Open Geospatial Consortium, Inc.

Date: 2010-08-18

Reference number of this document: OGC 10-130

Category: OGC® Public Engineering Report

Editor: Debbie Wilson

OGC® OWS-7 Aviation – FUSE Deployment Engineering Report

Document type:	OpenGIS® Engineering Report
Document subtype:	NA
Document stage:	Approved for public release
Document language:	English

Preface

This public Engineering Report (ER) is a deliverable of the Open Geospatial Consortium (OGC) Interoperability Program Open Web Service (OWS) Testbed phase 7 (OWS-7).

The document describes the integration results of deploying an OGC Web Feature Service onto the FUSE Enterprise Service Bus. FUSE ESB is the Federal Aviation Administration service bus of choice for its System Wide Information Management (SWIM) program.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

Contents

Page

Figures

Page

OGC® OWS-7 Aviation – FUSE Deployment Engineering Report

1 Introduction

1.1 Scope

This document describes the integration results of deploying OGC Web Services on the FAA chosen Enterprise Service Bus (ESB) – FUSE. Snowflake Software were commissioned to evaluate the impacts of the FAA SWIM security requirements for both secure messaging and user authentication and gain an understanding of the requirements for deploying OGC web services into the Apache FUSE Enterprise Service Bus (ESB).

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 Inc. 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.

1.2 Document contributor contact points

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

Name	Organization
Eddie Curtis	Snowflake Software Ltd.
Hardo Mueller	Snowflake Software Ltd.
Jeremy Maskell	Snowflake Software Ltd.
Debbie Wilson	Snowflake Software Ltd.
Ian Painter	Snowflake Software Ltd.

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
08.06.2010	0.0.0	DW	All	Definition of initial document outline for contribution from authors
11.06.2010	0.1.0	DW	All	Integration of all contributions from authors and revisions post review
01.07.2010	1.0	IP	All	Final contributions for final release

1.4 Forward

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 Inc. 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.

2 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, *OpenGIS® Web Services Common Standard*

OGC 09-050r1, *OpenGIS® OWS-6-AIM Engineering Report*

OGC 10-079, *OWS-7 Aviation Architecture Engineering Report*

NOTE This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

ISO 19115(all parts), *Geographic information - Metadata*

ISO 19136:2007, *Geographic information — Geography Markup Language (GML)*

ISO/DIS 19142, *Geographic information — Web feature service*

ISO/DIS 19143, *Geographic information – Filter Encoding*

OASIS: 2007, *WS-SecurityPolicy 1.2*

W3C:2007, *Web Services Policy 1.5 – Framework, W3C Recommendation*

3 Terms and definitions

For the purposes of this report, the terms and definitions are defined in the OWS-6 Event Architecture Engineering Report (09-032) and the OWS-6 AIM Engineering Report (09-050r1).

4 Conventions

4.1 Abbreviated terms

AIM	Aeronautical information Management

AIXM	Aeronautical Information Exchange Model
API	Application Programming Interface
EAR	Enterprise Archive Resource
EFB	Electronic Flight Bag
ESB	Enterprise Service Bus
ER	Engineering Report
FAA	Federal Aviation Authority
FES	Filter Encoding Specification
GML	Geography Markup Language
HTTP	HyperText Transport Protocol
ISO	International Standardization Organization
OGC	Open Geospatial consortium
OWS	OGC Web Services
OWS-7	OWS testbed phase 7
SAML	Security Assertion Markup Language
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
STS	Security Token Service
SWIM	System Wide Information Management
WAR	Web Archive File
WFS	Web Feature Service
WFS-T	Web Feature Service –Transactional
WSDL	Web Services Description Language
XML	Extensible Markup Language

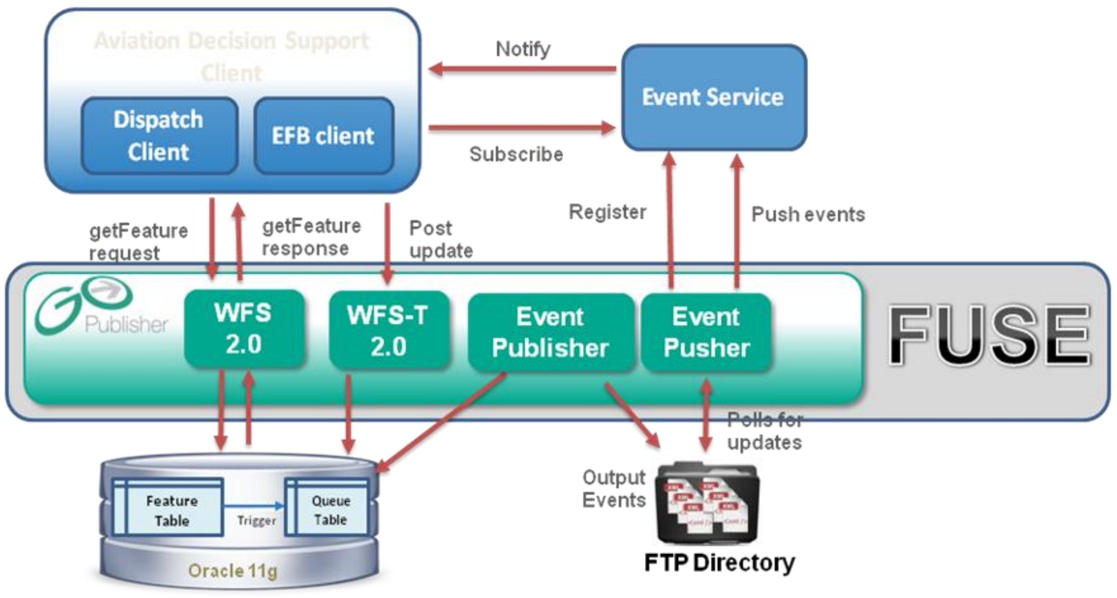
5 FUSE Integration

5.1 Overview

Chosen by FAA as its service oriented architecture (SOA) platform of choice for System Wide Information Management (SWIM), FUSE is a set of standards-based enterprise integration products based on the Apache SOA projects. FUSE includes four SOA components: FUSE ESB, FUSE Message Broker, FUSE Services Framework and FUSE Mediation Router. During OWS7-AIM Snowflake deployed its GO Publisher WFS product within the FUSE ESB using the WFS 2.0 SOAP binding.

5.2 Deployed Architecture

During OWS7 Snowflake deployed two parallel sets of the services, HTTP POST services on Apache Tomcat and SOAP services on the FUSE platform.



The full FUSE architecture includes:

- Red Hat Enterprise Linux Server
- FUSE ESB 4.2
- GO Publisher WFS 2.0
- GO Publisher Agent
- WS-Notification Event Pusher
- Oracle 11g Database

The server used to host the technology was a 64bit Intel powered system running Red Hat Enterprise Linux version 5.2. Running on this server was FUSE ESB 4.2 which is used to deploy the GO Publisher WFS, GO Publisher Agent and WS-Notification Event

Pusher. The final piece in the architecture is an Oracle database used to store the AIXM data.

5.3 OSGi

OSGi facilitates the componentization of software modules and applications and assures interoperability of applications and services. OSGi is a standard, established by the OSGi Alliance enabling different software components to collaborate in a friendly way. Originally designed for handheld devices, OSGi concepts of modularity and componentization have spread into the middle tier and are now extensively used as an alternative to traditional Enterprise Application Server deployments. Within the OSGi framework, applications or components (in the form of bundles for deployment) can be remotely installed, started, stopped, updated and uninstalled without requiring a reboot and management of Java packages/classes is specified in great detail. Life cycle management is done via APIs which allow for remote downloading of management policies. The OSGi service registry then allows bundles to detect the addition of new services, or the removal of services, and adapt accordingly.

A bundle is a key concept within OSGi, a bundle is a group of Java classes and additional resources equipped with a detailed manifest file on all its contents, as well as additional services needed to give the included group of Java classes more sophisticated behaviors, to the extent of deeming the entire aggregate a component.

5.4 Deployment of GO Publisher WFS 2.0 onto FUSE ESB

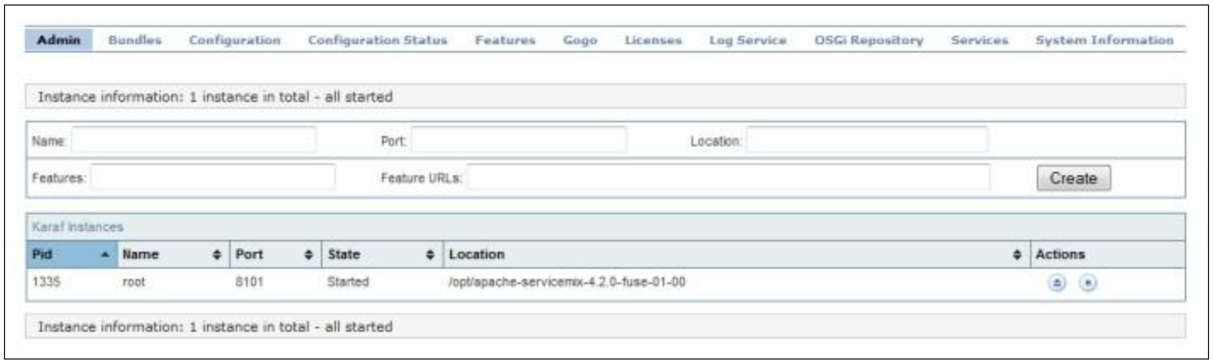
Unlike other Application Servers which make use of Enterprise Archive Resource (EAR) or Web Archive Resource (WAR) containers, FUSE ESB utilizes OSGi Bundles to deploy applications. These OSGi bundles are fundamentally different to EAR or WAR containers and posed a significant challenge for deploying GO Publisher WFS.

FUSE uses the OSGi framework to manage a set of resources (bundles) and the dependencies between them. In a typical OSGi deployment, applications use resources managed by the framework to perform a service or operation. This concept of different components is reinforced by the fact that as default Java Servlet/JSP based functionality is turned off in a basic installation of the FUSE technology stack.

GO Publisher WFS is a commercial off the shelf (COTS) software product built to run on the current generation of J2EE compliant web containers (JBoss, Tomcat, WebLogic etc). Therefore, its architecture reflects the requirements imposed upon it by this technology. GO Publisher Desktop, the configuration tool for GO Publisher WFS, creates a Web Archive file (WAR) which is deployed to a J2EE container as a complete application including the packages and libraries on which it depends.

Our challenge was to take the existing J2EE based solution and turn it into a bundle suitable for an OSGi container. The initial route decided on after investigating the functionality provided by the FUSE technology stack was to activate the FUSE web bundles and see if the standard GO Publisher WAR file could be deployed directly to the FUSE OSGi container.

FUSE was configured to allow access through an HTTP port and the relevant web/war features were installed and switched on within the technology stack.



Instance information: 1 instance in total - all started

Name:	Port:	Location:
Features:	Feature URLs:	

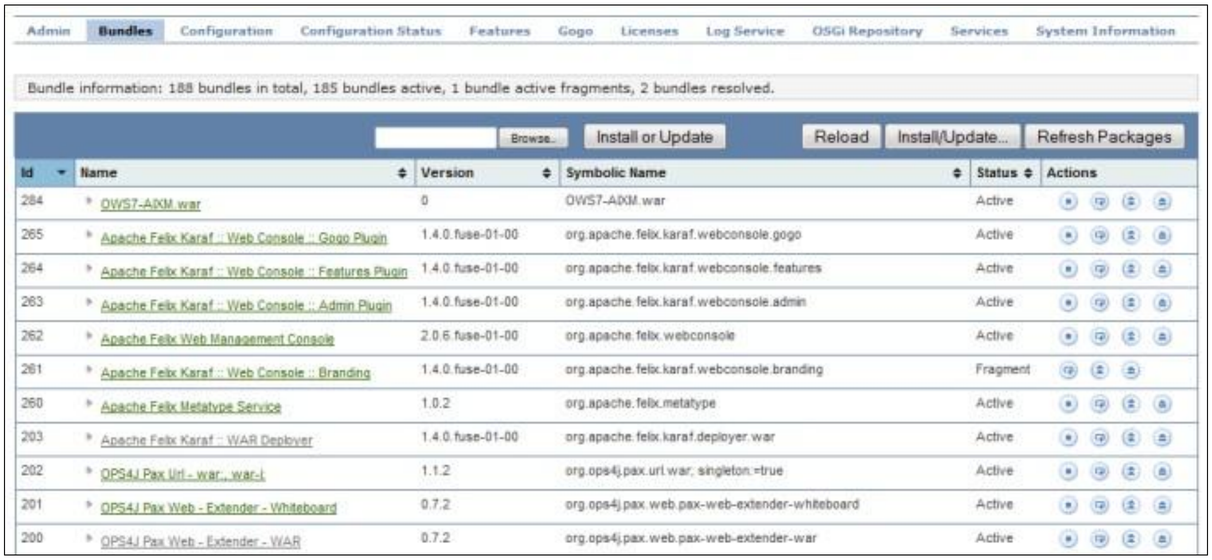
Karaf Instances

Pid	Name	Port	State	Location	Actions
1335	root	8101	Started	/opt/apache-servicemix-4.2.0-fuse-01-00	

Instance information: 1 instance in total - all started

Figure 1. FUSE up and running as viewed from the web console.

Once the stack was up and running deployment was made by converting the WAR file using the BND tool within FUSE into an OSGi bundle. For the bundle conversion process to be successful the GO Publisher WFS product code was changed to amend to the WAR deployment functionality. To create an OSGi bundle, libraries already available in the OSGi framework needed to be removed from the GO Publisher WFS WAR deployment. Use of those libraries was deferred to FUSE instead of using version included in the WAR file to make better use of the FUSE technology stack. This also had the benefit of reducing the incidence of library conflicts on the Java classpath when the WFS was running.



Bundle information: 188 bundles in total, 185 bundles active, 1 bundle active fragments, 2 bundles resolved.

Id	Name	Version	Symbolic Name	Status	Actions
284	OWS7-AIXM.war	0	OWS7-AIXM.war	Active	
265	Apache Felix Karaf :: Web Console :: Gogo Plugin	1.4.0.fuse-01-00	org.apache.felix.karaf.webconsole.gogo	Active	
264	Apache Felix Karaf :: Web Console :: Features Plugin	1.4.0.fuse-01-00	org.apache.felix.karaf.webconsole.features	Active	
263	Apache Felix Karaf :: Web Console :: Admin Plugin	1.4.0.fuse-01-00	org.apache.felix.karaf.webconsole.admin	Active	
262	Apache Felix Web Management Console	2.0.6.fuse-01-00	org.apache.felix.webconsole	Active	
261	Apache Felix Karaf :: Web Console :: Branding	1.4.0.fuse-01-00	org.apache.felix.karaf.webconsole.branding	Fragment	
260	Apache Felix Metatype Service	1.0.2	org.apache.felix.metatype	Active	
203	Apache Felix Karaf :: WAR Deployer	1.4.0.fuse-01-00	org.apache.felix.karaf.deployer.war	Active	
202	OPS4J Pax Url - war:, war-i:	1.1.2	org.ops4j.pax.url.war; singleton:=true	Active	
201	OPS4J Pax Web - Extender - Whiteboard	0.7.2	org.ops4j.pax.web.pax-web-extender-whiteboard	Active	
200	OPS4J Pax Web - Extender - WAR	0.7.2	org.ops4j.pax.web.pax-web-extender-war	Active	

Figure 2. GO Publisher WFS deployed as a OSGi bundle (Id 284)

Once GO Publisher WFS was added as a bundle it was a simple process to start the bundle and make the service available. A lot of the configuration was handled by the bundle to WAR conversion process which populated the bundles manifest file with most of the correct settings necessary to expose the WFS service. The conversion process does allow you to tailor the final bundle configuration should that be required.

From a technical point of view there was minimal change to the product itself in order to get it working within the FUSE environment, the issues mostly arose from trying to understand and select from the various different deployment options that are available in FUSE to identify the one that fits GO Publisher WFS.

5.5 Security

5.5.1 Advertising security in WSDL

To integrate with the security aspects of the FUSE platform a security policy statement needed to be inserted into the WSDL to communicate to potential clients that a service requires certain elements (SAML 1.1) to be present in a SOAP message. To fulfill this requirement a WS-SecurityPolicy statement was added to the WFS 2.0 WSDL which specifies the SAML 1.1 Token Profile as the basis for principal authentication and authorization.

The security policy statement is advertised by a wsp:Policy expression of the wsdl:definitions element and is imported into the entry point WSDL document. The wsp:Policy expression complies to the Web Services Policy 1.5 – Framework, W3C Recommendation 04 September 2007 (http://www.w3.org/TR/2007/REC-ws-policy-20070904) . This specification defines a normal form for policy expressions that is a straightforward XML Infoset representation of a policy, enumerating each of its alternatives that in turn enumerate each of their assertions.

The wsp:Policy expression contains a sp:SamlToken assertion, which represents a requirement for a SAML token. The sp:SamlToken assertion complies to the WS-SecurityPolicy 1.2, OASIS Standard, 1 July 2007 (http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html). The contents of the sp:SamlToken include the issuers name and another wsp:Policy expression, which specifies the Version of the SAML Token profile.

The following code sample shows the WSDL advertising the SAML 1.1 Token Profile:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<wsdl:definitions
        targetNamespace="http://www.opengis.net/wfs/soap/2.0"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:wsp="http://www.w3.org/ns/ws-policy"
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
    <wsdl:documentation xmlns:dc="http://purl.org/dc/elements/1.1/">
        <dc:date>2010-05-26</dc:date>
        <dc:description>
            Adds a WS-SecurityPolicy statement to the WFS WSDL
        </dc:description>
    </wsdl:documentation>
    <wsp:Policy>
        <sp:SamlToken>
            <sp:IssuerName>SampleIssuer</sp:IssuerName>
            <wsp:Policy>
                <sp:WssSamlV11Token11/>
            </wsp:Policy>
        </sp:SamlToken>
    </wsp:Policy>
</wsdl:definitions>
```

5.5.2 Enhancing GO Publisher WFS to apply security validation rules

The objective of this approach is to configure authentication so that only authenticated clients can connect to the WFS by following the security validation rules. The Specification of the Web Service Security by the U.S. Department of Transportation describes several Web Service Integration Scenarios for various situations. To decide, which scenario fits best to the WFS integration further investigation was required.

Following a dialog with other project partners, we assumed that the validation rules defined in sections 3.3.9 and 3.3.10 of the specification apply to the WFS integration. To realize this, the implementation has to consider the following profiles and specifications:

- *SWIM WS-Security BinarySecurityToken Profile*
- *SAML 1.1 Token Profile 1.1*

6 Accomplishments

- A WFS 2.0 was deployed into the FUSE ESB 1.2

- The WFS 2.0 SOAP binding was added to the WFS and tested using SOAPUI

- The WFS 2.0 WSDL was harvested by clients, validated and tested using the SOAPUI

- The WFS 2.0 WSDL was extended to support SAML Token and tested using SOAPUI

7 Issues and Next Steps

7.1 Deployment onto FUSE ESB using OSGi

OWS7-AIM demonstrated that an OGC WFS 2.0 serving AIXM 5.1 could be deployed FUSE environment. However, using the BND tool to transform the WAR file into an OSGi bundle does not necessarily take full advantage of the facilities offered by the FUSE platform. For future implementation, the possibility of exploiting more of the functionality offered by the FUSE environment should be investigated. Creation of an option to deploy GO Publisher directly as an OSGi bundle instead of as a WAR file should allow the WFS to more fully utilize the FUSE framework.

7.2 Tightly coupled SOAP Binding to WFS 2.0 Stored Queries

The OGC WFS 2.0 deployed onto the FUSE platform was exposed using the standard WSDL document as defined in the OGC WFS 2.0 standard. The WSDL describes a loose SOAP binding of the WFS 2.0 implementation, where any valid form of fes:Filter encoding can be placed in the <soap:Body> and the response can be formed of any available feature type defined within the schema. For SOA orchestration and automated code bindings, it is a considerable overhead to call and / or consume such a loosely coupled service. Furthermore, without any limitation on the types of filter being called the service is open to exploitation or erroneous queries which may flood the back end datastore. With the advent of Stored Queries in WFS 2.0 it could be possible to deploy stored queries as tightly bound web services enabling the request to be tightly bound to the Stored Query parameters and the Response to be bound to only specific Feature Type returned from the filter embedded in the stored query. This should be investigated as a future work item as it could considerably simplify service orchestration and re-use as OSGi controlled OSGi bundles.

7.3 Security

Unfortunately during OWS-7-AIM, a Security Token Service (STS) was not available to provide Security Tokens for testing. As a result no practical security testing on FUSE took place during the testbed. That said the SAML token extensions to the WFS 2.0 WSDL were implemented, tested and validated using the SOAPUI testing tool

For extensive security testing, we suggest defining test requests for the following scenarios:

- Sample WSDL security assertions, which are returned by the WFS to advertise the security requirements.
- Sample SOAP GetCapabilities request, which contains SAML 1.1 assertions in its header part to authenticate the client at the server successfully.
- Sample SOAP GetCapabilities requests, which contain SAML 1.1 assertions in its header part to try authenticating the client, but do not succeed.
- Sample SOAP GetCapabilities requests, which contain BinarySecurityTokens in its header part try authenticating the client, but do not succeed.

- Similar SOAP requests for DescribeFeatureType, GetFeature requests, etc. if their security assertions differ from the GetCapabilities request.
- Sample SOAP return message for an successful request
- Sample SOAP return message for an unsuccessful request
- Sample request to a Security Token Service (STS) to verify a token.
- Sample responses from a Security Token Service (STS) for successful and unsuccessful verifications.