Testbed-12 Web Feature Service Synchronization

Table of Contents

Publication Date: 2017-03-09

Approval Date: 2017-02-20

Posted Date: 2016-10-31

Reference number of this document: OGC 16-044

Reference URL for this document: http://www.opengis.net/doc/PER/t12-A011-3

Category: Public Engineering Report

Editor: Panagiotis (Peter) A. Vretanos

Title: Testbed-12 Web Feature Service Synchronization

OGC® Engineering Report

COPYRIGHT

WARNING

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.

Abstract

This engineering report describes a protocol for synchronizing data between two enterprise servers. While the protocol itself is generic, this engineering report describes its application to web feature servers.

In the simplest terms, the protocol involves each synchronization peer accessing the other's "Sync" resource to get the set of changed objects since the last time the "Sync" resource was accessed. In the case of web feature servers, the objects are features. The requesting peer then compare that list of changed features with the identically identified features in its data store and performs any necessary changes so that the feature states match.

Continuing the work done in Testbed-11, this engineering report describes the implementation of a Sync operation in a WFS server that:

1. Enhances the Sync operation from Testbed-11 to include an abstract query element where each service type can then substitute their specific query syntax for identifying the specific sub-set of changed features to be synchronized. In the case of the WFS, several query syntaxes may be used including the wfs:Query element and a REST based feature type URI with query parameters.
2. Extends the definition of the Sync operation with the addition of a "resultType" parameter to allow a client to obtain a hit count of the number of features that a Sync operation shall return.
3. Shall investigate the proper procedure for handling resource references. Implementing the resolvePath parameter alone is not sufficient to ensure complete data set synchronization.
4. Shall investigate concurrency and consistency issues.

NOTE	With regard to the resolvePath parameter; in order to be able to test the use of this parameter, the feature encoding needs to support the ability to reference resources (i.e. property values, other features, etc.). Although Testbed-11 used GeoJSON, this format is not sufficiently mature in this regard and so we propose using GML as the feature encoding format for this round of sync testing.

Business Value

Data synchronization is a fundamental means by which data consistency is

achieved and maintained over time between a source and target data system. It is fundamental to a wide variety of applications including, for example, file synchronization and synchronization between mobile devices and especially in a low-connectivity environment such as might exist after a natural disaster. Anyone who has ever synchronized their iPod/iPhone with iTunes has experienced data synchronization in action.

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

With regard to the WFS/FES SWG, this engineering report defines the means by which peer-to-peer data synchronization can be performed with web feature services. This can be synchronization between two web feature services or synchronization between a web feature service and some other system (e.g. a web processing service that manages geopackages).

With regard to the GeoSynchronization 1.0 SWG, the current draft GSS standard defines a publish-subscribe architecture for synchronization between services with the GeoSynchronization service acting as the broker. This engineering report describes and alternative synchronization architecture that should be considered by the GSS SWG.

Although the testing platform for synchronization for Testbed-12 is the web feature service, the protocol described in this engineering report is applicable to a number of OGC® standards and so there is a strong likelihood that some of the elements of the description should end up in the OWS Common standard.

For the OGC® in general, this engineering report defines a generic synchronization protocol that can be used to maintain data consistency across a variety of systems ranging from the very small (e.g. mobile devices with geopackages) to the very large (e.g. enterprise level data services such as web feature services).

How does this ER relate to the work of the Working Group

Although the protocol described in this engineering report is generic, the web feature service was used as the testing platform in Testbed 12. As such, this engineering report describes modification and extensions to the web feature service standard that need to be considered by the WFS/FES SWG.

As mentioned above, one part of the GSS draft standard describes a publish-subscribe architecture for synchronization. The work described in this engineering report is directly related to synchronization and is thus directly

related to the work of the GSS SWG.

Although the WFS was used as a testing platform, the synchronization protocol described in this engineering report is generic and could be applicable to a variety of OGC® standards. As such, it is related to one of the primary purposes of the OWS Common 1.2 SWG which is to collect common elements among OGC® standards (e.g. BBOX encoding) into a single standard where this information can be referenced rather than be copied over and over.

Keywords

ogcdocs, testbed-12, synchronization, web feature service, WFS, filter, change set, low connectivity, checkpoint, service identifier, REST, XML, KVP

Proposed OGC® Working Group for Review and Approval

This engineering report shall be submitted to the WFS/FES SWG, the GeoSynchronization 1.0 SWG and the OWS Common 1.2 SWG for review and comment.

Chapter 1. Introduction

1.1. Scope

This engineering report defines a generic synchronization protocol that is applicable to a variety of OGC® services.

This engineering report specifically describes the implementation of this synchronization protocol for web feature services.

This engineering report defines a Sync operation for services that implement one or more of a KVP-GET, XML-POST or SOAP request pattern(s).

This engineering report defines a sync resource for services that implement a REST architectural style.

This engineering report defines the XML encoding of the messages exchanged between synchronization peers.

1.2. Document contributor contact points

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

Table 1. Contacts

Name	Organization
Jeff Harrison	The Carbon Project
Mark Mattson	The Carbon Project
Panagiotis (Peter) A. Vretanos	CubeWerx Inc.

1.3. Change requests

No change requests where posted as a result of the work described in this ER.

1.4. Future Work

1. For the WFS, define a JSON encoding for the messages exchanged between synchronizing peers
2. Test the sync protocol for transactional data services other than WFS
3. Test the sync protocol between GeoPackages
4. Investigate, compare and recommend conflict resolution strategies

1.5. 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 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 06-121r9, OGC® Web Services Common Standard
- OGC 14-102, OGC® Web Feature Service 2.5 Interface Standard
- OGC 14-103, OGC® Filter Encoding 2.5 Encoding Standard
- OGC 15-011r1, OWS-11 Case Study of Multiple WFS-T Interoperability ER

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] shall apply. In addition, the following terms and definitions apply.

3.1. attribute <XML>

name-value pair contained in an element <XML> (see ISO 19136:2007, definition 4.1.3)

NOTE	In this document an attribute is an XML attribute unless otherwise specified.

3.2. client

software component that can invoke an operation from a server (see ISO 19128:2005, definition 4.1)

3.3. element <XML>

basic information item of an XML document containing child elements, attributes and character data (see ISO 19136:2007, definition 4.1.23)

3.4. feature

abstraction of real world phenomena (see ISO 19101:2002, definition 4.11)

NOTE	A feature can occur as a type or an instance. The term "feature type" or "feature instance" should be used when only one is meant.

3.5. feature identifier

identifier that uniquely designates a feature instance

3.6. filter expression

predicate expression encoded using XML (see OGC 14-103, definition 4.11)

3.7. interface

named set of operations that characterize the behaviour of an entity (see ISO 19119:2005, definition 4.2)

3.8. Multipurpose Internet Mail Extensions (MIME) type

media type and subtype of data in the body of a message that designates the native representation (canonical form) of such data (see IETF RFC 2045:1996)

3.9. namespace <XML>

collection of names, identified by a URI reference which are used in XML documents as element <XML> names and attribute <XML> names (see W3C XML Namespaces:1999)

3.10. operation

specification of a transformation or query that an object may be called to execute (see ISO 19119:2005, definition 4.3)

3.11. property

facet or attribute of an object, referenced by a name (see OGC 14-103, definition 4.21)

3.12. resource

asset or means that fulfills a requirement (see ISO 19115:2003, definition 4.10)

NOTE	In this International Standard, the resource is a feature, or any identifiable component of a feature (e.g. a property of a feature)

3.13. remote resource

resource that is not under direct control of a system

NOTE	In this International Standard, the system is a web feature service. The resource is not held in any data store that is directly controlled by that service and thus cannot be directly retrieved by the service.

3.14. request

invocation of an operation by a client (see ISO 19128:2005, definition 4.10)

3.15. response

result of an operation returned from a server to a client (see ISO 19128:2005, definition 4.11)

3.16. schema

formal description of a model (see ISO 19101:2002, definition 4.25)

NOTE In general, a schema is an abstract representation of an object's characteristics and relations to other objects. An XML schema represents the relationship between the attributes and elements of an XML object (for example, a document or a portion of a document).

3.17. schema <XML Schema>

collection of schema components within the same target namespace <XML> (see ISO 19136:2007, definition 4.1.54)

3.18. server

particular instance of a service (see ISO 19128:2005, definition 4.12)

3.19. service

distinct part of the functionality that is provided by an entity through interfaces (see ISO 19119:2005, definition 4.1)

3.20. service metadata | capabilities document

metadata describing the [operations] and geographic information available at a server (see ISO 19128:2005, definition 4.14)

3.21. Uniform Resource Identifier

unique identifier for a resource, structured in conformance with IETF RFC 2396 (see ISO 19136:2007, definition 4.1.65)

NOTE The general syntax is <scheme>::<scheme-specified-part>. The hierarchical syntax with a namespace <XML> is: <scheme>://<authority><path>?<query> Conventions

Chapter 4. Abbreviated terms

Table 2. Abbreviations

API	Application Program Interface
COTS	Commercial Off The Shelf
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
IDL	Interface Definition Language
CGI	Common Gateway Interface
CRS	Coordinate Reference System
DCP	Distributed Computing Platform
EPSG	European Petroleum Survey Group
FES	Filter Encoding Specification
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
KVP	Keyword-value pairs
MIME	Multipurpose Internet Mail Extensions
OGC®	Open Geospatial Consortium
OWS	OGC® Web Service
SQL	Structured Query Language
SOAP	Simple Object Access Protocol
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
VSP	Vendor Specific Parameter
WFS	Web Feature Service
WSDL	Web Services Description Language
XML	Extensible Markup Language

Chapter 5. Overview

The synchronization work described in this engineering report is a continuation of work that was begun in Testbed 11 and described in the document OGC 15-011r1, OWS-11 Reference Case Study of Multiple WFS-T Interoperability ER.

Although the synchronization protocol describe in this engineering report is generic, the primary concern of this document is describing synchronization between web feature services.

The synchronization protocol is based on the concept of a change set. A change set is a list of data objects that have changed, from some specified checkpoint, that is exchanged between synchronization peers.

Clause Components describe the components used in Testbed 12 as the test platform for synchronization.

Clause Synchronizing features describes how features listed in a changes set should be processed by a requesting peer.

Clause Accessing the change set defines the operations, resources and messages that are necessary of accessing the change set of a server.

Chapter 6. New Requirements Statement

6.1. Status Quo

The current synchronization protocol is defined in document OGC 15-010r4, OGC® Testbed-11 WFS-T Information Exchange Architecture. As defined in OGC 15-010r4, the protocol allows synchronizing peers to request each other's change sets since some specified checkpoint on a per-feature-type basis.

6.2. Requirements Statement

There are several issues with the synchronization protocol as defined in OGC 15-010r4:

First, there is no way for a requesting peer to determine the size of the change set that a responding peer would generate in response to a sync request. Thus, a means is required for a requesting peer to be able to request the size of the changes set since some specified checkpoint. This is akin to the resultType parameter defined for WFS GetFeature request (see OGC 14-102, 7.6.3.6).

Second, if a requesting peer is only interested in a subset of features defined by some filter, there is no means for the requesting peer to indicate that the responding peer prune the change set based on that filter.

Finally, there is no means currently defined for a requesting peer to request that the responding peer resolve references that might exists in the features in the change set. Thus a means is required for a requesting peer to be able to specify that the responding peer resolve references. This is akin to the resolve parameters defined for the WFS GetFeature request (see OGC 14-102, 7.6.4).

Chapter 7. Peer-to-peer enterprise web feature server synchronization

7.1. Introduction

This clause describes the peer-to-peer synchronization protocol implemented to satisfy the enterprise-to-enterprise data synchronization use case for OGC® Testbed 12.

7.2. Components

Figure 1 illustrates the components involved in the OGC® Testbed 12 enterprise-to-enterprise synchronization experiment.

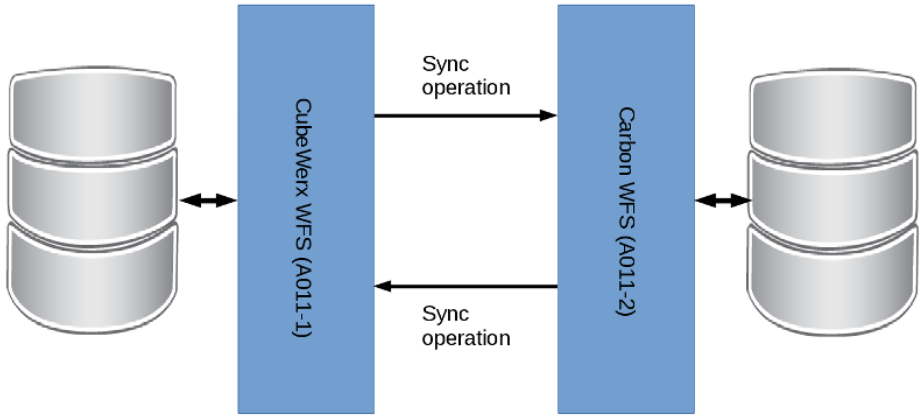

Figure 1. Component diagram for Synchronization experiment

The experiment was performed between two WFSs; the Carbon WFS and the CubeWerx WFS. *WFS product versions and end points* lists the product versions and endpoint of the servers.

Table 3. WFS product versions and end points

Vendor	Product	Supported WFS Versions	Endpoints
CubeWerx	CubeWerx Suite 8.1.1	1.0.0 +	https://tb12.cubewerx.com/a011/cubeserv/default/wfs/2.0.2/sync
Carbon	Carbon Cloud WFS	1.0.0 +	http://ows12.azurewebsites.net/wfs

Both servers implement the WFS standard with the REST binding (see OGC 11-080r1). Furthermore, each server implements the newly specified Sync resource (see 8.3) which allows the servers to synchronize the state of their features.

Server capabilities lists the specific capabilities of each server used in the experiment. Cross-walking the capabilities resulted in the experiment being performed using features encoded with GML. In other words, when the server exchange features through the "Sync" resource, those

features are encoded using GML.

Table 4. Server capabilities

Property	**CubeWerx**	**Carbon**
Versions	1.0.0 +	1.0.0 +
Operations	GetCapabilities DescribeFeatureType GetFeature ListStoredQueries DescribeStoredQueries GetPropertyValue Transaction Sync GET PUT POST DELETE	GetCapabilities DescribeFeatureType GetFeature Sync Transaction GET PUT POST DELETE
Output Formats	GML v3.2 GML v3.1.1 GML v2.1.2 GeoJSON KML SHAPE ATOM RSS HTML	GML v3.2 GML v3.1.1 GML v2.1.2 GeoJSON KML CSV TopoJSON
Spatial operators	Disjoint Equals Intersects Touches Crosses Contains Overlaps BBOX Within	Disjoint Equals Intersects Touches Crosses Contains Overlaps BBOX Within Beyond
Spatial operands	gml:Envelope gml:Point gml:LineString gml:Polygon gml:CircleByCenterPoint	gml:Envelope gml:Point gml:LineString gml:Curve gml:Polygon gml:Surface gml:MultiPoint gml:MultiLineString gml:MultiCurve gml:MultiPolygon gml:MultiSurface

Property	CubeWerx	Carbon
Scalar operators	PropertyIsBetween PropertyIsEqualTo PropertyIsGreaterThan PropertyIsGreaterThanOrEqualTo PropertyIsLessThan PropertyIsLessThanOrEqualTo PropertyIsLike PropertyIsNotEqualTo PropertyIsNull	PropertyIsBetween PropertyIsEqualTo PropertyIsGreaterThan PropertyIsGreaterThanOrEqualTo PropertyIsLessThan PropertyIsLessThanOrEqualTo PropertyIsLike PropertyIsNotEqualTo
Logical	And, Or, Not	And, Or
Available Stored Queries	GetFeatureById NearestNeighbours	None
Number of CRSs	> 10	> 10
Support for GML SF	Yes	Yes
REST API	Yes	Yes
GeoJSON	Yes	Yes
ATOM	Yes	No
XSLT vendor extension	Yes	No

7.3. Synchronizing features

7.3.1. Introduction

The synchronization protocol assumes that each peer can, upon request, identify for the other peer which data has changed since a previous checkpoint. In the case of web feature services, this means that a responding WFS can identify which features have changed for a specified feature type since a previous checkpoint. This set of changed data, or features in the case of a WFS, is called the *change set*.

The process of synchronization involves one peer, the requesting or client peer, requesting from the other peer, the responding or server peer, a change set from some specified checkpoint.

This engineering report describes an XML encoding for the change set that includes a standard wfs:FeatureCollection, for listing all newly created and modified existing features, and a list of feature identifiers that lists the features that have been deleted from the server.

The specific focus of Testbed 12 is the web feature service, however the protocol is sufficiently generic to work between any two peers as long as the encoding for the change set is defined. For example, this protocol could be used to sync geopackages (see GeoPackage Mobile Apps Integrations, OGC 16-030) in the field where the change set itself is encoded as a geopackage.

7.3.2. Change set processing

Introduction

This clause briefly describes how inserted, updated and/or deleted features appearing in a change set should be processed by a requesting peer.

Inserting a new feature

If a requesting peer encounters a feature in the change set that does not currently exist in its repository, then the requesting peer shall create an identical new feature in its repository.

The new feature shall be retrievable using the identifier provided by the responding peer.

NOTE	In general, when creating a new feature, a web feature service may assign its own identifier to a feature. This is perfectly fine as long as the server maintains the relationship between its own identifier and the identifiers provided by its synchronizing partners. In other words, a feature in a server that supports synchronization may be retrievable using any number of gml:id values in addition to the one that a server natively assigns to the feature.

7.3.3. Updating an existing feature

If a requesting peer encounters a feature in the change set that already exists in its repository then it shall update the state of its copy to match that for the responding peer.

Deleting features

If a requesting peer encounters a feature in the change set that has been deleted then it shall delete that feature from its repository. The question of how deleted features are represented in the changeset is discussed in the *synchronization response* clause.

7.3.4. Synchronization protocol

Introduction

Servers that implement one or more of the KVP-GET, XML-POST or SOAP bindings shall implement the Sync operation that may be used to retrieve the change set. This operation shall be advertised in the server's capabilities document in the standard way.

Servers implementing the REST architectural style shall provide a */sync* resource to which the appropriate query parameters may be appended in order to retrieve the change set.

7.3.5. No prior synchronization between the peers

This clause describes the messages exchanged between a requesting and responding peer when there has been no prior synchronization between the peers. For the purpose of discussion the requesting peer is called *alpha* and the responding peer is called *beta*. The synchronization proceeds as follows:

- Alpha requests a change set from beta.
- Since alpha and beta have never synchronized before, beta generates a change set for alpha that contains all the features of the specified type.
    - The canonical format of the response shall be GML 3.2; other formats are allowed but they are not defined is this engineering report.
- Alpha reads the checkpoint value from the HTTP header of the response and saves it in order to retrieve the next change set.
- Alpha processes the change set and synchronizes its repository with that of beta.

For servers implementing a REST architectural style, the following example sequence diagram illustrates this exchange:

- alpha service Id = 052350f2-70ca-4201-837d-15f2af7ed15c
- beta service Id = e039611a-198a-474a-800d-763f79bf6ec5

```
alpha                                                                  beta
  |                                                                      |
  | GET /sync?TYPENAME=InWaterA_1M&SERVICEID=urn:uuid:0523...d15c  HTTP/1.1  |
  | Host: www.alpha.com                                                  |
  | Accept: application/gml+xml; version=3.2                             |
  +--------------------------------------------------------------------->|
  |                                                                      |
  |  HTTP/1.1 200 OK                                                     |
  |  Host: www.beta.com                                                  |
  |  Content-Type: application/gml+xml; version=3.2                      |
  |  OGC-SYNC-ServiceId: urn:uuid:e039611a-198a-474a-800d-763f79bf6ec5   |
  |  OGC-SYNC-Checkpoint: urn:uuid:4dc01e43-8cf8-4d8b-875e-0ee110eb2a75  |
  |                                                                      |
  |  <?xml version="1.0"?>                                               |
  |  <wfs:ChangeSet                                                      |
  |     serviceId="urn:uuid:e039611a-198a-474a-800d-763f79bf6ec5"        |
  |     checkPoint="urn:uuid:4dc01e43-8cf8-4d8b-875e-0ee110eb2a75">      |
  |     <wfs:FeatureCollection timestamp="2016-04-08T12:26:25"           |
  |                            numberMatched="525"                       |
  |                            numberReturned="525" ... />               |
  |        .                                                             |
  |        .                                                             |
  |        .                                                             |
  |        <wfs:member>                                                  |
  |           ...                                                        |
  |           <InWaterA_1M gml:id="1013">                                |
  |              ...                                                     |
  |           </InWaterA_1M>                                             |
  |           ...                                                        |
  |        </wfs:member>                                                 |
  |        .                                                             |
  |        .                                                             |
  |        .                                                             |
  |     </wfs:FeatureCollection>                                         |
  |     <wfs:DeletedObjects>                                             |
  |        <fes:ResourceId fid="1013"/>                                  |
  |     </wfs:DeletedObjects>                                            |
  |  </wfs:ChangeSet>                                                    |
  |<---------------------------------------------------------------------+
```

NOTE

Practically speaking this initial synchronization scenario is only feasible if beta does not already contain too great a volume of data. In this more likely scenario, one possible approach would be to export beta's data and define a checkpoint that signifies this point in time and is distributed with the export. Alpha can then be seeded with beta's data using some other, more rapid, means and normal peer-to-peer synchronization can proceed from then on using the accompanying checkpoint value.

NOTE	A server that implement the sync operation must do two things: (a) it must keep track of changes changes made to all the features that it offers and (b) for each requester, it must keep track of the checkpoints in order to know up to which point each requester has been sync'ed. Requesters are identified using the *serviceId* parameter and the *checkpoints* are pointers into the tracked changes list indicating that the requester is synced to that point

7.3.6. Subsequent synchronization between peers

In the case where the participating peers alpha and beta have previously synchronized, the protocol proceeds as follows.

- Alpha requests a change set from beta using the checkpoint from the previous sync..
- The response is a wfs:FeatureCollection document containing the features that have changed since the specified checkpoint value.

For servers implementing a REST architectural style, the following example sequence diagram illustrates this exchange:

```
alpha                                                                  beta
  |                                                                      |
  |                                                                      |
  | GET /sync?TYPENAME=InWaterA_1M                                       |
  |     &SERVICEID=urn:uuid:052350f2-70ca-4201-837d-15f2af7ed15c         |
  |     &CHECKPOINT=urn:uuid:4dc01e43-8cf8-4d8b-875e-0ee110eb2a75  HTTP/1.1  |
  | Host: www.alpha.com                                                  |
  | Accept: application/gml+xml; version=3.2                             |
  +--------------------------------------------------------------------->|
  |                                                                      |
  |  HTTP/1.1 200 OK                                                     |
  |  Host: www.beta.com                                                  |
  |  Content-Type: application/gml+xml; version=3.2                      |
  |  OGC-SYNC-ServiceId: urn:uuid:e039611a-198a-474a-800d-763f79bf6ec5   |
  |  OGC-SYNC-Checkpoint: urn:uuid:a4e9819b-2139-47cf-b297-7656a9da0a00  |
  |                                                                      |
  |  <?xml version="1.0"?>                                               |
  |  <wfs:ChangeSet                                                      |
  |     serviceId="urn:uuid:e039611a-198a-474a-800d-763f79bf6ec5"        |
  |     checkPoint="urn:uuid:a4e9819b-2139-47cf-b297-7656a9da0a00">      |
  |     <wfs:FeatureCollection timestamp="2016-04-10T12:12:57"           |
  |                            numberMatched="5"                         |
  |                            numberReturned="5" ... />                 |
  |        .                                                             |
  |        .                                                             |
  |        .                                                             |
  |        <wfs:member>                                                  |
  |           <InWaterA_1M gml:id="3764">...</InWaterA_1M>               |
  |        </wfs:member>                                                 |
  |        <wfs:member>                                                  |
  |           <InWaterA_1M gml:id="1154">...</InWaterA_1M>               |
  |        </wfs:member>                                                 |
  |        <wfs:member>                                                  |
  |           <InWaterA_1M gml:id="72">...</InWaterA_1M>                 |
  |        </wfs:member>                                                 |
  |        .                                                             |
  |        .                                                             |
  |        .                                                             |
  |     </wfs:FeatureCollection>                                         |
  |     <wfs:DeletedObjects>                                             |
  |        <fes:ResourceId fid="101"/>                                   |
  |        <fes:ResourceId fid="47625"/>                                 |
  |        <fes:ResourceId fid="1783"/>                                  |
  |        <fes:ResourceId fid="5534"/>                                  |
  |        <fes:ResourceId fid="3"/>                                     |
  |    </wfs:DeletedObjects>                                             |
  |  </wfs:ChangeSet>                                                    |
  |<---------------------------------------------------------------------+
```

NOTE Alpha and beta can reverse roles in order to achieve bi-directional synchronization.

7.3.7. Conflict resolution

Conflicts arise when the same feature property is updated with different values on synchronizing peers during the inter-sync period. For example, consider the following sequence of events:

```
        PEER 1                      PEER 2
T1      A=10                        A=10
T2      A=20                        A=30
T3      A=30  <------ SYNC -------- A=30
T4      A=30  ------- SYNC -------> A=30
```

At T1, peer 1 and peer 2 are synchronized. At T2 an update occurs on each server that changes the value of A on peer 1 to 20 and the value of A on peer 2 to 30. The problem arises when the next synchronization between peer 1 and peer 2 occurs. Without intervention one of these updates will be lost depending on which peer first acts as the server peer on the next sync. Say peer 1 initiates the synchronization by executing the Sync operation on peer 2 at T3. Peer 1 will receive the value of 30 for A and update its A value to 30. When peer 2 performs the subsequent complimentary sync at T4, it will read the value 30 from peer 1 and do nothing. The fact that on peer 1 A had a value of 20 will be lost. Ideally the synchronizing server should detect this situation and then perform some sort of conflict resolution. For Testbed 12, the CubeWerx server can detect this conflict but does not currently implement any resolution strategy.

This engineering report acknowledges this problem but does not offer a specific solution since there are a number of conflict resolution strategies — with the goal of data convergence between the synchronizing peers — that would need to be implemented and tested and this work is beyond the scope of work possible with the time and resources available in TB12. This is left as a future work item.

7.4. Accessing the change set

7.4.1. Introduction

This clause defines how the change set may be retrieved by a requesting peer.

This clause defines the encoding of the Sync operation for servers that implement KVP-GET, XML-POST or SOAP encoded requests.

This clause also defines the */sync* resource for servers that implement a REST architectural style.

7.4.2. Sync operation

XML encoding

The following XML-fragment declares the XML-encoding for the Sync operation:

```
<xsd:element name="Sync" type="wfs:SyncType"/>
<xsd:complexType name="SyncType">
   <xsd:complexContent>
      <xsd:extension base="wfs:GetFeatureType">
         <xsd:attribute name="serviceId"  type="xsd:AnyURI" use="required"/>
         <xsd:attribute name="checkpoint" type="wfs:AnyURI"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The Sync operation extends the WFS GetFeature only for the purpose of inheriting syntax because the required encoding of the Sync operation looks very much like a GetFeature request. However, the semantics of the Sync operation are different than those of the GetFeature operation in that the Sync operation only operates on the restricted subset of features that are members of the change set.

The features that are members of a change set is determined independently of the Sync operation and is a time dependent function of the transaction operations executed on the server between checkpoints. This implies that any filter specified on the Sync operation, for example, has no effect on the set of features that appear in the change set of a feature type between checkpoints. The filter only has an effect at the time the change set is to be transmitted from the server to the client. Consider a client that is only interested in changes in a particular area. Without a filter, the entire change set would e transmitted to the client and the client would have to prune the change set keeping only the features of interest and discarding the rest. The filter on the Sync operation is simply a way for the client to tell the server to prune the set before transmitting it. Regardless of any filter specified on the Sync operation, the sync checkpoints remain sequential. This description applies generally to all Sync operation parameters in that their scope is restricted to the features that are members of the change set.

The following table lists the parameters of a Sync operation and describes how they operate differently (or not) than those used with a GetFeature operation:

Table 5. Sync request parameter restrictions

Parameter Name	Description
service	As described in OGC 14-102, clause 7.6.2
version	As described in OGC 14-102, clause 7.6.2
handle	As described in OGC 14-102, clause 7.6.2
startIndex	As described in OGC 14-102, clause 7.6.3 but its scope is restricted to the features in the change set
count	As described in OGC 14-102, clause 7.6.3 but its scope is restricted to the features in the change set
resultType	As described in OGC 14-102, clause 7.6.3 but its scope is restricted to the features in the change set

Parameter Name	Description
outputFormat	As defined in clause *Response*; other output formats are allowed but not defined in this document
resolve	As described in OGC 14-102, clause 7.6.4 but its scope is restricted to the features in the change set
resolveDepth	As described in OGC 14-102, clause 7.6.4 but its scope is restricted to the features in the change set
resolveTimeout	As described in OGC 14-102, clause 7.6.4 but its scope is restricted to the features in the change set
namespaces	As described in OGC 14-102, clause 7.6.6
vsp	Vendor-specific parameters are allowed for the KVP-encoded Sync request but their meaning is not described in this document
typeNames	As described in OGC 14-102, clause 7.9.2.4.1 but restricted to a single feature type
schema-element()	As described in OGC 14-102, clause 7.9.2.4.2
aliases	As described in OGC 14-102, clause 7.9.2.4.3 but restricted to a single feature type
srsName	As described in OGC 14-102, clause 7.9.2.4.4
projection clause	As described in OGC 14-102, clause 7.9.2.4.5, 7.9.2.4.6
resolution path	As described in OGC 14-102, clause 7.9.2.4.7
selection clause	As described in OGC 14-102, clause 7.9.2.5 but its scope is restricted to the feature in the changes set and joins of any kind are not supported
sorting clause	As described in OGC 14-102, clause 7.9.2.5.4 but its scope is restricted to the features in the change set
stored query expressions	Not considered for this testbed

The following conformance WFS conformance classes from OGC 14-102, related to retrieving features from the server, are incompatible with the sync operation:

- Locking WFS
- Standard joins
- Spatial joins
- Temporal joins

KVP encoding

This clause defines the KVP-encoding for the Sync operation.

The following table, _Keywords for KVP-encoded Sync operation, describes the parameters for the

KVP-encoding of the Sync operation:

Table 6. Keywords for KVP-encoded Sync operation

URL Component	O/M	Description
Common KeyWords (REQUEST=Sync)		See Table 4, OGC 14-102 for additional parameters that may be used
Standard Presentation Parameters		See Table 5, OGC 14-102
Standard Resolve Parameters		See Table 6, OGC 14-102
Additional common keywords		See Table 7, OGC 14-102
TYPENAMES	Mandatory	Name of single, feature type to synchronize (joins not supported)
SRSNAME	Optional	See 7.9.2.4.2, OGC 14-102
Projection clause	Optional	See Table 9, 14-102
Selection clause (see KVP-encoding for selection clause, Additional selection clause parameters)	Optional	The selection clause may be used to constrain the set of change set of new or modified features that are presented to a requesting peer is a response document. The complete syntax for specifying the selection clause is presented in *KVP-encoding for selection clause* and *Additional selection clause parameters* below.

NOTE All KVP parameters are subject to the restrictions described the *Sync request parameter restrictions* table.

NOTE The canonical output format for the *Sync* operation is GML 3.2 (i.e. application/gml+xml; version=3.2). Other output formats are allowed but are not described in this engineering report.

The following tables, KVP-encoding for selection clause and Additional selection clause parameters define the parameters for a selection clause.

Table 7. KVP-encoding for selection clause (see Table 3, OGC 14-103)

URL Component	O/M	Description
FILTER	O	The value of the parameter shall be a filter expression encoded using the language specified by the FILTER_LANGUAGE parameter

URL Component	O/M	Description
FILTER_LANGUAGE	O	Indicate the predicate language used to encode the filter expression that is the value of the FILTER parameter. The default value of urn:ogc:def:queryLanguage:OGC-FES:Filter shall be used to indicate that the value of the FILTER parameter is a string encoding the filter using an XML fragment as defined in OGC 14-103.

Table 8. Additional selection clause parameters (see Table 3a, OGC 14-103)

Query class	Query subclass	URL Component	OpenSearch Parameter(s)	Description	Data type and value
Text search[3]	N/A	Q	searchTerms	A space separated list of search terms that are used to search all text fields in a catalogue record.	Character string
FeatureById search	N/A	RESOURCEIDS	Uid	Comma separated list of record identifiers to retrieve.	Character string

Query class	Query subclass	URL Component	OpenSearch Parameter(s)	Description	Data type and value
Spatial search					

		M		measure used to express the value of the	string (see http://unitsofmeasure.org/ucum.html)
Query class	**Query subclass**	**URL Component**	**OpenSearch Parameter(s)**	**Description** parameter	**Data type and value**
	Proximity search[5]	LAT	lat	Latitude expressed in WGS84	Number
		LON	lon	Longitude expressed in WFS84	Number
		RADIUS	radius	Search radius expressed in meters along the surface of the Earth	Number
Temporal search	N/A	TIME	start/end	A time instance or time period	Character string (see http://www.w3.org/TR/NOTE-datetime)
		TRELATION	N/A	The temporal operator to apply using the value of the TIME parameter	- Character string; One of: **After** Before **Begins** BegunBy **TContains** During **EndedBy** Ends **TEquals**[1] Meets **MetBy** TOverlaps **OverlappedBy** **AnyInteracts**[2]

1. Default for a time instance. 2. Default of a time period. 3. Basic text searching using the "q" parameter is expected to proceed as follows:. a. Math against the full text of all character-valued properties of a feature. b. Matching is case-insensitive. c. In the case of multiple search terms, if any of the property values being tested, as per (a), contains at least one of the specified search terms then that feature shall appear in the result set. d. Exact phrases can be delimited using quotations (i.e. 0x22 in ASCII). 4. When specified, the GEOMETRY parameter of the Arbitrary Geometry Search subclass is mandatory. The other parameters are optional and if not specified on a request assume the default values denoted in the "Data type and value" column. 5. When specified, the Proximity Search parameters are all mandatory

NOTE Tables KVP-encoding for selection clause and Additional selection clause parameters are copied from OGC 14-103, OpenGIS Filter Encoding 2.5 Encoding Standard and are presented here for convenience only.

WARNING	The use of the parameters BBOX and GEOMETRY from Table 8, Additional selection clause parameters in conjunction with the SYNC operation between WFS peers is subject to patent restrictions (see US patent 7873697).

The following table, *Sync-specific parameters* lists additional parameters that shall be appended to a KVP-encoded Sync operation according to the specified obligation.

Table 9. Sync-specific parameters

URL Component	Description	Default Value	Obligation
SERVICEID	A unique identifier for the entity making the request.	None	Mandatory
CHECKPOINT	Identifies that last point of synchronization for the specified serviceId	None	Optional on first sync; Mandatory thereafter.

7.4.3. /sync resource

For a service implementing the REST architectural style, the resource */sync* is define and may be accessed by a requesting peer to obtain the change set from a responding peer.

Tables *Sync parameters*, *Additional selection clause parameters*, Sync-specific parameters define parameters that may be appended as query parameters to the */sync* resource.

7.4.4. Custom HTTP headers

The synchronization protocol defined in this engineering report defines two custom HTTP headers named *OGC-SYNC-ServiceId* and *OGC-SYNC-Checkpoint*.

The value of the OGC-SYNC-ServiceId header is used by a peer to communicate its unique service identifier when communicating with its partner peer.

The value of the OGC-SYNC-Checkpoint uniquely identifies a point in time from which a responding server shall generate a change set.

The format of both these values is opaque.

These headers shall be set by a responding peer to communicate its service identifier and the checkpoint value to a requesting peer.

The OGC-SYNC-ServiceId header shall be set by a requesting peer to communicate its service identifier to the responding peer. The OGC-SYNC-Checkpoint header shall not be set by a requesting peer on its **initial** sync but shall be set by the requesting peer on all **subsequent** sync requests.

These parameters may be set by a requesting peer to communicate its service identifier and a checkpoint value to a responding peer.

NOTE	The query parameters SERVICEID and CHECKPOINT (see Sync-specific parameters may also be used to communicate the service identifier and checkpoint values. If both the query parameters and HTTP headers are set in any communication, then the corresponding values shall match; otherwise an exception shall be generated.

7.4.5. Response

Introduction

The response to accessing the */sync* resource or executing a *Sync* operation is a change set. This clause defines the encoding for a change set.

The response document contains the collection of features that is composed of the newly created and updated features since a specified checkpoint. The response document also contains a list of feature identifiers for features that have been deleted since that same specified checkpoint.

This encoding is easily adaptable to output formats other than GML (e.g. GeoJSON). It is also more easily adaptable to synchronization in a heterogeneous system where the synchronizing peers are not both web feature services (e.g. geopackage-to-geopackage synchronization or geopackage-to-wfs synchronization). A drawback of this encoding is that it may require more processing by the requesting peer, especially if that peer is a web feature service.

XML response

The following XML-fragment defines the default synchronization response (i.e. the output format is set to application/gml+xml; version=3.2).

The response is a composed of a wfs:FeatureCollection element containing all the newly created and modified features of a responding peer from a specified checkpoint and a wfs:DeletedObjects element containing the identifiers of all deleted features of a responding peer from the same specified checkpoint.

```
<xsd:element name="ChangeSet" type="wfs:ChangeSetType"/>
<xsd:complexType name="ChangeSetType">
   <xsd:complexContent>
      <xsd:sequence>
         <xsd:element ref="wfs:FeatureCollection" minOccurs="0"/>
         <xsd:element name="DeletedObjects" minOccurs="0">
            <xsd:complexType>
               <xsd:sequence>
                 <xsd:element ref="fes:_Id"
                              minOccurs="0" maxOccurs="unbounded"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="ConflictObjects" minOccurs="0">
            <xsd:complexType>
               <xsd:sequence>
                 <xsd:element ref="fes:_Id"
                              minOccurs="0" maxOccurs="unbounded"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="serviceId" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="checkPoint" type="xsd:anyURI"/>
      <xsd:attribute name="numberOfFeatures" type="xsd:nonNegativeInteger"/>
   </xsd:complexContent>
</xsd:complexType>
```

The serviceId parameter contains the unique service identifier of the responding peer. The specific format of this identifier is not specified in this engineering report but it is strongly recommended that a UUID be used.

The checkPoint parameter contains the new, opaque checkpoint value that may be used to obtain a subsequent change set.

If the resultType parameter (see X.X) is set to "hits" when the Sync operation is executed then the server shall NOT perform a checkpoint but rather return an empty change set document (see Note below) with the value of the "numberOfFeatures" parameter set to the number of features that would be presented to the client if the checkpoint was actually performed.

NOTE If a server detects conflicts in the course of computing the number of features that the sync operation would return, those conflicts may be listed in the ConflictObject section of the response document.

In order for a responding peer to communicate to a requesting peer which features have been deleted, the synchronization response document shall contain a list of of feature identifiers (see 7.2, OGC 14-102) from the responding peer that have been deleted. These shall be encoded using the wfs:DeletedObjects element in the change set. The content of the wfs:DeletedObject element is simply an unordered list of the feature identifiers of features (see 7.2, OGC 14-102) that have been

deleted.

NOTE	(Point for discussion): Another implementation option for handling deleted features would be to allow a requesting peer to make an additional synchronization request with the same checkpoint that includes a parameter to indicate that the deleted feature identifiers should be returned rather than the complete change set. This approach is more complicated but has the added benefit that other output formats can be supported without modification.

In the situation where a synchronizing peer has detected conflicts, the server shall not synchronize the conflicted feature but shall instead list their identifiers in the ConflictObject section of the change set document.

JSON response

In the OGC® Testbed-11 experiment, the list of changed, modified or deleted features was encoded using GeoJSON which was an output format that both participating servers supported. However, GeoJSON lacks a number of necessary structures to meet the requirements of peer-to-peer synchronization and so for Testbed-12 an XML response shall be used since it allows already-existing structures such as wfs:FeatureCollection and/or wfs:Transaction to be reused.

7.5. Relationship to GSS

This clause discusses the relationship between the synchronization protocol describe herein and the GeoSynchronization Service (GSS, see OGC 10-069r2).

The purpose of a GeoSynchronization service is to allow organizations to deploy a transactional WFS to a "crowd" without giving the crowd direct transactional access to the server. Rather the GSS mediates the transactional interaction between members of the crowd and the WFS allowing the organization to validate data before being committed to the server. In this sense a GSS is a workflow manager for crowdsourcing WFSs. A convenient way to think about GSS is as an OGC® standards-based version of Open Street Map. Unlike Open Street Map, however, which replies on the crowd to verify data, the GSS enforces a more formal and mandatory approach to data validation.

In addition to acting as a crowdsourcing manager, the GSS specification describes a subscription subsystem that allows interested parties to be notified by a GSS whenever changes are committed to the WFS that the GSS is managing. This subscription subsystem can also be used synchronize interested third-party WFSs with the crowdsourced WFS that is managed by the GSS.

The current GSS synchronization subsystem is a master-slave, subscription-based, push synchronization. This means that whenever a set of changes is committed to the crowdsourced WFS server managed by the GSS, the GSS will push those changes (via a WFS transactions) to whichever third-party servers have subscribed to the GSS for the purpose of synchronization (see 7.2, OGC 15-010r4).

By contrast, the enterprise-to-enterprise synchronization capability tested in the OGC® Testbed-12 describes the mechanism and protocol for synchronizing two peer WFSs but does not describe when synchronization occurs or how it is triggered. It is a more light-weight synchronization

capability since no mediating or coordinating service such as GSS is required. However, only WFSs that have implemented the necessary resources and protocol can participate in the synchronization thus preventing legacy WFSs for being synchronized. The GSS, on the other hand, does not require any changes to the WFSs participating in a synchronization activity and so legacy as well as current WFSs can be synchronized. It seems reasonable, moving forward, that the enterprise-to-enterprise synchronization protocol be integrated into the GSS as part of its synchronization subsystem. However, time and resource limitation during the OGC® Testbed-11 did not allow for this investigation to be performed and so this is flagged as a future work item.

Appendix A: Technical Review comments

- **Comment 1**

Not sure you can reference 14-102 as it has never been uploaded outside of the SWG as pending for the TC, let alone public.

- **Response to comment 1**

The reasons for referencing OGC 14-102 are:

- The servers deployed for this experiment using a REST binding of WFS. This binding is most completely described in OGC 14-102.
- This work is experimental. By the time all the kinks have been worked out and the content of this ER makes its was into the WFS specification, 14-102 will likely be an adopted version.
- **Comment 2**

In 7.4.4 you have:

- "These parameters shall be set by a responding peer to communicate its service identifier and the checkpoint value to a requesting peer."

and

- "These parameters may be set by a requesting peer to communicate its service identifier and a checkpoint value to a responding peer."

Does the responding server (beta) keep the record of requester and checkpoint? I do see where the requester (alpha) keeps the responder and checkpoint. I thought you had mentioned that in a telecon, but I missed it here. Is it optional for the responding server to store that record?

- **Response to comment 2**

In order for a server to implement the sync operation it must do 2 things:

1. It must keep track of all changes made to all the feature that is offers (i.e. an AUDIT TRAIL)
2. For each requester it must keep track of the checkpoints in order to know up to which point each requester has been synced.

Requesters are identified using the "serviceId" parameter and the checkpoints are pointers into the responder’s audit trail indicating that, that specific requester has sync’ed up to that point.

- **Comment 3**

In 7.4.2, If I, as Alpha, change my filter between requests, does that negate the checkpoint? Say I want just schools within 1000 meters of me and we sync. Then I say I want schools within 2000 meters. Will the server limit the schools within 1000 meters response to changes since the sync and

then give all schools between 1000 and 2000 meters, or would it think of it as a new sync and give all features within 2000 meters?

- **Response to comment 3**

Except for the initial sync, every sync request has to include a checkpoint and every sync response (i.e. ChangeSet) has to include the next checkpoint to use. Put another way, any Sync request will negate the checkpoint specified on the request once the operation has completed (because the responding server will give you a new checkpoint each time). This is true whether there is a filter on the Sync request or not. The filter has no effect on which records appear in the change set or the checkpoint. The filter only affects which records from the change set are transmitted (or presented) to the requester.

Explaining just a bit further, you said "Say I want just schools within 1000 meters of me and we sync". The way you pose your question implies that it is the filter the determines which features appear in the change set. This is not the case at all. It is the Transactions that have been executed on the responding server that determine which features appear in the change set. Once the set of changed features is known, THEN you can say give me all the schools in the change set that are within 1000 meters of me. If no schools were modified (insert, update or delete) since the last checkpoint, no schools will appear in the change set and so your "schools within 1000 meters of me" filter will not return any rows. How does the responding server know which features are in the change set? It simply looks that information up in the AUDIT TRAIL that I mentioned in answer to your first question.