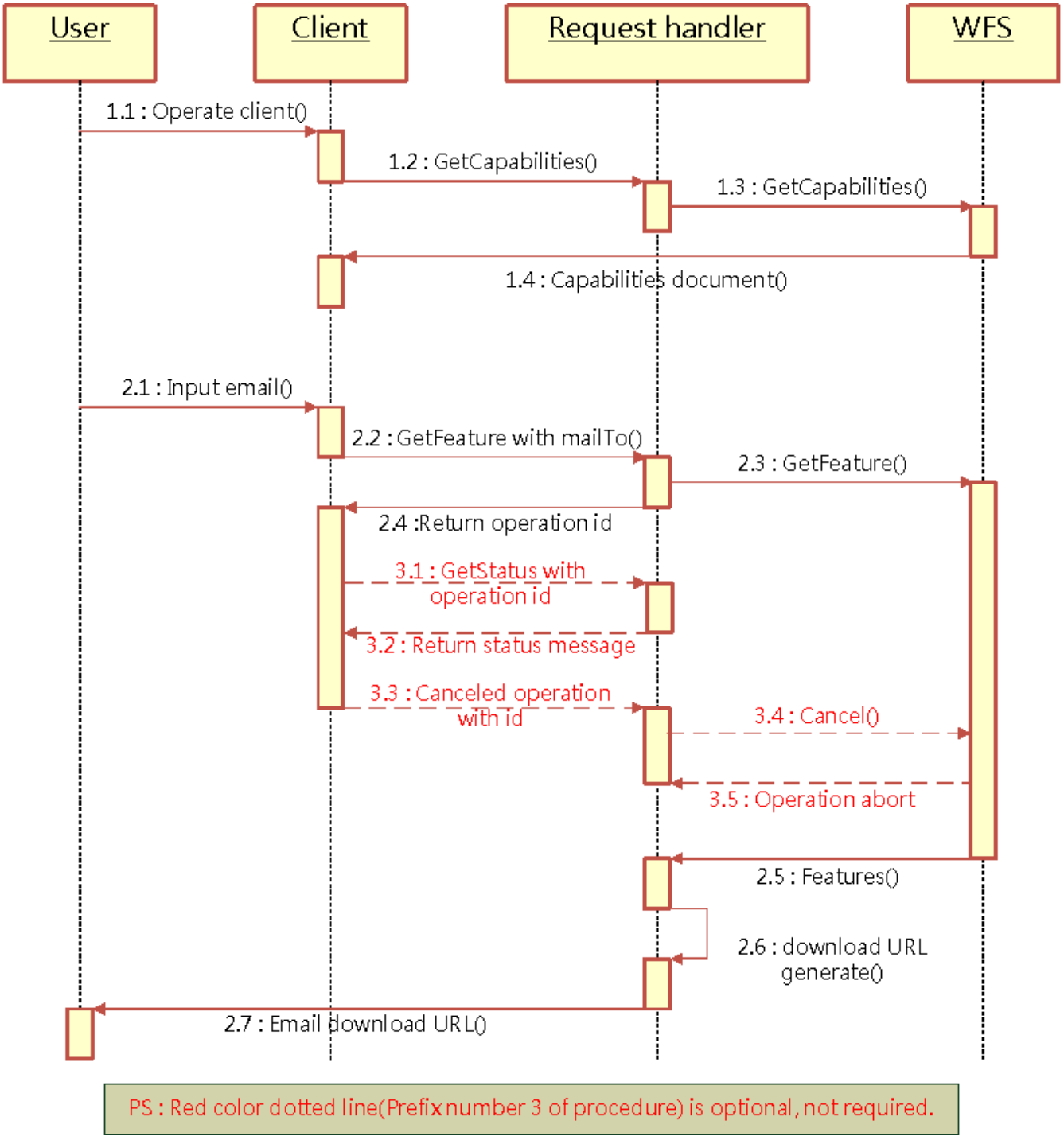

Figure 17. Sequence diagram illustrating the usage of Asynchronous WFS-2.

8.2.1. Data source

The data used for this testbed were downloaded from the Ordnance Survey Open Data website, including Road-node map and Hydro-Node map in U.K. (Figure 18).

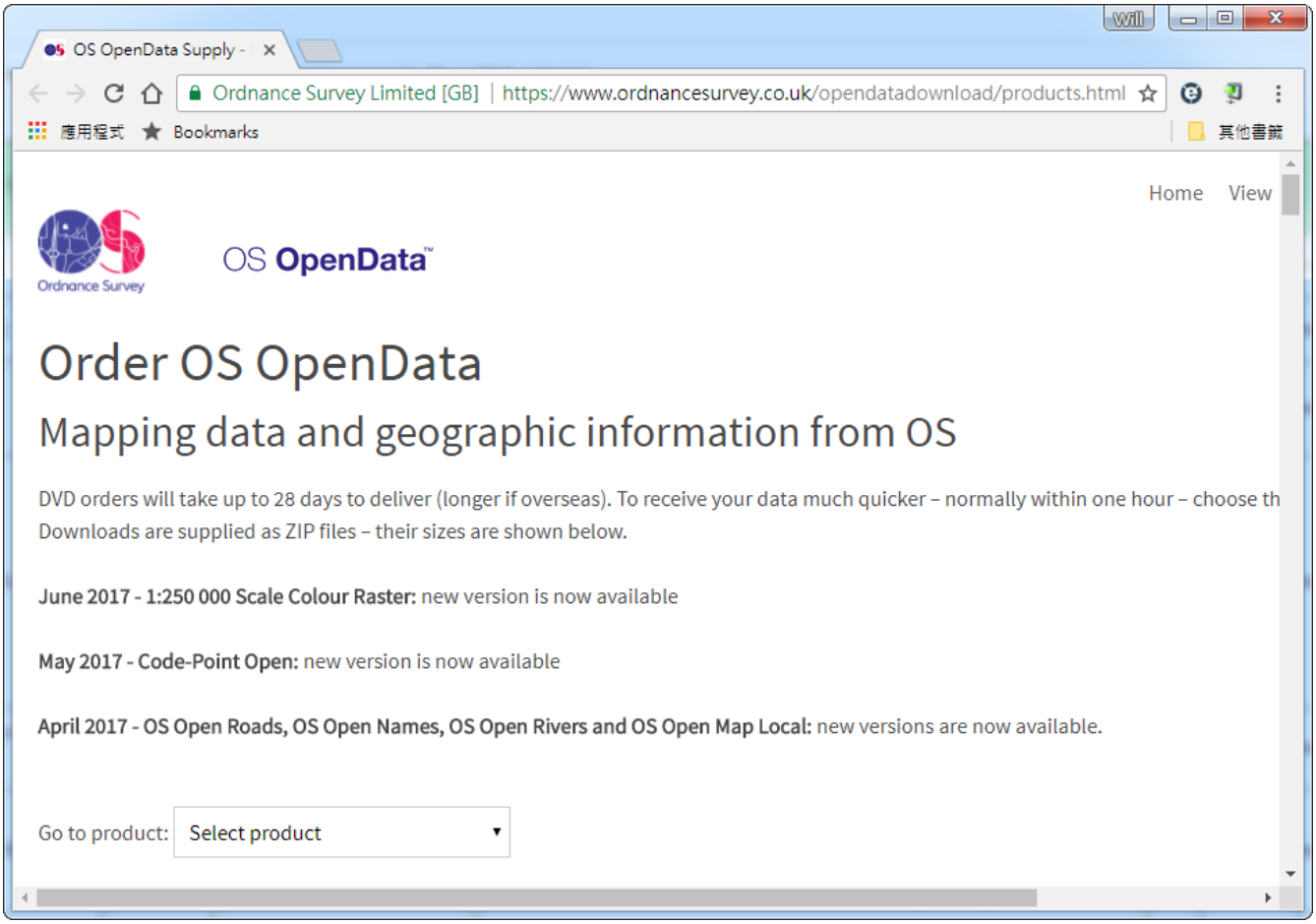

Figure 18. Screenshot of data download website of Ordnance Survey, UK.

8.2.2. GetCapabilities operation

Clients can call the Async-WFS2 service with `GetCapabilities` using the following URL: (http://140.134.48.46/WFS/WFS.ashx?typeName=cite:HydroNode_WGS84&request=GetCapabilities). A screenshot has been taken showing the `WFS_Capabilities` response (Figure 19). It describes the methods that the WFS-2 service provides.

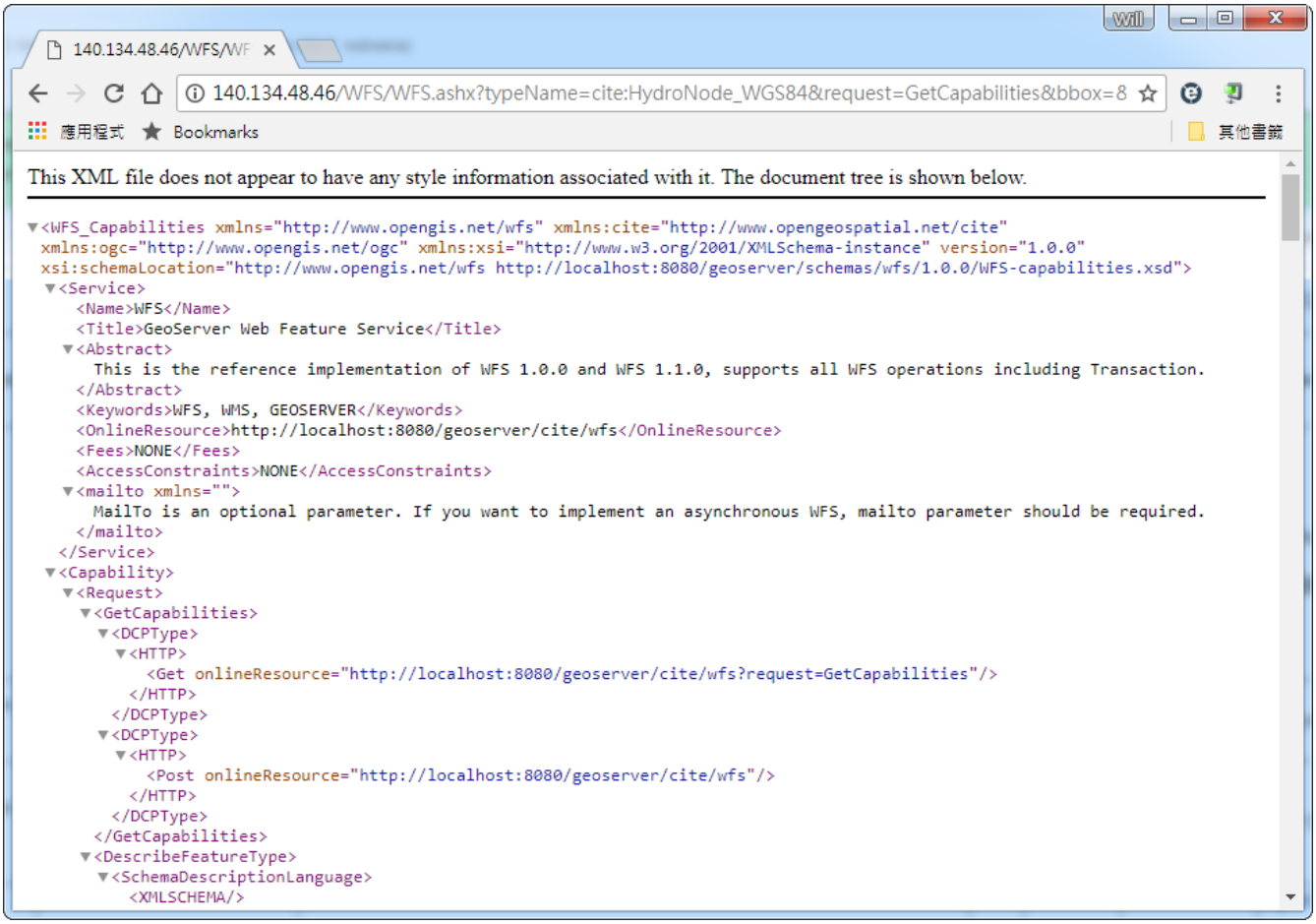

Figure 19. XML excerpt of the Capabilities Document returned.

8.2.3. GetFeature operation

Besides `GetCapabilities`, a Client can call the Async-WFS2 service with `GetFeature` as well. The testing URL is like:

```
http://140.134.48.46/WFS/WFS.ashx?typeName=cite:HydroNode_WGS84&request=GetFeature&bbox=&mailto=chihwai.kuan@gmail.com
```

The difference from `GetCapabilities` is the `mailto` parameter, the screenshot is the mail content including the download URL returned by the service. The user can download the GML file by resolving this URL.

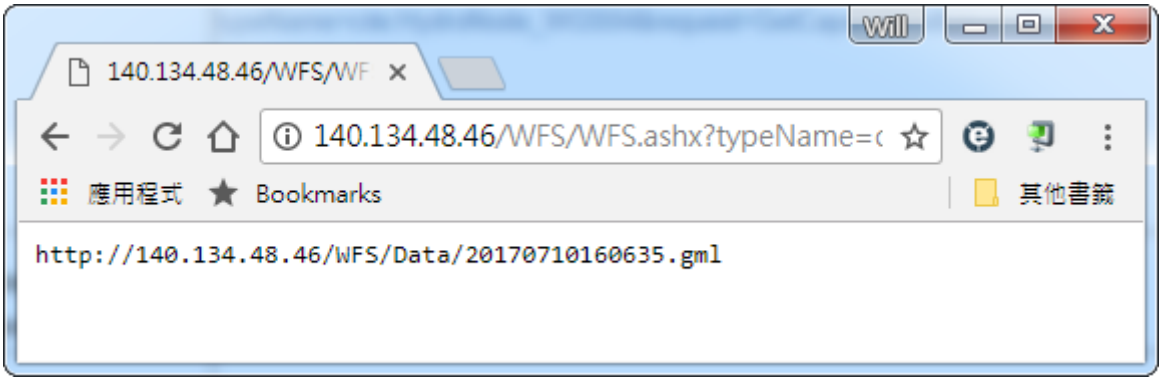

Figure 20. URL for downloading a feature data set.

Figure 21. Mail content for downloading a feature data set.

8.2.4. GetStatus operation

The client can call the `GetStatus` to retrieve the operation status in any procedure stage. The testing URL is like:

```
http://140.134.48.46/WFS/WFS.ashx?typeName=cite:M_WGS84&request=GetStatus&OperationID=
20170809142912
```

8.2.5. Cancel operation

If the client would like to abort the operation during the service processing, it could send a `Cancel` request containing the operation ID to abort it. Testing URL is like:

```
http://140.134.48.46/WFS/WFS.ashx?typeName=cite:M_WGS84&request=Cancel&OperationID=201
70809142912
```

8.2.6. View the result

The final result can be viewed by QGIS software. The user can access it or overlay the map layer onto GIS portal.

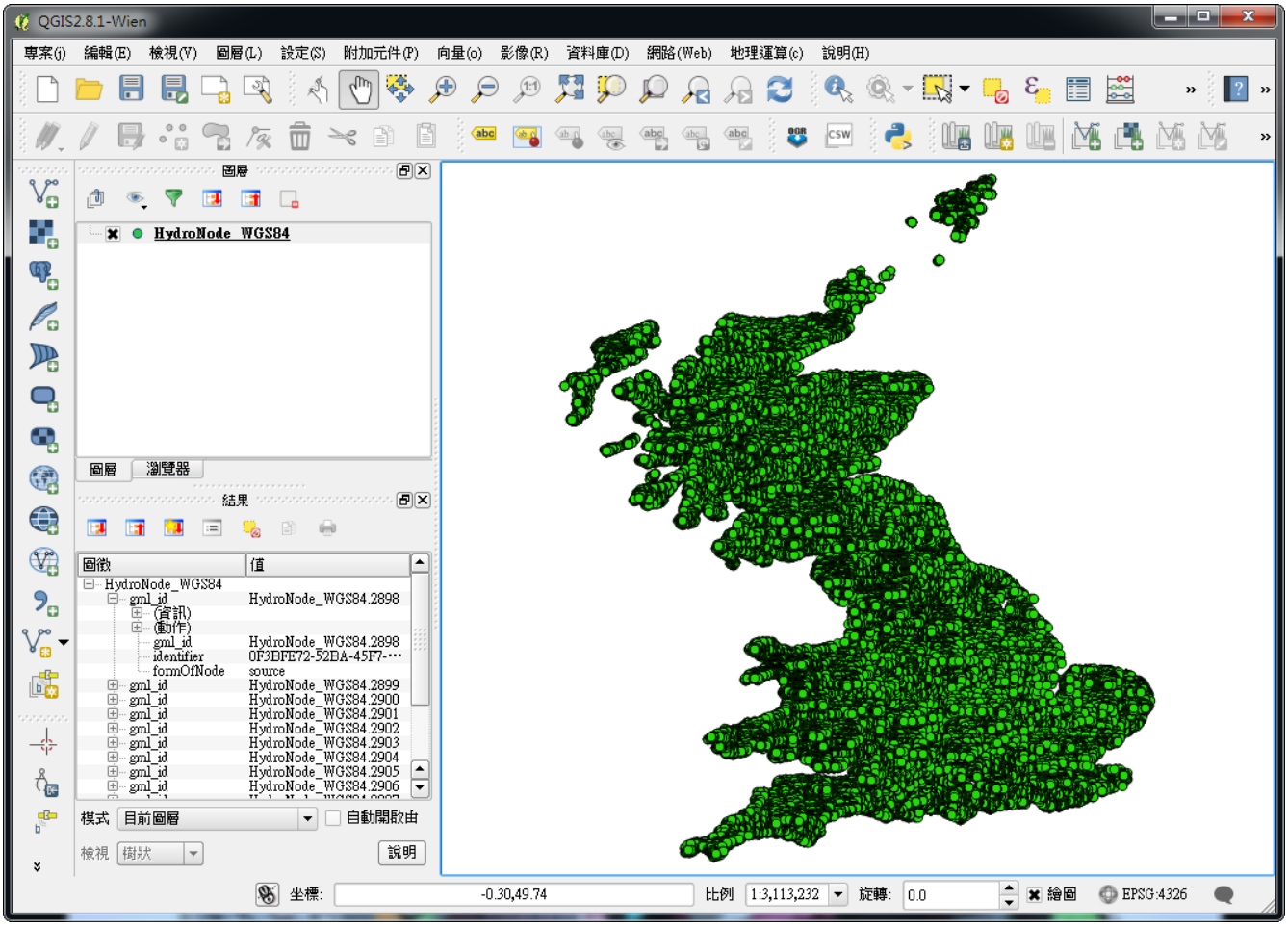

Figure 22. Downloaded feature data set visualized in Quantum GIS.

8.2.7. Scenario UI

In order to simulate the WFS-2 service scenario easily, we designed a simple scenario UI to generate request command strings, and to call WFS2. The dropdown list (Request) includes GetCapabilities, GetFeature, GetStatus, and Cancel methods. The Layer dropdown list contains all kinds of layers available to the service. The boundary box of map layer is entered into the Boundary field, if client wants to select a specific range that it needs. The client's mail address is entered into the Mailto field. The Operation ID field is used by the GetStatus/Cancel methods.

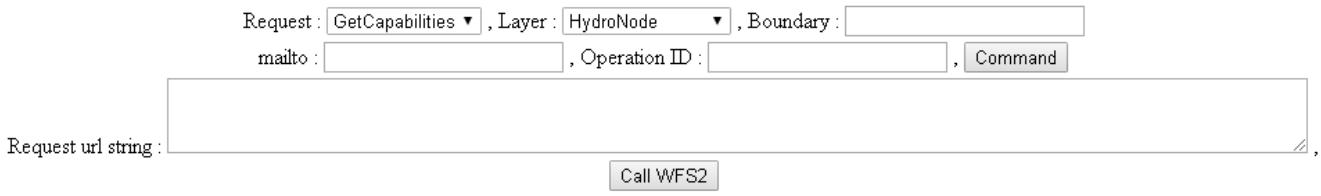

Figure 23. Scenario UI.

Chapter 9. Recommendations & Future Work

9.1. Exploring the role of Asynchronous Web Feature Services in Geospatial Enterprise Architectures

Web Feature Services are still mostly used in a synchronous communication fashion for rendering feature data sets in maps following the common pattern of 'request-response-parse-render'. However, the rapidly increasing availability of large feature data sets, e.g. OpenStreetMap features, require novel approaches. The Asynchronous WFSs developed in this testbed demonstrate that the extensions for asynchronous communication are able to solve the issue of transferring large feature data sets. However, the roles that such Asynchronous WFSs may play in large geospatial enterprise architectures have not been fully explored yet. Hence, we recommend exploring these roles in future testbeds, e.g. serving initial or intermediate feature data sets in geoprocessing workflows or the role of Asynchronous WFSs in Big Data architectures.

9.2. Asynchronous Messaging for other OGC Service Types

The activities regarding asynchronous messaging were focusing on OGC Web Feature Services. The approach chosen relies on additional query parameters that indicate that the operation shall be executed in an asynchronous mode. The approach has been re-used and extended from Testbed 12 (described in the previous ER on Asynchronous Service Responses (OGC 16-023r3) [http://docs.opengeospatial.org/per/16-023r3.html]). Furthermore, as outlined in Section 8 additional operations may be defined to retrieve the status of the operation execution. The operations defined, e.g. GetStatus, are the same as for the OGC WPS and may also be used for other OGC services. A change request for OWS common [http://ogc.standardstracker.org/show_request.cgi?id=416] has already been submitted in the last testbed. The extensions and applications provided in this testbed should be considered when working on this change request. We therefore added a comment to the existing change request.

9.3. PubSub Extension of Web Feature Service

The approach for asynchronous messaging described in this engineering report relies on additional request parameters. However, the OGC PubSub standard also defines a general way to add asynchronous messaging to OGC Web Services. So far, a PubSub extension for WFS has not yet been explored. We, therefore, recommend to also apply the OGC PubSub standard to Web Feature Services for asynchronous delivery of features. A PubSub extension for the purpose of Geosynchronization has been defined as a result of the Geosynchronization Service activities (see OGC Geo-Synchronization Standard (OGC 17-031) [https://portal.opengeospatial.org/wiki/pub/Testbed13/ConvertDocsT13Output/T13/NG011-GeoSynchronization.html]). This may serve as a basis for a general Pub-Sub extension for WFS.

9.4. Additional Response Formats for WFS responses

As demonstrated for Asynch WFS-1 (Section 7), JSON may be a simpler and more efficient encoding than GML responses from Web Feature Services. A study of using JSON in OGC services has already been conducted in Testbed 12 (see Testbed-12 Javascript-JSON-JSON-LD Engineering Report [http://docs.opengeospatial.org/per/16-053r1.html]). Binary encodings, supported by frameworks such as Google Protocol Buffers [https://developers.google.com/protocol-buffers/] (Protobuf) or Apache Avro [https://avro.apache.org/] may even further reduce the amount of data that needs to be transferred. The usage of Protobuf and Avro for serialization of spatial data has been evaluated in Testbed 13. The results are described in OGC Testbed-13: SWAP Engineering Report (OGC 17-037) [https://docs.opengeospatial.org/per/17-037.html].

To indicate additional response formats served by a WFS, the `responseFormat` parameter can already be used in a GetFeature request. However, best practices/extensions on how to add additional encodings for responses from WFS and how to use and evaluate them for the use cases described in this ER are considered as future work.

Appendix A: Revision History

Table 1. Revision History

Date	Release	Editor	Primary clauses modified	Descriptions
May 10, 2017	.1	C. Stasch	all	initial version
May 31, 2017	.2	C.Stasch	all	IER version
August 7th, 2017	.3	C.Stasch, J. Harrison, C.-Y. Hao	all	added scenario and component descriptions from Carbon and FCU
September 13th, 2017	.4	C.Stasch,	1, 8	updated key findings and description of Asynchronous WFS-2
September 20th, 2017	.6	C. Stasch	all	updated summary and overview, removed typos throughout all sections
September 20th, 2017	.5	C.-W. Khuan	6, 8	updated scenario and description of Asynchronous WFS-2
September 22nd, 2017	.7	J. Harrison, C. Stasch	7,8,9	update of WFS-1 description, demo scenarios moved to component descriptions (Sections 7,8)
September 22nd, 2017	.8	C.-W. Khuan, C. Stasch	9	added additional response formats to future work section
October 24th, 2017	.9	C. Stasch	all	incoporated changes according to review by thread architect G. Buehler

Date	Release	Editor	Primary clauses modified	Descriptions
November 28th, 2017	.9	B. Pross, C. Stasch	all	incoporated changes according to review by G. Hobona

Appendix B: Bibliography

[1] Davies, A.: York Flood Inquiry, https://www.york.gov.uk/download/downloads/id/12456/york_flood_inquiry_main_report.pdf.