# Apéndice D
# Especificación Web Feature Service

## [D. Web Feature Service]

### i. Prefacio

Un logro importante de la iniciativa *Open GIS Consortium (OGC) Web Mapping Test bed (WMT1)* fue el desarrollo de un largo consenso alrededor de especificaciones de interfaces abiertas basadas en *web*. Tales especificaciones permiten a los vendedores de software implementar sus productos usando interfaces interoperables y proveer a los usuarios finales un gran conjunto de herramientas interoperables basadas en *web* para el acceso a geo-datos y servicios relacionados con geo-procesamiento.

Durante el proyecto OGC WMT1, fueron desarrollados los dos documentos borradores de especificaciones basadas en *web*:

1. OpenGIS© *Web Map Service Implementation Specification*
2. OpenGIS© *Geography Markup Language (GML) 2.0 Implementation Specification.*

El primer documento especifica interfaces *web* basadas en un modelo que soporta reglas generales de petición y respuesta (*request & response*) usando *Hypertex Transfer Protocol* (HTTP) y *eXtensible Markup Language* (XML). Se han desarrollado productos *Web Map Server* como resultado de la adopción por parte del OGC del documento OpenGIS© *Web Map Service Implementation Specification*. Las interfaces definidas en esta especificación incluyen *GetCapabilities, GetMap* y *GetFeatureInfo*[1]. El segundo documento describe una especificación para codificar geo-datos en XML. La codificación descrita en esta especificación permite el transporte y almacenamiento de información geográfica en XML incluyendo las propiedades y la geometría de los *features* geográficos.

Este documento (OpenGIS© *Web Feature Service Implementation Specification*) toma el siguiente paso lógico y propone intefaces para describir operaciones de manipulación de datos sobre *features* geográficos usando HTTP como la plataforma de cómputo distribuida. Las operaciones de manipulación de datos incluye la habilidad de:

1. Crear una nueva instancia de *feature*.
2. Borrar una instancia de *feature*.
3. Actualizar una instancia de *feature*.
4. Obtener o Consultar *features* mediante restricciones espaciales o no espaciales.

La petición de un *Web Feature Service* (WFS) consiste de una descripción de consulta o de una operación de transformación que será aplicada a uno o más *features*. La petición es generada en el cliente y es enviado a un *web feature service* usando HTTP. El *web feature service* entonces lee y (en un sentido) ejecuta la petición.

[1] Estas son operaciones de un WMS. Esta es una breve descripción: ***GetCapabilities***: operación que regresa un meta-dato con el nivel de servicio, ***GetMap:*** operación que regresa un mapa, cuyos parámetros geo-espaciales y dimensionales están bien definidos, ***GetFeatureInfo:*** operación que regresa la información de un *feature* particular mostrado en el mapa.

## Introducción

El OGC *Web Map Service* permite a un cliente cargar imágenes de mapas para mostrarlas desde múltiples *Web Map Services* en Internet. De una manera similar, el OGC *Web Feature Service* permite a un cliente recuperar datos geo-espaciales codificados en *Geography Markup Language* (GML) desde múltiples *Web Feature Services*.

Los requisitos para un *Web Feature Service* son:

1. Las interfaces deben estar definidas en XML.
2. GML debe ser usado para expresar *features* en las interfaces.
3. Como mínimo, un WFS debe poder presentar *features* usando GML.
4. El predicado o lenguaje filtro será definido en XML y ser derivado de CQL como se definió en el OpenGIS *Catalogue Interface Implementation Specification*.
5. La base de datos usada para almacenar *feature* geográficos debe ser opaca[2] a las aplicaciones clientes y su única vista de los datos debe ser a través de la interface del WFS.
6. Usar subconjuntos de expresiones XPath para referirnos a las propiedades.

El propósito de este documento es describir una interface *Web Feature Service*, como se ilustra en la figura 1.

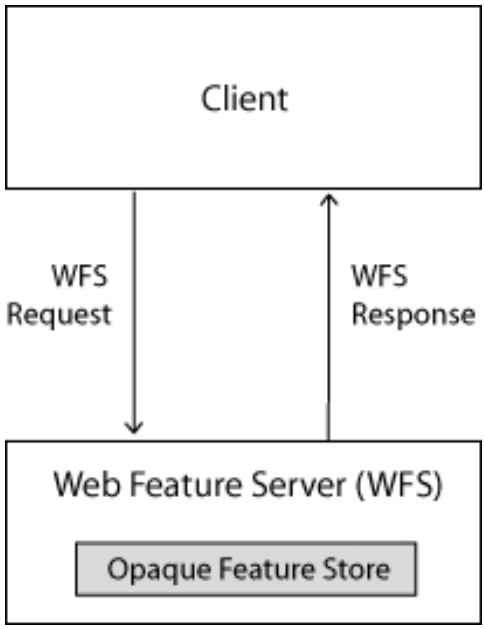


**Figura 1 – Web Feature Service**

Este documento es derivado de un largo consenso entre sus colaboradores y toma sus bases de dos especificaciones propuestas independientemente tiuladas: "OGC *Transaction Encoding Specification"* y "*Spatial Object Transfer Format* (SOTF)". Además un gran número de conceptos, comunes entre los servicios OGC, son tomados de la especificación de la implementación de un *Web Map Service*.

[2] **Opaca**: No visible, sin significado o sin importancia a la aplicación cliente**.**

## 1 Ámbito

Este documento describe las operaciones del OGC *Web Feature Service* (WFS). Las operaciones soportadas por un WFS sobre *features* geográficos son *INSERT, DELETE, QUERY* y *DISCOVERY*[3] usando HTTP como la plataforma de cómputo distribuida.

En el contexto de este documento, una transacción es la unidad lógica de trabajo que esta compuesta por una o más operaciones de manipulación de datos. Ya que la manera en la cual los *features* geográficos son almacenados persistentemente no es tratada en este documento, se asume que no existen semánticas de transacción (como puede ser una falla atómica). Es la función de un *web feature service*, en su interacción con los sistemas de almacenamiento de datos usados para almacenar persistentemente *features*, el garantizar que los cambios a los datos son consistentes. Sin embargo, el documento también reconoce el hecho de que varios sistemas sí soportan semánticas de transacción concurrente estándar, así que propone operaciones opcionales que permitirán a un *web feature service* tomar ventaja de tales sistemas (p.e. sistemas de bases de datos relacionales basadas en SQL).

### *Features* (rasgos) geográficos

Este documento adopta el mismo concepto de un *feature* geográfico como está descrito en la especificación abstracta del OGC e interpretado en el documento OpenGIS© *Geographic Markup Language (GML) Implementation* Specification, la cual dice que el estado de un *feature* geográfico esta descrito por un conjunto de propiedades donde cada una de ellas puede pensarse como una tupla {nombre, tipo, valor}. El nombre y el tipo de cada propiedad del *feature* es determinada por su propia definición de tipo. Los *features* geográficos son aquellos que pueden tener al menos una propiedad que es valorada geométricamente. Esto, por supuesto, implica que los *features* pueden ser definidos con ninguna propiedad geométrica. Las geometrías de los *features* geográficos están restringidas a lo que el OGC llama geometrías simples. Una geometría simple es aquella para la cual las coordenadas están definidas en dos dimensiones y la delineación de la curva está sujeta a la interpolación lineal. Las geometrías tradicionales de 0,1 y 2 dimensiones definidas en un sistema de referencia espacial de 2 dimensiones están representadas por puntos, líneas y polígonos. Además, el modelo geométrico del OGC permite, o colecciones homogéneas de multipuntos, multilíneas o multipolígonos, o permite colecciones de geometrías heterogéneas para las geometrías que son colecciones de otras geometrías. Finalmente, GML permite *features* que tienen propiedades no geométricas, complejas o agregadas.

### Procesando peticiones (*requests*)

Aquí se explica, en términos generales, el protocolo a ser seguido para procesar peticiones en un *web feature service*. El procesamiento de peticiones funciona de la siguiente manera:

1. Una aplicación cliente pide el documento de capacidades (*capabilities*) de un WFS. Tal documento contiene una descripción de todas las operaciones que un WFS soporta y la lista de todos los *features* que puede servir.
2. Una aplicación cliente (opcionalmente) hace la petición a un *web feature service* de la definición de uno o más tipos de *features* que un WFS puede servir.
3. Basado en la definición de los tipos de *feature*, la aplicación cliente genera una petición como se especifica en este documento.
4. La petición es enviada a un *web Server*.
5. El WFS es invocado a leer y servir la petición.

[3] ***Insert***: inserta un nuevo *feature*, ***Delete***: elimina un *feature* existente, ***Query***: consulta un *feature* mediante restricciones espaciales y no espaciales, ***Discovery***: operaciones que permiten saber que *features* estan disponibles en el servidor, que consultas se pueden ejecutar sobre esos *features* y que estructura guardan internamente.

6. Cuando el WFS ha completado el procesamiento de la petición, generará un reporte de *status* y lo enviará de vuelta al cliente. En el caso de que un error ocurriera, el reporte de *status* indicará el hecho.

**Operaciones**

Para soportar el procesamiento de transacciones y de consultas, están definidas las siguientes operaciones:

- *GetCapabilities*
  - Un *web feature service* debe ser capaz de describir sus capacidades. Específicamente, debe indicar cuáles tipos de *feature* puede servir y qué operaciones están soportadas sobre cada tipo de *feature*.

- *DescribeFeatureType*
  - Un *web feature service* debe ser capaz, en una petición, de describir la estructura de cualquier tipo de *feature* que pueda servir.

- *GetFeature*
  - Un *web feature service* debe ser capaz de servir una petición para recuperar instancias de *feature*.

- *Transaction*
  - Un *web feature service* puede ser capaz de servir peticiones de transacciones. Una petición de transacción está compuesta de operaciones que modifican *features*; estas operaciones son crear, actualizar y borrar *features* geográficos.

- *LockFeature*
  - Un *web feature service* puede ser capaz de procesar una petición de bloqueo sobre una o más instancias de un tipo de *feature* en la duración de una transacción. Esto garantiza que son soportadas las transacciones en serie.

Basado en la descripción de operaciones anterior, dos clases de *web feature service* pueden ser definidas:

- *Basic WFS*
  - Un WFS básico implementa las operaciones *GetCapabilities, DescribeFeatureType* y *GetFeature*. Este podría ser considerado un *web feature service READ-ONLY*.

- *Transaction WFS*
  - Un WFS transaccional soporta todas las operaciones de un *web feature service* básico y además implementa las operaciones de transacción. Opcionalmente, un WFS transaccional puede implementar la operación *LockFeature*.

La figura 2 es un diagrama del protocolo simplificado, ilustrando los mensajes que pueden ser pasados atrás y adelante entre la aplicación cliente y un *web feature service* para lograr procesar una petición típica. Los elementos referenciados en el diagrama están definidos en este documento.

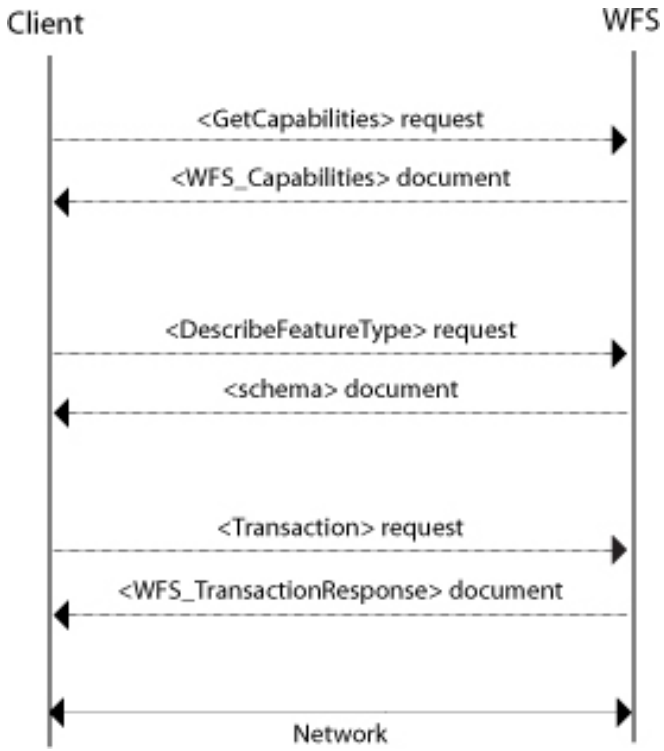


**Figura 2 – Diagrama del protocolo**

## 2 Conformidad

Para estar conforme a esta especificación se deberá checar usando todas las pruebas relevantes específicadas en el anexo D (normativo). El marco de trabajo, conceptos y metodología para pruebas, y criterios para lograr estar conforme, se especifican en ISO 19105: Información geográfica – *Conformance and Testing.*

## 3 Referencias Normativas

[1] Bradner, Scott, "RFC 2119 Key words for use in RFCs to Indicate Requirement Levels," March 1997, ftp://ftp.isi.edu/in-notes/rfc2119.txt .

[2] Cox, S., Cuthbert, A., Lake, R., and Martell, R. (eds.), "OpenGIS Implementation Specification #02-009: OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.1", April 2002

[3] Vretanos, Panagiotis (ed.), "OpenGIS Implementation Specification #01-067: Filter Encoding Implementation Specification", May 2001

[4] Percivall, George, ed., “The OpenGIS Abstract Specification, Topic 12: OpenGIS Service Architecture”, 2002

[5] Bray, Paoli, Sperberg-McQueen, eds., "Extensible Markup Language (XML) 1.0", 2nd edition, October 2000, W3C Recommendation, http://www.w3.org/TR/2000/REC-xml.

[6] Beech, David, Maloney, Murry, Mendelson, Noah, Thompson, Harry S., "XML Schema Part 1: Structures", May 2001, W3C Recommendation, http://www.w3c.org/TR/xmlschema-1.

[7] Bray, Hollander, Layman, eds., "Namespaces In XML", January 1999, W3C Recommendation, http://www.w3.org/TR/2000/REC-xml-names.

[8] Clark, James, DeRose, Steve, "XML Path Language (XPATH), Version 1.0", November 1999, W3C Recommendation, http://www.w3c.org/TR/XPath.

[9] Fielding et. al., "Hypertext Transfer Protocol – HTTP/1.1," IETF RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt.

[10] Berners-Lee, T., Fielding, N., and Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", IETF RFC 2396, http://www.ietf.org/rfc/rfc2396.txt.

[11] National Center for Supercomputing Applications, "The Common Gateway Interface," http://hoohoo.ncsa.uiuc.edu/cgi/.

[12] Freed, N. and Borenstein N., "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", IETF RFC 2045, November 1996, http://www.ietf.org/rfc/rfc2045.txt.

[13] Internet Assigned Numbers Authority, http://www.isi.edu/in-notes/iana/assignments/media-types/.

[14] de La Beaujardière, Jeff (ed.), "OpenGIS Implementation Specification #01-047r2: Web Map Service Implementation Specification", June 2001

[15] Vretanos, Panagiotis, "OpenGIS Discussion Paper: Transaction Encoding Specification Version 0.0.5", March 2000

[16] Arctur, D., Pilkington, P., Cuthbert, A., "Spatial Object Transfer Format (SOTF): Initial High-Level Design, Version 1.2", Laser-Scan Inc., November 1999

[17] Rumbach, James, et al., "Unified Modeling Language Reference Manual", 1999

[18] Murata, St. Laurent, Kohn, "XML Media Types, January 2001, http://www.ietf.org/rfc/rfc3023.txt

## 4 Términos y definiciones

Para el propósito de este documento, se aplican los siguientes términos y definiciones.

**4.1 Operación:**
Especificación de una transformación o consulta que se le puede pedir a un objeto que ejecute.

**4.2 Interface:**
Un conjunto nombrado de operaciones que caracteriza el comportamiento de una entidad.

**4.3 Servicio:**
Una parte definida de la funcionalidad que es ofrecida por una entidad a través de interfaces.

**4.4 Instancia de servicio:**
Una implementación actual de un servicio; instancia de servicio es sinónimo de servidor.

**4.5 Cliente:**
Un componente de software que puede invocar una operación de un servidor.

**4.6 Petición (request):**
Una invocación hecha por un cliente para que se realice una operación.

**4.7 Respuesta (response):**
El resultado de una operación que regresa el servidor al cliente.

**4.8 Capacidades XML:**
Meta datos acerca del nivel de servicio, el cual describe las operaciones y el contenido disponible en una instancia de servicio.

**4.9 Sistema de referencia espacial:**
Como está definido en ISO 19111

**4.10 Opaco:**
No visible, sin significado o sin importancia a la aplicación cliente.

## 5 Convenciones

### 5.1 Verbos normativos

En las secciones etiquetadas normativas, las palabras **debe, no debe, requisito, recomendado, puede, opcional,** se interpretarán en este documento como se describe en el documento RFC 2119 *Keywords for use in RFCs to Indicate Requirement Levels.*

### 5.2 Términos abreviados

- CGI Common Gateway Interface
- DCP Distributed Computing Platform
- DTD Document Type Definition
- EPSG European Petroleum Survey Group
- GIS Geographic Information System
- GML Geography Markup Language
- HTTP Hypertext Transfer Protocol
- IETF Internet Engineering Task Force
- MIME Multipurpose Internet Mail Extensions
- OGC Open GIS Consortium
- OWS OGC Web Service
- URL Uniform Resource Locutor
- WFS Web Feature Service
- XML Extensible Markup Language

### 5.3 Uso de ejemplos

Esta especificación hace uso extensivo de ejemplos XML, con la intención de ilustrar los distintos aspectos del *web feature service* que se trata en esta especificación. A pesar de que se hizo un esfuerzo por garantizar que los ejemplos estuvieran bien formados y fueran válidos, esta meta fue sacrificada en muchos casos por razones de claridad, es decir, varios ejemplos están con un formato específico para destacar un aspecto particular que haría inválido el ejemplo desde la perspectiva de una herramienta de validación de XML. Además, la mayoría de los ejemplos hacen referencia a datos y servidores ficticios.

De este modo, esta especificación no afirma que algún ejemplo codificado en XML o en par de palabra-valor (*keyword-value pair*), copiado de este documento, se ejecutará ó validará correctamente en una herramienta de validación de XML. Solo en las secciones marcadas como "normativas" deberá esperarse documentos o esquemas XML bien formados y válidos.

## 6 Elementos del servicio básico

### 6.1 Introducción

Esta sección describe aspectos del comportamiento de un OGC *web feature service* que son independientes de operaciones particulares o son comunes a varias operaciones o interfaces.

### 6.2 Negociación y numerado de versiones

#### 6.2.1 Forma de numerar las versiones

El número de versión de la especificación publicada contiene tres enteros positivos, separados por puntos decimales, en la forma "x.y.z". Los números "y" y "z" nunca excederán de 99. Cada especificación OWS es numerada independientemente.

#### 6.2.2 Cambios en las versiones

El número de versión de una especificación particular debe ser cambiada con cada revisión. El número debe incrementar monolíticamente y debe comprender no más de tres enteros separados por puntos decimales, siendo el primer entero el más significativo. Puede haber espacios en la secuencia numérica. Algunos números pueden denotar versiones experimentales o provisionales. Las instancias de servicio y sus clientes no necesitan soportar todas las versiones, pero deben obedecer las reglas de negociación que vienen más adelante.

#### 6.2.3 Aspecto en peticiones y en meta-datos de servicio

El número de versión aparece en al menos dos lugares: en el XML de capacidades que describe a un servicio y en la lista de parámetros del cliente que solicita ese servicio. El número de versión usado en la petición de una instancia de un servicio particular hecha por el cliente debe ser igual al número de versión que la instancia ha declarado que soporta (excepto durante la negociación como se describe a continuación). Una instancia de servicio puede soportar varias versiones cuyos valores pueden ser descubiertos por los clientes según las reglas de negociación.

#### 6.2.4 Negociación de número de versión

Un cliente OWS puede negociar con una instancia de servicio para determinar una versión mutuamente conforme de la especificación. La negociación es llevada a cabo usando la operación *GetCapabilities* conforme a las siguientes reglas.

Todos los XML de capacidades deben incluir un número de versión del protocolo. En respuesta a la petición *GetCapabilities* que contiene un número de versión, un OGC *web service* debe responder con la salida que se ajusta a la versión de la especificación o negociar una versión mutuamente conforme, si es que la versión solicitada no está implementada en el servidor. Si no es especificado el número de versión en la petición, el servidor deberá responder con la versión más alta que entienda y etiquetar de acuerdo a eso la respuesta.

La negociación del número de versión ocurre como sigue:

1. Si el servidor implementa el número de versión solicitado, entonces deberá enviar esa versión.
2. Si la petición del cliente es por una versión desconocida más grande que la versión más baja que el servidor entiende, el servidor deberá enviar la versión más alta que no pase a la versión solicitada.
3. Si la petición del cliente es por una versión inferior que cualquiera de las conocidas por el servidor, entonces el servidor deberá enviar la versión más baja que conozca.
4. Si el cliente no entiende la nueva versión enviada por el servidor, puede interrumpir la comunicación con el servidor o enviar una nueva solicitud con un número de versión que el cliente entienda, pero la cual es inferior a la enviada por el servidor (si el servidor había respondido con una versión más baja).
5. Si el servidor había respondido con una versión superior (porque la petición fue por una versión inferior que cualquiera conocida por el servidor), y el cliente no entiende la versión superior propuesta, entonces el cliente puede enviar una nueva petición con una versión superior que la que envió el servidor.

El proceso es repetido hasta que sea alcanzada una versión mutuamente entendible, o hasta que el cliente determine que no puede o que no se comunicará con un servidor particular.

**Ejemplo 1:** El servidor entiende las versiones 1, 2, 4, 5 y 8. El cliente entiende las versiones 1, 3, 4, 6 y 7. El cliente solicita la versión 7. El servidor responde con la versión 5. El cliente solicita la versión 4. El servidor responde con la versión 4, la cual el cliente entiende, y la negociación termina satisfactoriamente.

**Ejemplo 2:** El servidor entiende las versiones 4, 5 y 8. El cliente entiende la versión 3. El cliente solicita la versión 3. El servidor responde con la versión 4. El cliente no entiende esa versión o cualquier versión superior, por lo tanto la negociación falla y el cliente interrumpe la comunicación con el servidor.

### 6.3 Reglas generales de solicitudes HTTP

#### 6.3.1 Introducción

Actualmente, la única plataforma de cómputo distribuido (DCP) explícitamente soportada por los OGC *Web Services* es la propia *World Wide Web*, o más específicamente, los *hosts* en Internet que implementan el protocolo de transferencia de hipertexto (HTTP). Por consiguiente el recurso en línea de cada operación soportada por una instancia de servicio es ubicado por un Localizador de Recursos Uniforme (URL) de HTTP. El URL puede ser diferente para cada operación, o puede ser el mismo, esto es a discreción del proveedor del servicio. Cada URL debe ajustarse a la descripción del documento *"Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies"*[4], pero es de otra manera dependiente de la implementación; sólo los parámetros que abarcan la petición misma del servicio están bajo el mandato de las especificaciones de los OGC *web services*.

HTTP soporta dos métodos de petición: *GET* y *POST*. Uno o ambos de estos métodos pueden ser definidos por un tipo de OGC *web service* particular y ser ofrecidos por una instancia de servicio. El uso del recurso en línea URL difiere en cada caso.

#### 6.3.2 HTTP GET

Un recurso en línea URL pensado para las peticiones HTTP GET es, de hecho, solo un prefijo del URL al cual se deben agregar parámetros adicionales para lograr construir una petición de operación válida. Un prefijo URL se define como una cadena opaca que incluyen el protocolo, *hostname*, número de puerto opcional, ruta, signo de interrogación '?' y, opcionalmente, uno o

[4] [FREED, Nov. 1996] http://www.ietf.org/rfc/rfc2045.txt

más parámetros específicos del servidor terminando en un *ampersand* '&'. El prefijo únicamente identifica la instancia particular del servicio. Un cliente anexa los parámetros necesarios a la petición como pares, nombre/valor de la forma, "nombre=valor&". El URL resultante debe ser válido conforme al estándar HTTP *Common Gateway Interface (CGI)*, el cual exige la presencia de '?' antes de la secuencia de los parámetros de consulta , y '&' entre cada parámetro. Como en las aplicaciones CGI, el URL de consulta es codificado para proteger caracteres especiales.

El prefijo del URL debe terminar en un '?' (en la ausencia de parámetros específicos del servidor) o en un '&'. En la práctica, sin embargo, los clientes deberían estar preparados para agregar una cola necesaria de '?' ó de '&' antes de anexar los parámetros de operación definidos en esta especificación, con el fin de construir una petición URL válida.

La tabla 1 resume los componentes de una operación de petición URL.

| **Componente URL** | **Descripción** |
|---|---|
| http://host[:puerto]/ruta?{nombre[=valor]&} | Prefijo URL de una operación del servicio. [ ] indica ocurrencia de 0 ó 1 de una parte opcional; { } indica 0 ó más ocurrencias. El prefijo es totalmente a discreción del proveedor del servicio. |
| nombre=valor& | Uno o más parámetros de petición estándar de la forma de pares nombre/valor definidos por un OGC *web service*. La lista actual de los parámetros opcionales y requeridos es exigida para cada operación por la especificación OWS apropiada. |

**Tabla 1 - Petición general de un OGC *Web Service***

### 6.3.3 HTTP POST

Un recurso en línea URL pensado para peticiones HTTP POST es un URL completo y válido al cual los clientes transmiten peticiones codificadas en el cuerpo del documento POST. Un WFS no debe requerir parámetros adicionales que se deban anexar al URL para lograr construir un objetivo válido para una petición de operación.

## 6.4 Reglas generales para respuestas HTTP

Al recibir una petición válida, el servicio debe enviar una respuesta que corresponda exactamente a la petición, como está detallado en la especificación apropiada.

Al recibir una petición inválida, el servicio debe emitir una Excepción de Servicio como se describirá en la sección 7.7.[5]

Los objetos de respuesta (*response*) deben ser acompañados por el tipo apropiado de *Multipurpose Internet Mail Extensións (MIME)* para ese objeto.

Los objetos de respuesta deberían estar acompañados por otros encabezados de entidades HTTP como es apropiado y tantos como sea posible. En particular, los encabezados de expiración y

[5] **Nota:** Como una materia práctica, en el ambiente WWW un cliente debe estar preparado a recibir un resultado válido, o nada, o cualquier otro resultado. Esto es porque el cliente mismo puede haber hecho una petición mal formada que, inadvertidamente, provoca una respuesta por parte de algo que no sea un OGC *web service*, ya que el servicio mismo puede no estar bien formado.

última modificación (*Expires, Last*-Modified) proveen una información importante para el caché; el encabezado de longitud del contenido (*Content-Length*) puede ser usado por los clientes para saber cuándo se ha completado la transmisión de los datos y asignar eficientemente espacio para los resultados, los encabezados de codificación de contenido y codificación de transferencia de contenido (*Content-Encoding, Conten-Transfer-Encoding*) pueden ser necesarios para la propia interpretación de los resultados.

### 6.5 Codificación de la petición

Este documento define dos métodos para codificar peticiones de un WFS. El primero usa XML como lenguaje de codificación. El segundo método usa pares de palabra-valor para codificar los numerosos parámetros de una petición. Un ejemplo de un par palabra-valor es:

"*REQUEST=GetCapabilities*"

Donde "*REQUEST*" es la palabra y "*GetCapabilities*" es el valor. En ambos casos, la respuesta a la petición o el reporte de una excepción deben ser idénticos.

La tabla 2 correlaciona las operaciones del WFS y sus semánticas de codificación como se definió en este documento.

| Operación | Codificación de la petición |
|---|---|
| *GetCapabilities* | XML y KVP |
| *DescribeFeatureType* | XML y KVP |
| *GetFeature / GetFeatureWithLock* | XML y KVP |
| *LockFeature* | XML y KVP |
| *Transaction* | XML y KVP limitado |

KVP = *Keyword-value pair* (par nombre-valor)

**Tabla 2 – Codificación de una petición de operación**

### 6.6 Espacios de nombre (*Namespaces*)

Los espacios de nombre son usados para distinguir vocabularios XML unos de otros. Para los WFS hay tres definiciones normativas para espacios de nombre, concretamente:

- (**http://www.opengis.net/wfs**) – para el vocabulario de la interface WFS
- (**http://www.opengis.net/gml**) – para el vocabulario GML
- (**http://www.opengis.net/ogc**) – para el vocabulario del filtro OGC

Una implementación de un WFS dada hará uso de uno o más esquemas de aplicación GML y estos esquemas usarán, en cambio, uno o más espacios de nombre de aplicación (p.e.: **http://www.algunservidor.com/myne**). Muchos de los ejemplos en este documento usan un espacio de nombre sencillo, pero pueden usarse espacios de nombre múltiples, como se muestran en la sección 11.2.6.

## 7 Elementos Comunes

### 7.1 Identificador de *feature*

Este documento asume que cada instancia de *feature* que una implementación particular de WFS puede manejar, es únicamente identificable. Es decir, cuando una implementación de WFS reporta un identificador de *feature* para una instancia de *feature*, ese identificador de *feature* es único al servidor y puede ser usado para referenciar repetidamente la misma instancia de

*feature* (asumiendo que no ha sido borrado). Más adelante se asume que un identificador de *feature* está codificado como se describe en el documento OpenGIS© *Filter Encoding Implementation Specification*. Un identificador de *feature* puede ser usado donde sea que sea requerida una referencia de *feature*. Para propósitos de referencia, se copió de la especificación de codificación el fragmento de esquema XML que define al elemento identificador del *feature*:

```
<xsd:element name="FeatureId" type="ogc:FeatureIdType"/>
<xsd:complexType name="FeatureIdType">
   <xsd:attribute name="fid" type="xsd:anyURI" use="required"/>
</xsd:complexType>
```

El propósito de un identificador de *feature* es hacer posibles las operaciones de bases de datos.

### 7.1.1 Identificadores únicos globales (Informativo)

Para propósitos de un WFS, un identificador único local es suficiente. Sin embargo, hay una necesidad dentro de los OGC *web services* de tener identificadores únicos para objetos de todos los tipos. El enfoque hasta aquí ha sido referenciar objetos usando componentes *feature-id* y *scope* independientes, donde el *scope* es el URL del servidor que sirve un *feature* y el *feature-id* es el identificador local para ese *feature*. Este enfoque, sin embargo, puede ser difícil de transportar y usar en otros contextos, por ejemplo en un registro si uno quisiera crear un metadato para una sola instancia del depósito de datos.

El propósito de esta sección de la especificación es precisar que una sola cadena única global sería más conveniente de usar en contextos múltiples, y que esa cadena puede ser generada por un *web feature service* usando alguna combinación del URL del servicio y el identificador local.

Esta cadena puede ser usada como si fuera totalmente opaca en muchos contextos, pero sería más útil si fuera en realidad un URL o un URN el cual puede ser usado para tener acceso directo al objeto que identifica en el formato nativo del objeto. La codificación del URL o URN sería completamente específica de la implementación. Una nota sobre el uso de URN: no muchas implementaciones en realidad podrán resolver y traer objetos de datos; la mayor parte de las veces sólo utilizable como una cadena de identificación única.

Usar un URL o URN es útil para aplicaciones que necesitan sólo un acceso simple a los objetos en bruto ya que los detalles de la interface no necesitan conocerse. Este modo de acceder/identificar es también de ayuda para la integración con tecnologías XML de alto nivel, como por ejemplo RDF o XSLT, e incluso para propósitos de depuración.

## 7.2 Estado del *feature*

La definición de *features* servida por un WFS es proporcionada por un esquema de aplicación de GML. La sección 5 describe cómo un cliente puede solicitar un documento XML que contiene la definición de esquema de aplicación de GML de uno o más tipos de *features* servidos por un WFS. Tal definición de esquema de aplicación deberá ajustarse al documento OpenGIS *Geography Markup Language (GML) Implementation Specification, version 2.1.2*.

Una aplicación cliente usa la definición del esquema de aplicación de un tipo de *feature* para referirse a instancias de *feature* de ese tipo de *feature*, y para referirse a los nombres y tipos de todas las propiedades de esas instancias de *feature*. Los valores de todas las propiedades de una instancia de *feature* constituyen el estado de esa instancia de *feature*. Una aplicación cliente se refiere a las instancias de *feature* por el nombre de sus tipos de *features* y por los nombres y valores de las propiedades del *feature*. Una aplicación cliente pide a un WFS transaccional alterar el estado de un *feature* mediante peticiones de operaciones *insert, update* y *delete*.

### 7.3 Nombres de propiedades

Un *web feature service* hace referencia a los nombres de propiedades del *feature* definidos en el esquema de aplicación de GML. Sin embargo, ya que el estado del *feature* debe ser expresado en GML y por lo tanto XML, los nombres de propiedades usados por un *web feature service* deben también ser nombre de atributos y elementos válidos como se describe en el documento *Extensible Markup Language (XML) 1.0 specification*. Además, los nombres de propiedades deben ser espacios de nombre calificados como se describe en el documento *Namespaces in XML*. Las siguientes definiciones fueron tomadas de las secciones 2 y 3 de ese documento:

```
[4]  NCName ::=  (Letter | '_') (NCNameChar)*
/*  An XML Name, minus the ":" */
[5]  NCNameChar ::=  Letter | Digit | '.' | '-' | '_' | CombiningChar | Extender
[6]  QName ::=  (Prefix ':')? LocalPart
[7]  Prefix ::=  NCName
[8]  LocalPart ::=  NCName
```

Las definiciones de los componentes *Letter, Digit, CominingChar* y *Extender* son especificadas en el anexo B del documento *Extensible Markup Language (XML) 1.0 specification*.

**Ejemplo**

Ejemplos de nombres de propiedades válidos:

Edad, Temperatura,_Khz, myne:ENAGUAA_1.WKB_GEOM

Ejemplos de nombres de propiedades inválidos:

+Dominio, 123_AlgunNombre

### 7.4 Referencias de la propiedad

#### 7.4.1 Introducción

Como se mencionó en la introducción, GML permite a los *features* geográficos tener propiedades no geométricas complejas o agregadas. Un problema surge acerca de cómo tales propiedades deben ser referenciadas en los varios lugares donde las referencias de la propiedad son requeridas. (p.e.: expresiones de consulta y de filtrado). Un WFS debe usar Expresiones XPath, como se definen en el documento *XML Path Language (XPath) version 1.0*, para referirse a las propiedades y sub-propiedades de un *feature* codificado como atributos o elementos XML.

#### 7.4.2 Expresiones XPath

XPath, es un lenguaje para tratar partes de un XML, o en nuestro caso, para referirse a propiedades de *feature* que son referidas por medio de atributos o elementos XML.

Esta especificación no requiere que la implementación del WFS soporte el lenguaje XPath completamente. Para lograr mantener el costo de entrada de la implementación tan bajo como sea posible, esta especificación exige que una implementación de WFS deba soportar el siguiente subconjunto del lenguaje XPath:

1. Una implementación de WFS debe soportar rutas de localización relativas abreviadas.
2. Las rutas de localización relativas están compuestas de uno o más pasos separados por el separador '/'.
3. El primer paso de una ruta de localización relativa puede corresponder al elemento raiz de la propiedad del *feature* que es referida o puede ser el elemento raíz del tipo de *feature* con el siguiente paso correspondiente al elemento raíz de la propiedad del *feature* que es referida.

4. Cada paso subsiguiente en la ruta debe estar compuesto de la forma abreviada: especificador de eje *child::* y el nombre de la propiedad del *feature* codificada como el tipo principal de nodo de *element*. La forma abreviada del especificador de eje *child::* es simplemente omitir el especificador del paso de localización.
5. Cada paso en la ruta opcionalmente puede contener un predicado compuesto de los delimitadores '[' y ']' y un número que indica cual hijo del contexto del nodo está seleccionado. Esto permite a las propiedades del *feature* que pueden estar repetidas, ser referidas específicamente.
6. El paso final en una ruta opcionalmente puede estar compuesto de la forma abreviada: especificador de eje *attribute::*, '@', y el nombre de la propiedad del *feature* codificada como tipo principal de nodo de *attribute*.

**Ejemplo**

Para ilustrar prácticamente el uso de expresiones XPath para referirnos a las propiedades de un *feature* complejo (codificadas como elementos o atributos), consideremos el *feature* complejo ficticio *Person* definido por el siguiente esquema XML:

```
<?xml version="1.0" ?>
<schema
   targetNamespace="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="1.0">

   <import namespace="http://www.opengis.net/gml"
           schemaLocation="../gml/2.1/feature.xsd"/>

   <element name="Person" type="myns:PersonType"
            substitutionGroup="gml:_Feature"/>
   <complexType name="PersonType">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="LastName" nillable="true">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="30"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="FirstName" nillable="true">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="10"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="Age" type="integer" nillable="true"/>
               <element name="Sex" type="string"/>
               <element name="Spouse">
                  <complexType>
                     <attribute name="sin" type="xsd:anyURI" use="required" />
                  </complexType>
               </element>
               <element name="Location"
                        type="gml:PointPropertyType"
                        nillable="true"/>
               <element name="Address" type="myns:AddressType" nillable="true"/>
               <element name="Phone" type="xsd:string"
                        minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="sin" type="xsd:anyURI" use="required"/>
         </extension>
      </complexContent>
   </complexType>
```

```
   <complexType name="AddressType">
      <sequence>
         <element name="StreetName" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="StreetNumber" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="10"/>
               </restriction>
            </simpleType>
         </element>
         <element name="City" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="Province" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
         <element name="PostalCode" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="15"/>
               </restriction>
            </simpleType>
         </element>
         <element name="Country" nillable="true">
            <simpleType>
               <restriction base="string">
                  <maxLength value="30"/>
               </restriction>
            </simpleType>
         </element>
      </sequence>
   </complexType>
</schema>
```

Hay que notar que la propiedad *Address* es una propiedad compleja y del tipo *AddresType*. Un ejemplo de una instancia del *feature Person*, podría ser:

```
<?xml version="1.0" ?>
<myns:Person
   sin="111222333"
   xmlns:myns="http://www.opengis.net/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/myns Person.xsd">

   <myns:LastName>Smith</myns:LastName>
   <myns:FirstName>Fred</myns:FirstName>
   <myns:Age>35</myns:Age>
   <myns:Sex>Male</myns:Sex>
   <myns:Spouse sin="444555666" />
   <myns:Location>
      <gml:Point><gml:coordinates>15,15</gml:coordinates></gml:Point>
   </myns:Location>
   <myns:Address>
      <myns:StreetName>Main St.</myns:StreetName>
      <myns:StreetNumber>5</myns:StreetNumber>
      <myns:City>SomeCity</myns:City>
```

```
        <myns:Province>SomeProvince</myns:Province>
        <myns:PostalCode>X1X 1X1</myns:PostalCode>
        <myns:Country>Canada</myns:Country>
    </myns:Address>
    <myns:Phone>416-123-4567</myns:Phone>
    <myns:Phone>416-890-1234</myns:Phone>
</myns:Person>
```

Usando expresiones XPath, cada propiedad de un *feature Person* puede ser referido como sigue (se omiten los espacios de nombre calificadores por motivos de claridad)

```
LastName
FirstName
Age
Sex
Source
Location
Address
Address/StreetNumber
Address/StreetName
Address/City
Address/Province
Address/Postal_Code
Address/Country
Phone[1]
Phone[2]
```

Hay que notar que en esta instancia, cada ruta relativa a una localización comienza con el elemento raíz de una propiedad del *feature* que es referida. Esto simplemente corresponde al nombre de la propiedad del *feature*. Opcionalmente, cada propiedad del *feature* puede ser referida con una ruta relativa a la localización comenzando con el elemento raíz del *feature* (p.e.: el nombre del tipo de *feature*). Así, la propiedad *LastName* puede ser referida como *Person/LastName*, la propiedad *City* puede ser referida como *Person/Address/City* etcétera.

Cada paso de la ruta está compuesto por el especificador de eje *child::* abreviado (p.e.: podemos omitir el especificador *child::*) y el nombre de la propiedad especificada la cual es de tipo de nodo *element*.

El elemento P*hone* aparece en varias ocasiones y los predicados [1] y [2] son usados para indicar específicamente los elementos. El predicado [1] es usado para indicar la primera ocurrencia del elemento *Phone*. El predicado [2] es usado para indicar la segunda ocurrencia del elemento *Phone*.

Además, el atributo *sin*[6] en los elementos <*Person*> y <*Spouse*> puede ser referido usando las siguientes expresiones XPath:

```
Person/@sin
Person/Spouse/@sin
```

En esos casos el paso final de la ruta contiene el especificador de eje abreviado *attribute::* (p.e.: @) y el tipo de nodo es *attribute* (p.e.: *sin* en este caso)

### 7.5 Elemento <*native*>

Está claro que una interface abierta puede soportar sólo un cierto conjunto común de capacidades. El elemento <*native*> está destinado a permitir el acceso a capacidades específicas del vendedor de cualquier *web feature server* o base de datos particular.

El elemento <*native*> está definido por el siguiente fragmento de esquema XML:

[6] *SIN = Social Insurance Number*

```
<xsd:element name="Native" type="wfs:NativeType"/>
<xsd:complexType name="NativeType">
   <xsd:any />
   <xsd:attribute name="vendorId" type="xsd:string" use="required"/>
   <xsd:attribute name="safeToIgnore" type="xsd:boolean" use="required"/>
</xsd:complexType>
```

El elemento <*native*> contiene simplemente el comando u operación específico del vendedor.

El atributo *vendorID* es usado para identificar al vendedor que reconoce el comando o la operación encerrado por los elementos <*Native*>. El atributo es proporcionado como un medio para permitir a un *web feature service* determinar si puede negociar con el comando o no.

El atributo *safeToIgnore* es usado para guiar las acciones de un *web feature service* cuando no se reconoce el comando o la operación <*Native*>. El atributo *safeToIgnore* tiene dos posibles valores *True* ó *False*. Los valores tienen los siguientes significados:

- *safeToIgnore=False*
  Un valor de *False* indica que el elemento <*Native*> no puede ser ignorado y que la operación a la que ese elemento está asociada debe fallar si el *web feature service* no puede negociar con ella.

- *safeToIgnore=True*
  Un valor de *True* indica que el elemento <*Native*> puede ser ignorado de manera segura.

**Ejemplo**

Este ejemplo ilustra el uso del elemento <*Native*> para activar una característica especial de una base de datos relacional basada en SQL. En esta instancia, el elemento indica que esto es un comando *Oracle* y que el comando puede ser ignorado de manera segura.

```
<Native vendorId="Oracle" safeToIgnore="True">
ALTER SESSION ENABLE PARALLEL DML
</Native>
```

### 7.6 Filtro

Un filtro es usado para definir un conjunto de instancias de *feature* sobre las que se va a operar. El conjunto de operaciones puede estar comprendido de uno o más *features* enumerados o de un conjunto de *features* definidos por restricciones espaciales y no espaciales sobre propiedades geométricas y escalares de un tupo de *feature*. La especificación del filtro debe ser codificada como se describe en el documento OGC *Filter Encoding Implementation Specification*.

### 7.7 Reporte de excepciones

En el caso de que un *web feature service* encuentre un error mientras procesa una petición o recibe una petición irreconocible, debe generar un documento XML indicando que error ha ocurrido. El formato de la respuesta de error en XML está especificada por, y debe ser validada contra, el esquema de respuesta de excepción definido en la sección A.2.

Un elemento <*ServiceExceptionReport*> puede contener una o más excepciones de proceso de un WFS. El atributo *version* obligatorio es usado para indicar la versión del esquema de reporte de excepción del servicio. Para la versión de esta especificación el valor es ajustado a 1.2.0.

Los mensajes de excepción individual están contenidos dentro del elemento <*ServiceException*>. El atributo opcional *code* puede ser usado para asociar un código de excepción con el mensaje acompañante. El atributo opcional *Locutor* puede ser usado para

indicar donde fue encontrada la excepción en la petición que genera el error. Un número de elementos definidos en este documento incluye un atributo *handle* que puede ser usado para asociar nombre mnemónicos con el elemento. Si tal *handle* existe, su valor puede ser reportado usando el atributo *Locutor* del elemento <*ServiceExcption*>. Si el atributo *handle* no es pacificado entonces la implementación del *web feature* service puede intentar localizar el error usando otro medios como una línea de números, etc.

**Ejemplo**

El siguiente es un ejemplo de un reporte de excepción. Esta excepción indica que falló la primera declaración insertada porque faltó cerrar un etiqueta XML en la petición.

```
<?xml version="1.0" ?>
<ServiceExceptionReport
   version="1.2.0"
   xmlns="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/ogc ../wfs/1.0.0/OGC-exception.xsd">
   <ServiceException code="999" locator="INSERT STMT 01">
      parse error: missing closing tag for element WKB_GEOM
   </ServiceException>
</ServiceExceptionReport>
```

Hay que notar que la salida de este ejemplo se valida contra el esquema de reporte de excepciones presentado arriba.

### 7.8 Atributos XML comunes

#### 7.8.1 Atributo *version*

Todas las peticiones de WFS codificadas en XML incluyen un atributo llamado *version*. El atributo obligatorio *version* es usado para indicar a cuál versión de la especificación WFS se ajusta la petición codificada y es usada en la negociación de versiones descrita en la sección 6.2.4. El valor *deafult* del atributo *version* es 1.0.0, el cual corresponde a la versión de este documento.

#### 7.8.2 Atributo *service*

Todas las peticiones de WFS codificadas en XML incluyen un atributo llamado *service*. El atributo obligatorio *service* es usado para indicar cual de los tipos de servicio disponibles, en una instancia particular, está siendo invocado. Cuando se invoca un servicio de un *web feature service*, el valor del atributo *service* debe ser WFS.

#### 7.8.3 Atributo *Handle*

El propósito del atributo *handle* es permitir a la aplicación cliente asociar nombres mnemónicos a peticiones para propósitos de manejo de errores. Si se especifica un *handle*, y una excepción ocurre, un *web feature service* puede usar el *handle* para identificar el elemento ofendido.

## 8 Operación *DescribeFeatureType*

### 8.1 Introducción

La función de la operación *DescribeFeatureType* es generar un esquema con la descripción de los tipos de *feature* servidos por una implementación de WFS. Las descripciones del esquema definen cómo se espera que sean codificadas en la entrada las instancias de feature y cómo se espera que serán generadas en la salida las instancias de *features* en la implementación de un WFS.

### 8.2 Petición

Un elemento *DescribeFeatureType* contiene cero o más elementos *TypeName* que codifican los nombres de tipo de *feature* que van ser descritos. Si el contenido de un elemento *DescribeFeatureType* es vacío, entonces eso debe ser interpretado como una solicitud de todos los tipos de *feature* que un WFS puede servir. La codificación en XML para una petición *DescribeFeatureType* está definida por el siguiente fragmento de esquema XML:

```
<xsd:element name="DescribeFeatureType" type="wfs:DescribeFeatureTypeType"/>
<xsd:complexType name="DescribeFeatureTypeType">
   <xsd:sequence>
      <xsd:element name="TypeName" type="xsd:QName"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
   <xsd:attribute name="outputFormat"
                  type="xsd:string" use="optional" default="XMLSCHEMA"/>
</xsd:complexType>
```

El atributo *outputFormat*, es usado para indicar el lenguaje para el esquema de descripción que debe ser usado para describir los esquemas de tipos de *feature*. El único formato de salida obligatorio en la respuesta a una operación *DescribeFeatureType* es un esquema XML, denotado por el valor *XMLSCHEMA* en el atributo *outputFormat*. Otros formatos específicos de algún vendedor son también posibles, pero ellos deben ser anunciados en el documento de capacidades (sección 12).

Como se especificó en el documento OpenGIS© *Geography Markup Language (GML) Implementation Specification, version 2.1.2*, la definición de esquema de *feature* es completamente a la discreción de la implementación particular del WFS que describe sus tipos de *feature*. Las únicas advertencias son:

1. La geometría del f*eature* debe ser expresada usando la descripción geométrica de GML (geometry.xsd).
2. El sistema de referencia espacial debe ser expresado como se definió en el OpenGIS© *Geography Markup Language (GML) Implementation Specification, version 2.1.2*.
3. El esquema de *feature* debe ser consistente con el modelo de *feature* del OGC. Esto significa que los esquemas de *feature* definen propiedades del *feature*. La interpretación del GML de esta declaración es que los elementes anidados bajo el elemento raíz de un tipo de *feature* define las propiedades de ese *feature*.

### 8.3 Respuesta

En respuesta a una petición *DescribeFeatureType*, donde el valor del atributo *outputFormat* ha sido puesto en *XMLSCHEMA*, una implementación WFS debe poder presentar un esquema XML que es un esquema de aplicación GML válido y definir el esquema de los tipos de *features* listados en la petición. El(los) documento(s) presentado(s) por la petición *DescribeFeauterType* pueden ser usados para validar instancias de *features* generadas por el WFS en la forma de colecciones de *features* sobre una salida ó de instancias de *features* especificadas como entrada para operaciones de transacción.

También es posible que los esquemas de descripción usen otros lenguajes de descripción, como DTD, tantos como se hayan declarado en el documento de capacidades.

### 8.3.1 Soportando múltiples espacios de nombre

Un esquema XML puede describir elementos que pertenezcan a un espacio de nombre sencillo. Esto significa que un *Web Feature Service* no puede describir *features* de múltiples espacios de nombre en un esquema XML sencillo. Para superar esta limitación, un WFS puede generar un esquema XML el cual es un esquema *wrapper* que importa los esquemas de los *features* de diversos espacios de nombre en la petición. Por ejemplo, consideremos la siguiente petición:

```
<?xml version="1.0" ?>
<DescribeFeatureType
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ns01="http://www.server01.com/ns01"
   xmlns:ns02="http://www.server02.com/ns02"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <TypeName>ns01:TREESA_1M</TypeName>
   <TypeName>ns02:ROADL_1M</TypeName>
</DescribeFeatureType>
```

Un WFS puede generar la siguiente respuesta a esta petición:

```
<?xml version="1.0" ?>
<schema
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">

   <import namespace="http://www.server01.com/ns01"
           schemaLocation="http://www.myserver.com/wfs.cgi?
                           request=DescribeFeatureType&amp;typeName=ns01:TREESA_1M"/>

   <import namespace="http://www.server02.com/ns02"
           schemaLocation="http://www.yourserver.com/wfs.cgi?
                           request=DescribeFeatureType&amp;typeName=ns02:ROADL_1M"/>

</schema>
```

En este ejemplo, el WFS esta usando una petición *DescribeFeatureType* para obtener los esquemas de los *features* de los diversos espacios de nombre. Este es un ejemplo simple, pueden ser implementados otros métodos para obtener los esquemas (p.e.: hacer referencia a documentos esquemas estáticos).

## 8.4 Excepciones

En el caso de que un WFS encuentre un error al servir una petición *DescribeFeatureType*, debe generar una excepción como se describió en la sección 7.7.

## 8.5 Ejemplos

### Ejemplo 1

Consideremos *features* geográficos de tipos TREESA_1M y ROADL_1M que están definidos en una base de datos SQL. La descripción de estos tipos de *features* se reporta en la base de datos de la siguiente manera:

```
SQL> describe TREESA_1M
Name                      Null?    Type
------------------------  -------- -------------
WKB_GEOM                  NOT NULL LONG RAW
ID                                 NUMBER(10)
TREE_TYPE                          VARCHAR2(80)
```

```
SQL> describe ROADL_1M
Name                      Null?    Type
------------------------- -------- ------------
WKB_GEOM                  NOT NULL LONG RAW
DESIGNATION                        VARCHAR2(30)
SURFACE_TYPE                       VARCHAR2(30)
NLANES                             NUMBER(2)
```

En respuesta a la petición *DescribeFeatureType*:

```
<?xml version="1.0" ?>
<DescribeFeatureType
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.myserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <TypeName>myns:TREESA_1M</TypeName>
   <TypeName>myns:ROADL_1M</TypeName>
</DescribeFeatureType>
```

Un *web feature service* puede generar el siguiente esquema XML:

```
<?xml version="1.0" ?>
<schema
   targetNamespace="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:gml="http://www.opengis.net/gml"
   elementFormDefault="qualified" version="0.1">

   <import namespace="http://www.opengis.net/gml"
           schemaLocation="../gml/2.1/feature.xsd"/>

   <!-- ============================================
        define global elements
        ============================================ -->
   <element name="TREESA_1M"
            type="myns:TREESA_1M_Type"
            substitutionGroup="gml:_Feature"/>

   <element name="ROADL_1M"
            type="myns:ROADL_1M_Type"
            substitutionGroup="gml:_Feature"/>

   <!-- ============================================
        define complex types (classes)
        ============================================ -->
   <complexType name="TREESA_1M_Type">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="WKB_GEOM"
                        type="gml:PolygonPropertyType" nillable="false"/>
               <element name="ID" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="integer">
                        <totalDigits value="10"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="TREE_TYPE" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="80"/>
                     </restriction>
                  </simpleType>
               </element>
            </sequence>
         </extension>
      </complexContent>
```

```
   </complexType>

   <complexType name="ROADL_1M_Type">
      <complexContent>
         <extension base="gml:AbstractFeatureType">
            <sequence>
               <element name="WKB_GEOM"
                        type="gml:LineStringPropertyType"
                        nillable="false"/>
               <element name="DESIGNATION" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="string">

                        <maxLength value="30"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="SURFACE_TYPE" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="string">
                        <maxLength value="30"/>
                     </restriction>
                  </simpleType>
               </element>
               <element name="NLANES" nillable="true" minOccurs="0">
                  <simpleType>
                     <restriction base="integer">
                        <totalDigits value="2"/>
                     </restriction>
                  </simpleType>
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
</schema>
```

Usando este esquema de descripción, un cliente puede expresar el estado de una instancia de *feature* TREESA_1M y/o una instancia de *feature* ROADL_1M como se muestra en el siguiente ejemplo:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
   xmlns="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd
                       http://www.someserver.com/myns ex07.xsd">
   <gml:boundedBy>
      <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
         <gml:coordinates>-180.0,-90.0 180.0,90.0</gml:coordinates>
      </gml:Box>
   </gml:boundedBy>
   <gml:featureMember>
      <TREESA_1M>
         <WKB_GEOM>
            <gml:Polygon>
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                     <gml:coordinates decimal="." cs="," ts=" ">-120.000000,65.588264 -
120.003571,65.590782 -120.011292,65.590965 -120.022491,65.595215 -120.031212,65.592880
-120.019363,65.586121 -120.030350,65.585365 -120.045082,65.581848 -120.059540,65.584938
-120.067284,65.590500 -120.067284,65.595436 -120.067337,65.613441 -120.067337,65.613777
-120.060997,65.606346 -120.045517,65.605545 -120.022675,65.599777 -120.003975,65.601036
-120.000000,65.602081 -120.000000,65.602081 -120.000000,65.588264</gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </WKB_GEOM>
         <ID>0000000002</ID>
         <TREE_TYPE>Maple</TREE_TYPE>
```

```
      </TREESA_1M>
   </gml:featureMember>
   <gml:featureMember>
      <ROADL_1M>
         <WKB_GEOM>
            <gml:LineString srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates decimal="." cs="," ts=" ">-59.478340,-52.226578 -
59.484871,-52.223564 -59.488991,-52.198524 -59.485958,-52.169559 -59.480400,-52.152615
-59.465576,-52.141491 -59.462002,-52.136417 -59.447968,-52.127190 -59.422928,-52.120701
-59.411915,-52.117844 -59.397972,-52.116440 -59.371311,-52.121300</gml:coordinates>
            </gml:LineString>
         </WKB_GEOM>
         <DESIGNATION>HYW 401</DESIGNATION>
         <SURFACE_TYPE>ASPHALT</SURFACE_TYPE>
         <NLANES>12</NLANES>
      </ROADL_1M>
   </gml:featureMember>
</wfs:FeatureCollection>
```

**Ejemplo 2**

Este ejemplo describe un tipo colección, *People*, compuesto de instancias de *features* de un tipo de *feature Person*, que incluye una propiedad compleja *Address*.

En respuesta a una petición *DescribeFeatureType*:

```
<?xml version="1.0" ?>
<DescribeFeatureType
   version="1.0.0"
   service="WFS"
   outputFormat="XMLSCHEMA"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.myserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <TypeName>myns:People</TypeName>
</DescribeFeatureType>
```

Un *web feature service* puede generar un esquema XML como este:

```
<?xml version="1.0" ?>
<xsd:schema
   targetNamespace="http://www.someserver.com/myns"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" version="0.1">

   <xsd:import namespace="http://www.opengis.net/gml"
               schemaLocation="../gml/2.1/feature.xsd"/>

   <xsd:element name="Person"
                type="myns:PersonType"
                substitutionGroup="gml:_Feature"/>

   <xsd:complexType name="PersonType">
      <xsd:complexContent>
         <xsd:extension base="gml:AbstractFeatureType">
            <xsd:sequence>
               <xsd:element name="LastName" nillable="true">
                  <xsd:simpleType>
                     <xsd:restriction base="string">
                        <xsd:maxLength value="30"/>
                     </xsd:restriction>
                  </xsd:simpleType>
               </xsd:element>
               <xsd:element name="FirstName" nillable="true">
                  <xsd:simpleType>
                     <xsd:restriction base="string">
```

```
                        <xsd:maxLength value="10"/>
                     </xsd:restriction>
                  </xsd:simpleType>
               </xsd:element>
               <xsd:element name="Age"
                            type="integer"
                            nillable="true"/>
               <xsd:element name="Sex"
                            type="string"/>
               <xsd:element name="Location"
                            type="gml:PointPropertyType"
                            nillable="true"/>
               <xsd:element name="Address"
                            type="myns:AddressType"
                            nillable="true"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="AddressType">
      <xsd:sequence>
         <xsd:element name="StreetName" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="30"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="StreetNumber" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="10"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="City" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="30"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="Province" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="30"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="PostalCode" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="15"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="Country" nillable="true">
            <xsd:simpleType>
               <xsd:restriction base="string">
                  <xsd:maxLength value="30"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

Una instancia de ejemplo que sea válido contra este esquema puede ser:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
xmlns="http://www.someserver.com/myns"
```

```
 xmlns:myns="http://www.someserver.com/myns"
 xmlns:wfs="http://www.opengis.net/wfs"
 xmlns:gml="http://www.opengis.net/gml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd
                     http://www.someserver.com/myns ex10.xsd">
    <gml:boundedBy>
       <gml:Box>
          <gml:coord>
             <gml:X>10</gml:X>
             <gml:Y>10</gml:Y>
          </gml:coord>
          <gml:coord>
             <gml:X>20</gml:X>
             <gml:Y>20</gml:Y>
          </gml:coord>
       </gml:Box>
    </gml:boundedBy>
    <gml:featureMember>
       <Person>
          <myns:LastName>Smith</myns:LastName>
          <myns:FirstName>Fred</myns:FirstName>
          <myns:Age>35</myns:Age>
          <myns:Sex>Male</myns:Sex>
          <myns:Location>
             <gml:Point><gml:coordinates>15,15</gml:coordinates></gml:Point>
          </myns:Location>
          <myns:Address>
             <myns:StreetName>Main St.</myns:StreetName>
             <myns:StreetNumber>5</myns:StreetNumber>
             <myns:City>SomeCity</myns:City>
             <myns:Province>SomeProvince</myns:Province>
             <myns:PostalCode>X1X 1X1</myns:PostalCode>
             <myns:Country>Canada</myns:Country>
          </myns:Address>
       </Person>
    </gml:featureMember>
 </wfs:FeatureCollection>
```

## 9 Operación *GetFeature*

### 9.1 Introducción

La operación *GetFeature* permite recuperar *features* de un *web feature service*. Una petición *GetFeature* es procesada por un WFS y un documento XML, el cual contiene el resultado, y es regresado al cliente.

### 9.2 Petición

La codificación XML para una petición *GetFeature* está definida por el siguiente fragmento de esquema XML:

```
<xsd:element name="GetFeature" type="wfs:GetFeatureType"/>
<xsd:complexType name="GetFeatureType">
   <xsd:sequence>
      <xsd:element ref="wfs:Query" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
   <xsd:attribute name="handle"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="outputFormat"
                  type="xsd:string" use="optional" default="GML2"/>
   </xsd:attribute>
   <xsd:attribute name="maxFeatures" type="xsd:positiveInteger"
                  use="optional"/>
</xsd:complexType>
```

```
<xsd:element name="Query" type="wfs:QueryType"/>
<xsd:complexType name="QueryType">
  <xsd:sequence>
    <xsd:element ref="ogc:PropertyName" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="handle"
                 type="xsd:string" use="optional"/>
  <xsd:attribute name="typeName"
                 type="xsd:QName" use="required"/>
  <xsd:attribute name="featureVersion"
                 type="xsd:string" use="optional"/>
</xsd:complexType>
```

El elemento <*GetFeature*> contiene uno o más elementos <*Query*>, cada uno de los cuales contiene la descripción de una petición. El resultado de todas las peticiones contenidas en una solicitud *GetFeature* son concatenadas para producir el conjunto respuesta.

El atributo *outputFormat* define el formato que se va a usar para generar el conjunto respuesta. El valor por *deafault* es GML2 indicando que se debe usar GML. También es posible usar los formatos de vendedores específicos (incluyendo formatos no-XML y binarios), declarados en el documento de capacidades.

El atributo opcional *maxFeatures* puede ser usado para limitar el número de *features* que una petición *GetFeature* recupera. Una vez que el límite de *maxFeatures* es alcanzado, el conjunto respuesta es truncado en ese punto.

Cada petición individual empaquetada en una petición *GetFeature* es definida usando los elementos <*Query*>. El elemento <*Query*> define cuál tipo de *feature* cuestionar, cuáles propiedades recuperar y cuáles restricciones (espaciales y no espaciales) aplicar a esas propiedades.

El atributo *typeName* es usado para indicar el nombre del tipo de *feature* o clase que se solicitará.

El atributo *featureVersion* es incluido con el fin de satisfacer sistemas que soportan varias versiones de *feature*. Un valor de *ALL* indica que se pueden recuperar todas las versiones de un *feature*. De otra manera, se puede especificar un entero 'n', mediante el cual se regresará la $n^{va}$ versión de un *feature*. El número de versión empieza en 1, el cual es la versión más vieja. Si el valor de la versión es más grande que el número de versión más grande especificado, entonces se regresará la última versión. La acción *default* para una petición debe ser regresar la última versión. Los sistemas que no soportan varias versiones pueden ignorar el parámetro y regresar la única versión que manejan.

El elemento <*PropertyName*> es usado para enumerar las propiedades de los *features* que se seleccionarán durante una petición y cuyos valores se incluirán en la respuesta de una petición *GetFeature*. Una aplicación cliente puede determinar las propiedades de un *feature* haciendo primero una petición *DescribeFeatureType*. La operación *DescribeFeatureType* (sección 8) generará un esquema de aplicación GML definiendo el esquema de un tipo de *feature*. El cliente puede entonces seleccionar las propiedades a ser recuperadas. Además, el cliente puede determinar cuáles propiedades de *feature* son obligatorias y que deben ser recuperadas para que el WFS pueda generar una instancia del tipo de *feature* que será válido contra el esquema de aplicación GML generado. En el caso de que un WFS encuentre una petición que no selecciona todas las propiedades obligatorias de un *feature*, el WFS internamente aumentará la lista de nombres de propiedades para incluir los nombres de propiedades necesarios. Un cliente WFS debe así, estar preparado para tratar con una situación donde recibe más valores de propiedades que los que solicita.

Si no son especificados elementos <*PropertyName*>, entonces todas las propiedades del *feature* deben ser recuperadas.

El elemento <*Filter*> puede ser usado para definir restricciones sobre una petición. Se pueden especificar ambas restricciones espaciales y/o no espaciales como se describe en el documento OpenGIS© *Filter Encoding Implementation Specification*. Si no hay elementos <*Filter*> dentro del elemento <*Query*>, entonces la petición es libre de restricciones y todas las instancias de *features* deben ser recuperadas.

El elemento <*GetFeatureWithLock*> es funcionalmente similar al elemento <*GetFeature*>, excepto que indica a un *web feature service* que intente asegurar los *features* seleccionados, seguramente para actualizar/modificar los *features*.

### 9.3 Respuesta

El formato de la respuesta a una petición *GetFeature* es controlado por el atributo *outputFormat*. El valor por default para el atributo *outputFormat* debe ser GML2. Este indicará que un WFS debe generar un documento GML con el conjunto respuesta que cumple con el documento OpenGIS© *Geographic Markup Language (GML) Implementation Specification* y más específicamente, la salida debe ser válida contra el esquema de aplicación GML generado por la operación *DescribeFeatureType* (sección 8).

Cualquier documento GML generado por una implementación de WFS, en respuesta a una petición donde el *outputFormat* es GML2, debe hacer referencia al esquema de aplicación GML apropiado para que la salida pueda ser válida. Esto se puede lograr usando el atributo *schemaLocation*, como se define en el documento "*XML Schema Part 1: Structures"*. El atributo provee pautas en cuanto a la localización física de uno o más esquemas los cuales pueden ser usados para la evaluación de la validación local y de la validez del esquema. El valor del atributo *schemaLocation* contiene pares de valores. El primer miembro de cada par es el espacio de nombre para el cual el segundo miembro es la pauta que describe dónde encontrar un esquema apropiado. La localización física del esquema es especificada usando un URI.

El siguiente fragmento muestra el uso del atributo *schemaLocation* en el elemento raíz indicando la localización de un esquema XML que puede ser usado para la validación:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
   xmlns="http://www.opengis.net/myns"
   xmlns:myns="http://www.opengis.net/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
   xsi:schemaLocation="http://www.opengis.net/myns
                       http://www.someserver.com/wfs.cgi?
                       request=DescribeFeatureType&typename=TREESA_1M,ROADL_1M"> …
```

En esta instancia, el esquema correspondiente al espacio de nombre *myns* es generado dinámicamente haciendo antes una petición *DescribeFeatureType* al servidor que genera la salida, en la cual se solicita el esquema. Esta operación *DescribeFeatureService* solicita el esquema de los tipos de *feature* TREESA_1M y ROADL_1M, ambos en el espacio de nombre *myns*.

Depende de cada implementación de WFS arreglar que la salida GML haga las referencias *schemaLocation* apropiadas de tal forma que la salida pueda ser validada.

Para la petición <*GetFeatureWithLock*>, un WFS debe generar un resultado que incluya el identificador *lock*. El identificador *lock* es codificado usando el atributo *lockId* que está definido en el elemento <*wfs:FeatureCollection*>. El siguiente fragmento XML ilustra cómo incluir el atributo *lockId* en la respuesta a la operación:

```
<wfs:FeatureCollection lockId="00A01"… >
…
</wfs:FeatureCollection>
```

Los puntos suspensivos representan todos los otros componentes incluidos en la respuesta *GetFeatureWithLock* los cuales son idénticos a los componentes incluidos en la respuesta *GetFeature*.

### 9.4 Excepciones

En el caso de que un *web feature service* encuentre un error al servir una petición *GetFeature*, debe generar una excepción como se describió en la sección 7.7.

### 9.5 Ejemplos

Esta sección contiene numerosos ejemplos de peticiones *GetFeature*. Algunos ejemplos incluyen una salida de ejemplo.

**Ejemplo 1**

Este ejemplo recupera una instancia específica del tipo de *feature* INWATERA_1M identificado por el identificador de *feature* “INWATERA_1M.1234”.

```
<?xml version="1.0" ?>
<wfs:GetFeature
   service="WFS"
   version="1.0.0"
   outputFormat="GML2"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <wfs:Query typeName="myns:INWATERA_1M">
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1234"/>
      </ogc:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

**Ejemplo 2**

Este ejemplo recupera un subconjunto de propiedades del tipo de *feature* INWATER_1M. La instancia específica que es recuperada por la petición, se identifica por el identificador de *feature* “INWATERA_1M.1013”.

```
<?xml version="1.0" ?>
<wfs:GetFeature
   service="WFS"
   version="1.0.0"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <wfs:Query typeName="myns:INWATERA_1M">
      <ogc:PropertyName>myns:WKB_GEOM</ogc:PropertyName>
      <ogc:PropertyName>myns:TILE_ID</ogc:PropertyName>
      <ogc:PropertyName>myns:FAC_ID</ogc:PropertyName>
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1013"/>
      </ogc:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

**Ejemplo3**

En este ejemplo, todas las propiedades del tipo de *feature* INWATERA_1M son recuperadas por una lista enumerada de instancias de *features*. El elemento <*FeatureId*> es usado para identificar cada *feature* que será recuperado.

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:INWATERA_1M">
     <ogc:Filter>
        <ogc:FeatureId fid="INWATERA_1M.1013"/>
        <ogc:FeatureId fid="INWATERA_1M.1014"/>
        <ogc:FeatureId fid="INWATERA_1M.1015"/>
      </ogc:Filter>
   </Query>
</GetFeature>
```

**Ejemplo 4**

Este ejemplo es similar al ejemplo anterior excepto que en este caso sólo son recuperadas algunas de las propiedades de un conjunto enumerado de *features*. El elemento <*PropertyName*> es usado para listar las propiedades que serán recuperadas.

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:INWATERA_1M">
      <ogc:PropertyName>myns:WKB_GEOM</ogc:PropertyName>
      <ogc:PropertyName>myns:TILE_ID</ogc:PropertyName>
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1013"/>
         <ogc:FeatureId fid="INWATERA_1M.1014"/>
         <ogc:FeatureId fid="INWATERA_1M.1015"/>
      </ogc:Filter>
   </Query>
</GetFeature>
```

**Ejemplo 5**

Selecciona todas las instancias del tipo de *feature* INWATERA_1M hasta un máximo de 10,000 *features*.

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   maxFeatures="10000"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:INWATERA_1M"/>
</GetFeature>
```

**Ejemplo 6**

La siguiente petición libre de restricciones recupera todas las instancias de un conjunto enumerado de tipo de *features*. Hay que notar que no todos los tipos de *features* están en el mismo espacio de nombre.

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:yourns="http://demo.cubewerx.com/yourns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:INWATERA_1M"/>
   <Query typeName="myns:BUILTUPA_1M"/>
   <Query typeName="yourns:ROADL_1M"/>
</GetFeature>
```

**Ejemplo 7**

El siguiente ejemplo selecciona la geometría y la profundidad del *feature HYDROGRAPHY* para el área de *Grand Banks*. *Grand Banks* está limitado por el siguiente *bounding box*: [-57.9118,46.2023,-46.6873,51.8145].

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   handle="Query01"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:HYDROGRAPHY">
      <ogc:PropertyName>myns:GEOTEMP</ogc:PropertyName>
      <ogc:PropertyName>myns:DEPTH</ogc:PropertyName>
      <ogc:Filter>
         <ogc:Not>
            <ogc:Disjoint>
               <ogc:PropertyName>myns:GEOTEMP</ogc:PropertyName>
               <gml:Box>
               <gml:coordinates>-57.9118,46.2023 -46.6873,51.8145</gml:coordinates>
               </gml:Box>
            </ogc:Disjoint>
         </ogc:Not>
      </ogc:Filter>
   </Query>
</GetFeature>
```

La salida de tal petición puede ser:

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.someserver.com/myns HYDROGRAPHY.xsd
                       http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <gml:boundedBy>
      <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
         <gml:coordinates>10,10 20,20</gml:coordinates>
      </gml:Box>
   </gml:boundedBy>
   <gml:featureMember>
      <HYDROGRAPHY fid="HYDROGRAPHY.450">
```

```
            <GEOTEMP>
               <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                  <gml:coordinates>10,10</gml:coordinates>
               </gml:Point>
            </GEOTEMP>
            <DEPTH>565</DEPTH>
         </HYDROGRAPHY>
      </gml:featureMember>
      <gml:featureMember>
         <HYDROGRAPHY fid="HYDROGRAPHY.450">
            <GEOTEMP>
               <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                  <gml:coordinates>10,11</gml:coordinates>
               </gml:Point>
            </GEOTEMP>
            <DEPTH>566</DEPTH>
         </HYDROGRAPHY>
      </gml:featureMember>
      <!—
      .
      . … more HYDROGRAPHY instances …
      .
      -->
   </wfs:FeatureCollection>
```

**Ejemplo 8**

Este ejemplo describe dos consultas que recuperan instancias de *ROADS* y de *RAILS* que caen dentro de una región sencilla de interés.

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   handle="Example Query"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd">
   <Query typeName="myns:ROADS">
      <ogc:PropertyName>myns:PATH</ogc:PropertyName>
      <ogc:PropertyName>myns:LANES</ogc:PropertyName>
      <ogc:PropertyName>myns:SURFACETYPE</ogc:PropertyName>
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>myns:PATH</ogc:PropertyName>
            <gml:Box>
               <gml:coordinates>50,40 100,60</gml:coordinates>
            </gml:Box>
         </ogc:Within>
      </ogc:Filter>
   </Query>
   <Query typeName="myns:RAILS">
      <ogc:PropertyName>myns:TRACK</ogc:PropertyName>
      <ogc:PropertyName>myns:GAUGE</ogc:PropertyName>
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>myns:TRACK</ogc:PropertyName>
            <gml:Box>
               <gml:coordinates>50,40 100,60</gml:coordinates>
            </gml:Box>
         </ogc:Within>
      </ogc:Filter>
   </Query>
</GetFeature>
```

Los resultados de cada consulta son concatenados para formar la colección de *features* de salida.

```
<?xml version="1.0" ?>
<wfs:FeatureCollection
```

```
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd
                       http://www.someserver.com/myns ROADSRAILS.xsd">
   <gml:boundedBy>
      <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
         <gml:coordinates>0,0 180,360</gml:coordinates>
      </gml:Box>
   </gml:boundedBy>
   <gml:featureMember>
      <ROADS fid="ROADS.100">
         <PATH>
            <gml:LineString gid="1"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>10,10 10,11 10,12 10,13</gml:coordinates>
            </gml:LineString>
         </PATH>
         <SURFACE_TYPE>ASPHALT</SURFACE_TYPE>
         <NLANES>4</NLANES>
      </ROADS>
   </gml:featureMember>
   <gml:featureMember>
      <ROADS fid="ROADS.105">
         <PATH>
            <gml:LineString gid="2"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>10,10 10,11 10,12</gml:coordinates>
            </gml:LineString>
         </PATH>
         <SURFACE_TYPE>GRAVEL</SURFACE_TYPE>
         <NLANES>2</NLANES>
      </ROADS>
   </gml:featureMember>
   <!--
   ... more ROADS features ....
   -->
   <gml:featureMember>
      <RAILS fid="RAILS.119">
         <TRACK>
            <gml:LineString gid="n"
                 srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>15,10 16,11 17,12</gml:coordinates>
            </gml:LineString>
         </TRACK>
         <GAUGE>24</GAUGE>
      </RAILS>
   </gml:featureMember>
   <!--
   ... more RAILS features ....
   -->
</wfs:FeatureCollection>
```

**Ejemplo 9**

Este ejemplo ilustra cómo las propiedades complejas de los *features* pueden ser referenciados usando expresiones Xpath. Consideremos el tipo de *feature Person* definido como:

```
<?xml version="1.0" ?>
<schema
   targetNamespace="http://www.opengis.net/myns"
   xmlns:myns="http://www.opengis.net/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="1.0">

   <import namespace="http://www.opengis.net/gml"
           schemaLocation="../gml/2.1/feature.xsd"/>
```

```
<element name="Person" type="myns:PersonType"
         substitutionGroup="gml:_Feature"/>
<complexType name="PersonType">
   <complexContent>
      <extension base="gml:AbstractFeatureType">
         <sequence>
            <element name="LastName" nillable="true">
               <simpleType>
                  <restriction base="string">
                     <maxLength value="30"/>
                  </restriction>
               </simpleType>
            </element>
            <element name="FirstName" nillable="true">
               <simpleType>
                  <restriction base="string">
                     <maxLength value="10"/>
                  </restriction>
               </simpleType>
            </element>
            <element name="Age" type="integer" nillable="true"/>
            <element name="Sex" type="string"/>
            <element name="Spouse">
               <complexType>
                  <attribute name="sin" type="xsd:anyURI" use="required" />
               </complexType>
            </element>
            <element name="Location"
                     type="gml:PointPropertyType"
                     nillable="true"/>
            <element name="Address" type="myns:AddressType" nillable="true"/>
         </sequence>
         <attribute name="sin" type="xsd:anyURI" use="required"/>
      </extension>
   </complexContent>
</complexType>

<complexType name="AddressType">
   <sequence>
      <element name="StreetName" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
      <element name="StreetNumber" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="10"/>
            </restriction>
         </simpleType>
      </element>
      <element name="City" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
      <element name="Province" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="30"/>
            </restriction>
         </simpleType>
      </element>
      <element name="PostalCode" nillable="true">
         <simpleType>
            <restriction base="string">
               <maxLength value="15"/>
            </restriction>
         </simpleType>
      </element>
```

```
            <element name="Country" nillable="true">
               <simpleType>
                  <restriction base="string">
                     <maxLength value="30"/>
                  </restriction>
               </simpleType>
            </element>
         </sequence>
      </complexType>
   </schema>
```

La propiedad *Address* es una propiedad compleja.

El siguiente ejemplo recupera el apellido de todas las personas que viven en el bloque 10,000 de "*Main St.*" En el pueblo de "*SomeTown*", que son del sexo femenino y que ganan más de $35,000. Hay que notar el uso de expresiones XPath en el predicado para hacer referencia a propiedades complejas.

```
<?xml version="1.0" ?>
<GetFeature
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd
                       http://www.someserver.com/myns Person.xsd">
   <Query typeName="Person">
      <ogc:PropertyName>myns:Person/myns:LastName</ogc:PropertyName>
      <ogc:Filter>
         <ogc:And>
            <ogc:And>
               <ogc:PropertyIsGreaterThanOrEqualTo>
<ogc:PropertyName>myns:Person/myns:Address/myns:StreetNumber</ogc:PropertyName>
                  <ogc:Literal>10000</ogc:Literal>
               </ogc:PropertyIsGreaterThanOrEqualTo>
               <ogc:PropertyIsLessThanOrEqualTo>
<ogc:PropertyName>myns:Person/myns:Address/myns:StreetNumber</ogc:PropertyName>
                  <ogc:Literal>10999</ogc:Literal>
               </ogc:PropertyIsLessThanOrEqualTo>
            </ogc:And>
            <ogc:And>
               <ogc:PropertyIsEqualTo>
<ogc:PropertyName>myns:Person/myns:Address/myns:StreetName</ogc:PropertyName>
                  <ogc:Literal>Main St.</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>
<ogc:PropertyName>myns:Person/myns:Address/myns:City</ogc:PropertyName>
                  <ogc:Literal>SomeTown</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>myns:Person/myns:Sex</ogc:PropertyName>
                  <ogc:Literal>Female</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsGreaterThan>
                  <ogc:PropertyName>myns:Person/myns:Salary</ogc:PropertyName>
                  <ogc:Literal>35000</ogc:Literal>
               </ogc:PropertyIsGreaterThan>
            </ogc:And>
         </ogc:And>
      </ogc:Filter>
   </Query>
</GetFeature>
```

# 10 Operación *LockFeature*

## 10.1 Introducción

Las conexiones web son intrínsecamente *stateless*[7]. Como consecuencia de esto, no se preserva la semántica de las transacciones serializables. Para entender este asunto, consideremos una operación de actualización.

Un cliente recupera una instancia de *feature*. El *feature* es entonces modificado en el lado del cliente, y mandado de vuelta a la base de datos mediante una petición *Transaction* para actualizarlo. La habilidad de serializar se pierde ya que no hay garantía de que mientras el *feature* esta siendo modificado en el lado del cliente, otro cliente no vino antes y actualizó el mismo *feature* de la base de datos.

Una manera de asegurar la serialización es solicitar que el acceso a los datos sea hecho en una manera mutuamente exclusiva; esto es mientras una transacción accede a un dato, ninguna otra transacción puede modificar el mismo dato. Esto se puede lograr usando bloqueos que controlen el acceso a los datos.

El propósito de una operación *LockFeature* es exponer un mecanismo de bloqueo a largo plazo sobre los *features* para asegurar la consistencia. El bloqueo se considera de largo plazo porque la latencia de la red haría que el bloqueo del *feature* durara más tiempo que los bloqueos nativos sobre bases de datos comerciales.

La operación *LockFeature* es opcional y no necesita ser implementada para un WFS conforme a esta especificación. Si un WFS implementa la operación *LockFeature*, debe ser anunciado este hecho en el documento de capacidades como se describe en la sección 12.

## 10.2 Peticiones

### 10.2.1 Definición de esquema

La codificación XML de una petición *LockFeature* está definida por el siguiente fragmento de esquema XML:

```
<xsd:element name="LockFeature" type="wfs:LockFeatureType"/>
   <xsd:complexType name="LockFeatureType">
      <xsd:sequence>
         <xsd:element name="Lock" type="wfs:LockType" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="version"
                     type="xsd:string" use="required" fixed="1.0.0"/>
      <xsd:attribute name="service"
                     type="xsd:string" use="required" fixed="WFS"/>
      <xsd:attribute name="expiry"
                     type="xsd:positiveInteger" use="optional"/>
      <xsd:attribute name="lockAction"
                     type="wfs:AllSomeType" use="optional"/>
   </xsd:complexType>
   <xsd:complexType name="LockType">
      <xsd:sequence>
         <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="handle"
                     type="xsd:string" use="optional"/>
      <xsd:attribute name="typeName"
                     type="xsd:QName" use="required"/>
   </xsd:complexType>
```

[7] *Stateless* – Que no guarda el estado de las transacciones u operaciones.

El elemento <*LockFeature*> contiene uno o más elementos <*Lock*> que definen una operación de bloqueo sobre las instancias de *feature* de un tipo de *feature*.

El atributo *expiry* es usado para establecer cuánto tiempo debe un *web feature service* mantener un bloqueo sobre las instancias de *feature* en el caso de que nunca sea solicitada una transacción para liberar dicho bloqueo. El límite de *expiry* se especifica en minutos. Una vez que han pasado el número de minutos especificados, el *web feature service* puede liberar el bloqueo si es que existe. Cualquier otra transacción expedida contra ese bloqueo usando un identificador de bloqueo generado por el servicio, fallará. Esta especificación no restringe cuánto debe esperar un bloqueo si el atributo *expiry* no es especificado. Sin embargo, sería prudente para la implementación de un *web feature service* incluir métodos para detectar y liberar bloqueos que se han mantenido por un largo periodo de tiempo sin ninguna transacción que los libere.

El elemento <*Lock*> contiene un elemento simple <*Filter*> que se usa para definir un conjunto de instancias de *feature* del tipo de *feature* especificado que será bloqueado. Usando el elemento <*Filter*>, una o más instancias de *feature* pueden ser enumeradas usando sus identificadores; o un conjunto de *features* puede ser identificado especificando restricciones espaciales y no espaciales para la operación de bloqueo. El elemento <*Filter*> esta definido en el documento OpenGIS© *Filter Encoding Implementation Specification*.

El atributo opcional *lockAction* es usado para controlar cuántos *features* bloqueados son adquiridos. Una acción de bloqueo de *ALL* (todos) indica que un *web feature service* intenta adquirir un bloqueo sobre todas las instancias de *feature* solicitadas. Si todas las instancias de *feature* solicitadas no pueden ser bloqueadas, entonces la operación deberá fallar, y ninguna instancia de *feature* deberá permanecer bloqueada. Si la acción de bloqueo es establecida *SOME* (algunos), entonces un *web feature service* intentará bloquear tantas instancias de *feature* solicitadas como pueda. La acción de bloqueo default es *ALL*. La siguiente sección presenta una máquina de estado para la operación *LockFeature*.

### 10.2.2 Notación de la máquina de estado de UML

El enfoque al modelado dinámico usado, es el descrito en el manual de referencia UML. La técnica principal es la vista de la máquina de estado. En la figura 3 se muestra un resumen de la notación UML para los diagramas de estado:

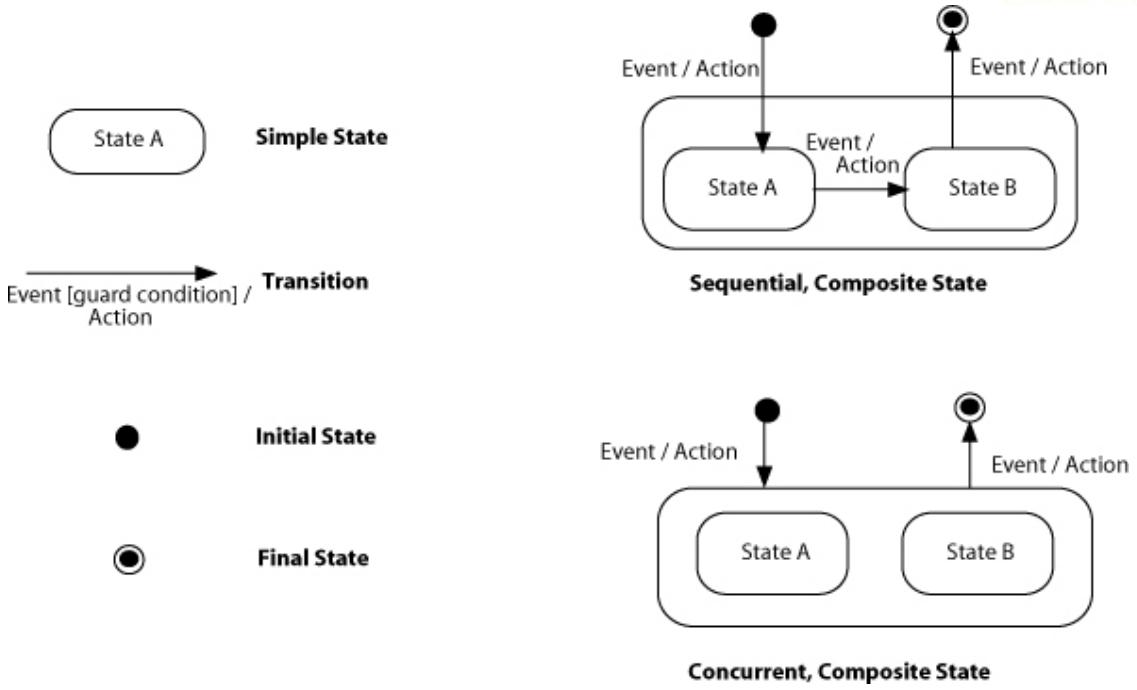


**Figura 3 – Resumen de la notación del diagrama de estado UML**

### 10.2.3 Máquina de estado para el bloqueo WFS (*locking)*

Esta sección define la máquina de estado del Estado Bloqueado para un servidor que provee la interface *web feature service*. El diagrama de estado muestra las transiciones permitidas entre los estados. Todas las demás transiciones de estado no son permitidas y son consideradas errores si se exhiben en el servidor.

Un servidor físico puede soportar más de un bloqueo. Cada uno de los bloqueos es independiente cuando es visto desde el servicio definido por la especificación WFS.

En el siguiente modelo de estado, una transición es provocada típicamente por una petición. Después del modelo de mensajes, una petición WFS se empareja con una respuesta WFS. Hay que observar que un par petición-respuesta no puede ser iniciado mientras esté activo. La petición puede ser cancelada por un comando a nivel HTTP.

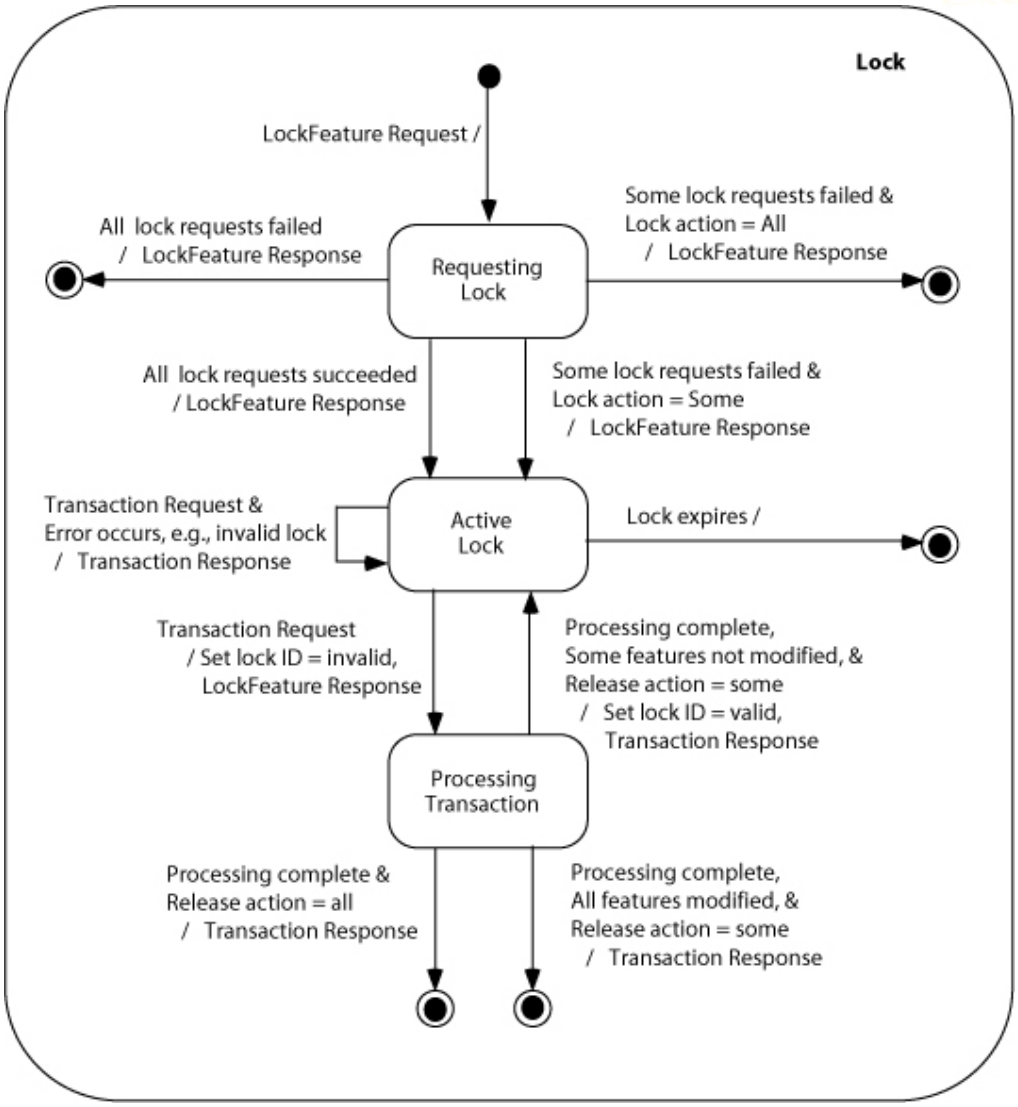


**Figura 4 – Diagrama de estado para una operación de bloqueo WFS**

### 10.3 Respuesta

La codificación XML de la respuesta a una petición *LockFeature* está definida por el siguiente fragmento de esquema:

```
<xsd:element name="WFS_LockFeatureResponse"
             type="wfs:WFS_LockFeatureResponseType"/>
   <!-- RESPONSE TYPES -->
   <xsd:complexType name="WFS_LockFeatureResponseType">
      <xsd:sequence>
         <xsd:element ref="wfs:LockId"/>
         <xsd:element name="FeaturesLocked"
                      type="wfs:FeaturesLockedType" minOccurs="0"/>
         <xsd:element name="FeaturesNotLocked"
                      type="wfs:FeaturesNotLockedType" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="FeaturesLockedType">
     <xsd:sequence maxOccurs="unbounded">
```

```
      <xsd:element ref="ogc:FeatureId"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="FeaturesNotLockedType">
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element ref="ogc:FeatureId"/>
    </xsd:sequence>
  </xsd:complexType>
```

En respuesta a una petición *LockFeature*, un *web feature service* debe generar un documento XML. Este documento contendrá un identificador de bloqueo que una aplicación cliente puede usar en operaciones WFS subsecuentes para operar sobre el conjunto de instancias de *features*. La respuesta puede también contener los elementos opcionales *<FeatureLocked>* y *<FeatureNotLocked>* dependiendo del valor del atributo *lockAction*.

Si la acción de bloqueo es especificada como *SOME*, entonces el elemento *<WFS_LockFeatureResponse>* debe contener los elementos *<FeatureLocked>* y *<FeatureNotLocked>*. El elemento *<FeatureLocked>* deberá listar los identifiadores de *feature* de todas las instancias de *feature* que fueron bloqueadas por la petición *LockFeature*. El elemento *<FeatureNotLocked>* deberá contener una lista de identificadores de *feature* para las instancias de *feature* que no pudieron ser bloqueadas por el *web feature service* (posiblemente porque ya estaban bloqueadas por alguien más).

No se hace ninguna suposición acerca del formato del identificador de bloqueo. El único requisito es que debe poder ser expresado en el conjunto de caracteres de la petición de la transacción.

### 10.4 Excepciones

Si un WFS no implementa la operación *LockFeature* entonces deberá generar una excepción, indicando que la operación no es soportada, en el caso de que se encontrara con tal petición.

En el caso de que un *web feature service* sí soporte la operación *LockFeature* y encuentre un error al servir la petición, deberá generar una excepción como se describió en la sección 7.7.

### 10.5 Ejemplos

#### Ejemplo 1

Bloquear un conjunto de *features* enumerados. El WFS es, en este caso, solicitado a intentar bloquear tantos *features* como pueda.

**Petición:**

```
<?xml version="1.0" ?>
<LockFeature
   version="1.0.0"
   service="WFS"
   lockAction="SOME"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <Lock typeName="myns:INWATERA_1M">
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1013"/>
         <ogc:FeatureId fid="INWATERA_1M.1014"/>
         <ogc:FeatureId fid="INWATERA_1M.1015"/>
         <ogc:FeatureId fid="INWATERA_1M.1016"/>
         <ogc:FeatureId fid="INWATERA_1M.1017"/>
      </ogc:Filter>
   </Lock>
```

```
</LockFeature>
```

**Respuesta ejemplo:**

```
<?xml version="1.0" ?>
<WFS_LockFeatureResponse
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <LockId>1</LockId>
   <FeaturesLocked>
      <ogc:FeatureId fid="INWATERA_1M.1013"/>
      <ogc:FeatureId fid="INWATERA_1M.1014"/>
      <ogc:FeatureId fid="INWATERA_1M.1016"/>
      <ogc:FeatureId fid="INWATERA_1M.1017"/>
   </FeaturesLocked>
   <FeaturesNotLocked>
      <ogc:FeatureId fid="INWATERA_1M.1015"/>
   </FeaturesNotLocked>
</WFS_LockFeatureResponse>
```

**Ejemplo 2**

Bloquear todas las instancias de *features* del tipo INWATERA_1M.

**Petición:**

```
<?xml version="1.0" ?>
<LockFeature
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <Lock typeName="myns:INWATERA_1M"/>
</LockFeature>
```

**Respuesta ejemplo:**

```
<?xml version="1.0" ?>
<WFS_LockFeatureResponse
   xmlns="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <LockId>2</LockId>
</WFS_LockFeatureResponse>
```

**Ejemplo 3**

En este ejemplo la expresión <*Filter*> con una restricción espacial es usada para identificar el conjunto de instancias de *feature* que serán bloqueadas.

**Petición:**

```
<?xml version="1.0" ?>
<LockFeature
   version="1.0.0"
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <Lock handle="Lock1" typeName="myns:INWATERA_1M">
      <ogc:Filter>
```

```
            <ogc:Within>
               <ogc:PropertyName>myns:WKB_GEOM</ogc:PropertyName>
               <gml:Polygon gid="1"
                    srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                  <gml:outerBoundaryIs>
                     <gml:LinearRing>
                     <gml:coordinates>-95.7,38.1 -97.8,38.2 ...</gml:coordinates>
                     </gml:LinearRing>
                  </gml:outerBoundaryIs>
               </gml:Polygon>
            </ogc:Within>
         </ogc:Filter>
      </Lock>
   </LockFeature>
```

**Respuesta ejemplo:**

```
   <?xml version="1.0" ?>
   <WFS_LockFeatureResponse
      xmlns="http://www.opengis.net/wfs"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
      <LockId>A1014375BD</LockId>
   </WFS_LockFeatureResponse>
```

**Ejemplo 4**

Este ejemplo bloquea los *features* del tipo BUILTUPA_1M e INWATERA_1M. El bloqueo con la etiqueta *LOCK1* bloquea todos los *features* dentro de la ventana definida. EL bloqueo con la etiqueta *LOCK2* bloquea los features INWATERA_1M.1212, INWATERA_1M.1213 e INWATERA_1M.10.

**Petición:**

```
   <LockFeature
      version="1.0.0"
      service="WFS"
      expiry="4"
      lockAction="SOME"
      xmlns="http://www.opengis.net/wfs"
      xmlns:ogc="http://www.opengis.net/ogc"
      xmlns:gml="http://www.opengis.net/gml"
      xmlns:myns="http://www.someserver.com/myns"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
      <Lock handle="LOCK1" typeName="myns:BUILTUPA_1M">
         <ogc:Filter>
            <ogc:Within>
               <ogc:PropertyName>BUILTUPA_1M/WKB_GEOM</ogc:PropertyName>
               <gml:Polygon gid="1"
                  srsName="http://www.opengis.net/gml/epsg.xml#4326">
                  <gml:outerBoundaryIs>
                     <gml:LinearRing>
                     <gml:coordinates>-95.7,38.1 -97.8,38.2 ...</gml:coordinates>
                     </gml:LinearRing>
                  </gml:outerBoundaryIs>
               </gml:Polygon>
            </ogc:Within>
         </ogc:Filter>
      </Lock>
      <Lock handle="LOCK2" typeName="myns:INWATERA_1M">
         <ogc:Filter>
            <ogc:FeatureId fid="INWATERA_1M.1212"/>
            <ogc:FeatureId fid="INWATERA_1M.1213"/>
            <ogc:FeatureId fid="INWATERA_1M.10"/>
         </ogc:Filter>
      </Lock>
   </LockFeature>
```

**Respuesta ejemplo:**

```
<?xml version="1.0" ?>
<WFS_LockFeatureResponse
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <LockId>LOCK1A</LockId>
   <FeaturesLocked>
      <ogc:FeatureId fid="BUILTUPA_1M.1" />
      <ogc:FeatureId fid="BUILTUPA_1M.10" />
      <ogc:FeatureId fid="BUILTUPA_1M.34" />
      <ogc:FeatureId fid="BUILTUPA_1M.786" />
      <ogc:FeatureId fid="BUILTUPA_1M.3" />
      <ogc:FeatureId fid="BUILTUPA_1M.13" />
      <ogc:FeatureId fid="BUILTUPA_1M.47563" />
      <ogc:FeatureId fid="INWATERA_1M.1212" />
      <ogc:FeatureId fid="INWATERA_1M.1213" />
      <ogc:FeatureId fid="INWATERA_1M.10" />
   </FeaturesLocked>
</WFS_LockFeatureResponse>
```

## 11 Operación *Transaction*

### 11.1 Introducción

La operación *Transaction* es usada para describir las operaciones de transformación de datos que se aplican a instancias de *features* accesibles mediante la *web*. Un *web feature service* puede procesar operaciones *Transaction* directamente o traducirla posiblemente al lenguaje de la base de datos a la cual está conectada y entonces hacer que la base de datos ejecute la transacción. Cuando ha sido completada una transacción, un *web feature service* generará un documento de respuesta XML indicando el status de la transacción.

La operación *Transaction* es opcional y una implementación de WFS no necesita soportarla conforme a esta especificación. Si la operación *Transaction* es soportada entonces este hecho debe anunciarse en el documento de capacidades como se describe en la sección 12.

### 11.2 Petición

#### 11.2.1 Definición de esquema

La codificación XML de una petición *Transaction* está definida por le siguiente fragmento de esquema XML:

```
<xsd:element name="Transaction" type="wfs:TransactionType"/>
   <xsd:complexType name="TransactionType">
      <xsd:sequence>
         <xsd:element ref="wfs:LockId" minOccurs="0"/>
         <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="wfs:Insert"/>
            <xsd:element ref="wfs:Update"/>
            <xsd:element ref="wfs:Delete"/>
            <xsd:element ref="wfs:Native"/>
         </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="version"
                     type="xsd:string" use="required" fixed="1.0.0"/>
      <xsd:attribute name="service"
                     type="xsd:string" use="required" fixed="WFS"/>
      <xsd:attribute name="handle"
                     type="xsd:string" use="optional"/>
      <xsd:attribute name="releaseAction"
                     type="wfs:AllSomeType" use="optional"/>
   </xsd:complexType>
```

```
<xsd:element name="LockId" type="xsd:string"/>
<xsd:element name="Insert" type="wfs:InsertElementType"/>
<xsd:complexType name="InsertElementType">
   <xsd:sequence>
      <xsd:element ref="gml:_Feature" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:element name="Update" type="wfs:UpdateElementType"/>
<xsd:complexType name="UpdateElementType">
   <xsd:sequence>
      <xsd:element ref="wfs:Property" maxOccurs="unbounded"/>
      <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="Delete" type="wfs:DeleteElementType"/>
<xsd:complexType name="DeleteElementType">
   <xsd:sequence>
      <xsd:element ref="ogc:Filter" minOccurs="1" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
</xsd:complexType>

<xsd:element name="Property" type="wfs:PropertyType"/>
<xsd:complexType name="PropertyType">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:QName"/>
      <xsd:element name="Value"/>
   </xsd:sequence>
</xsd:complexType>
```

### 11.2.2 Descripción de atributos

Como se describe en la sección 7.8, el atributo *handle* puede ser usado para asignar un nombre mnemónico a los elementos con los cuales está asociado, con el fin de hacer un reporte de errores más significativo para la aplicación cliente. En el caso de que se encuentre un error, el atributo *handle* puede ser usado por un *web feature service* para localizar el error al generar un reporte de excepción. En caso de que no se haya especificado el atributo *handle*, un *web feature service* puede intentar reportar la localización de la excepción relativa a la petición *Transaction* actual usando números de líneas u otro mecanismo conveniente.

Asumiendo que una implementación de WFS soporta las operaciones opcionales *LockFeature* y/o *GetFeatureWithLock*, el atributo *releaseAction* es usado para controlar cómo los features bloqueados son tratados cuando es completada una petición de transacción. Un valor de *ALL* indica que los bloqueos sobre todas las instancias de *features* bloqueadas usando el *<LockId>* especificado, deberán ser liberadas cuando la transacción se complete, sin importar si se operó o no sobre una instancia particular de *feature* en el conjunto bloqueado. Un valor de *SOME* indica que sólo los bloqueos sobre instancias de *feature* modificadas por la transacción serán liberadas. Las otras instancias de *feature*, las que no se modificaron, deberán permanecer bloqueadas usando el mismo *<LockId>* así las transacciones siguientes pueden operar sobre aquellas instancias de *feature*. En el caso de que el *releaseAction* esté establecido en *SOME*, y el período de expiración fue especificado en el elemento *<LockFeature>* o en *<GetFeatureWithLock>* usando el atributo *expiry*, el contador de expiración debe ser iniciado en cero después de cada transacción a menos que se haya operado sobre todas las instancias de *feature* del conjunto bloqueado. El valor default del atributo *releaseAtion* es *ALL*.

Por ejemplo, si una aplicación cliente bloquea 20 instancias de *feature* y luego envía una petición de transacción que sólo opera sobre 10 de las instancias de *feature* bloqueadas, un *realeaseAction* de *SOME* significaría que las 10 instancias de *feature* sobrantes inalteradas deberán permanecer bloqueadas cuando la transacción termine. Las operaciones de transacción

siguientes pueden entonces ser enviadas por la aplicación cliente, usando el mismo identificador de bloque para modificar las 10 instancias de *feature* restantes.

### 11.2.3 Elemento <*Transaction*>

Un elemento <*Transaction*> puede contener cero o más elementos <*Insert*>, <*Update*>, ó <*Delete*> para describir las operaciones de crear, modificar o eliminar instancias de *feature*. Una petición <*Transaction*> vacía es válida pero inútil.

El elemento opcional <*LockId*> es usado para especificar que la transacción se aplicará al conjunto bloqueado previo de instancias de *feature*. La sección 10 presenta una completa descripción de un mecanismo de bloqueo de *features*. Si el WFS no soporta bloqueo de *features*, entonces el elemento <*LockId*> puede ser ignorado. Si un WFS soporta bloqueo y se especifica un identificador de bloqueo inválido en la transacción, entonces la transacción deberá fallar y el *web feature service* reportará el error como se describe en la sección 7.7.

Al final de una transacción, el *web feature service* debe aplicar semánticas de transacción apropiadas al sistema particular usado para almacenar persistentemente los *features*. Por ejemplo, si la base de datos es una RDBMS basada en SQL, entonces un comando *commit* se ejecutará al final de la transacción (o un *rollback* si la transacción falla). Cualquier bloqueo que haya mantenido el *web feature service* durante la transacción deberá ser liberado de acuerdo al valor del atributo *releaseAction* descrito anteriormente.

El elemento <*Native*> se define en la sección 7.5.

### 11.2.4 Elemento <*Insert*>

El elemento <*Insert*> se usa para crear nuevas instancias de *feature*. El estado inicial de un *feature* para ser creado es expresado usando GML y debe ser válido con respecto al esquema de aplicación GML generado en la operación *DescribeFeatureType* (sección 8). Se pueden agrupar múltiples elementos <*Insert*> en una sola petición *Transaction* y se pueden crear múltiples instancias de *feature* usando un solo elemento <*Insert*>.

En respuesta a una operación <*Insert*>, un *web feature service* deberá generar una lista de los nuevos identificadores de *feature* asignados a las nuevas instancias de *feature*. Los identificadores de *feature* deben ser presentados en el orden en el cual fueron encontradas las operaciones <*Insert*> en la petición *Transaction*.

**Ejemplo**

La siguiente transacción crea dos instancias del tipo de *feature* INWATERA_1M.

```
<?xml version="1.0"?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.someserver.com/myns
                       http://www.someserver.com/wfs/cwwfs.cgi?
                       request=describefeaturetype&amp;typename=INWATERA_1M.xsd
                       http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Insert>
      <INWATERA_1M>
         <WKB_GEOM>
            <gml:Polygon gid="1"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:outerBoundaryIs>
```

```
                  <gml:LinearRing>
                     <gml:coordinates>-98.54,24.26 ...</gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </WKB_GEOM>
         <ID>150</ID>
         <F_CODE>ABCDE</F_CODE>
         <HYC>152</HYC>
         <TILE_ID>250</TILE_ID>
         <FAC_ID>111</FAC_ID>
      </INWATERA_1M>
      <INWATERA_1M>
         <WKB_GEOM>
            <gml:Polygon gid="1"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
                     <gml:coordinates>-99.99,22.22 ...</gml:coordinates>
                  </gml:LinearRing>
               </gml:outerBoundaryIs>
            </gml:Polygon>
         </WKB_GEOM>
         <ID>111</ID>
         <F_CODE>FGHIJ</F_CODE>
         <HYC>222</HYC>
         <TILE_ID>333</TILE_ID>
         <FAC_ID>444</FAC_ID>
      </INWATERA_1M>
   </wfs:Insert>
</wfs:Transaction>
```

Se asume que la referencia al esquema INWATERA_1M.xsd se creará usando la operación *DescribeFeatureType* (sección 8). En este ejemplo, el documento es referido estáticamente, pero habría podido apenas ser referido dinámicamente.

### 11.2.5 Elemento *<Update>*

El elemento <*Update*> describe una operación de actualización que será aplicada a un *feature* o a un conjunto de *features* de un sólo tipo de *feature*. Se pueden contener múltiples operaciones <*Update*> en una sola petición *Transaction*.

Un elemento <*Update*> contiene uno o más elementos <*Property*> que especifican el nombre y el valor de reemplazo para una propiedad que pertenece a un tipo de *feature* especificado usando el atributo obligatorio *typeName*. Un elemento <*Property*> contiene un elemento <*Name*> que, en cambio, contiene el nombre de la propiedad que será modificada y un elemento opcional <*Value*> que contiene el valor de reemplazo para la propiedad nombrada. La omisión del elemento <*Value*> significa que a la propiedad se le asignará el valor *NULL*. En el caso de que la propiedad no sea *nillable* (es decir, que no pueda ser nula), un WFS deberá generar una excepción indicando que el valor *NULL* no está permitido.

El alcance del elemento <*Update*> esta restringido por el elemento <*Filter*>. El elemento <*Filter*> puede ser usado para limitar el alcance de una operación de actualización a un conjunto enumerado de *features* o a un conjunto de *features* definido por restricciones espaciales y no espaciales.

La definición completa del elemento <*Filter*> está descrita en el documento OpenGIS© *Filter Encoding Implementation Specification.*

**Ejemplo**

El siguiente ejemplo actualiza la propiedad POPULATION del *feature* identificado por el identificador BUILTUPA_1M.1013

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Update typeName="BUILTUPA_1M">
      <wfs:Property>
         <wfs:Name>POPULATION</wfs:Name>
         <wfs:Value>4070000</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:FeatureId fid="BUILTUPA_1M.10131"/>
      </ogc:Filter>
   </wfs:Update>
</wfs:Transaction>
```

**Ejemplo**

Actualiza la propiedad POPULATION_TYPE de un conjunto enumerado de *features*, En este ejemplo, los *features* identificados por los identificadores:

```
BUILTUPA_1M.1013
BUILTUPA_1M.34
BUILTUPA_1M.24256
```

establecen sus atributos POPULATION_TYPE con al valor "*CITY*".

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Update typeName="BUILTUPA_1M">
      <wfs:Property>
         <wfs:Name>POPULATION_TYPE</wfs:Name>
         <wfs:Value>CITY</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:FeatureId fid="BUILTUPA_1M.1013"/>
         <ogc:FeatureId fid="BUILTUPA_1M.34"/>
         <ogc:FeatureId fid="BUILTUPA_1M.24256"/>
      </ogc:Filter>
   </wfs:Update>
</wfs:Transaction>
```

**Ejemplo**

Actualiza la propiedad NAME de un conjunto enumerado de *features*, y actualiza la propiedad FAC_ID de otro conjunto de *features* definido restringiendo el valor de la propiedad TILE_ID a valores más grandes de 1000.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Update typeName="myns:BUILTUPA_1M">
      <wfs:Property>
```

```
            <wfs:Name>myns:NAME</wfs:Name>
            <wfs:Value>somestring</wfs:Value>
         </wfs:Property>
         <ogc:Filter>
            <ogc:FeatureId fid="BUILTUPA_1M.1013"/>
            <ogc:FeatureId fid="BUILTUPA_1M.34"/>
            <ogc:FeatureId fid="BUILTUPA_1M.24256"/>
         </ogc:Filter>
      </wfs:Update>
      <wfs:Update typeName="myns:BUILTUPA_1M">
         <wfs:Property>
            <wfs:Name>myns:FAC_ID</wfs:Name>
            <wfs:Value>100</wfs:Value>
         </wfs:Property>
         <ogc:Filter>
            <ogc:PropertyIsGreaterThan>
               <ogc:PropertyName>BUILTUPA_1M/TILE_ID</ogc:PropertyName>
               <ogc:Literal>1000</ogc:Literal>
            </ogc:PropertyIsGreaterThan>
         </ogc:Filter>
      </wfs:Update>
   </wfs:Transaction>
```

**Ejemplo**

Este ejemplo actualiza dos clases de *features*, OCEANSA_1M y TREESA_1M. Todos los *features* de OCEANSA_1M con una DEPTH (profundidad) mayor a 2400 mts. son actualizados y el *feature* TREESA_1M.1010 también es actualizado.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Update typeName="myns:OCEANSA_1M">
      <wfs:Property>
         <wfs:Name>myns:DEPTH</wfs:Name>
         <wfs:Value>2400</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:PropertyIsGreaterThan>
            <ogc:PropertyName>OCEANSA_1M.DEPTH</ogc:PropertyName>
            <ogc:Literal>2400</ogc:Literal>
         </ogc:PropertyIsGreaterThan>
      </ogc:Filter>
   </wfs:Update>
   <wfs:Update typeName="myns:TREESA_1M">
      <wfs:Property>
         <wfs:Name>myns:TREETYPE</wfs:Name>
         <wfs:Value>CONIFEROUS</wfs:Value>
      </wfs:Property>
      <ogc:Filter>
         <ogc:FeatureId fid="TREESA_1M.1010"/>
      </ogc:Filter>
   </wfs:Update>
</wfs:Transaction>
```

### 11.2.6 Elemento *<Delete>*

El elemento *<Delete>* es usado para indicar que una o más de las instancias de *feature* deben ser eliminadas. El alcance de la operación de borrado está restringido por el elemento *<Filter>* como se describe en el documento OpenGIS© *Filter Encoding Implementation Specification.*

**Ejemplo**

Eliminar un *feature* simple.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Delete typeName="INWATERA_1M">
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1013"/>
      </ogc:Filter>
   </wfs:Delete>
</wfs:Transaction>
```

**Ejemplo**

Este ejemplo elimina un conjunto enumerado de instancias de *feature*.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Delete typeName="myns:INWATERA_1M">
      <ogc:Filter>
         <ogc:FeatureId fid="INWATERA_1M.1013"/>
         <ogc:FeatureId fid="INWATERA_1M.10"/>
         <ogc:FeatureId fid="INWATERA_1M.13"/>
         <ogc:FeatureId fid="INWATERA_1M.140"/>
         <ogc:FeatureId fid="INWATERA_1M.5001"/>
         <ogc:FeatureId fid="INWATERA_1M.2001"/>
      </ogc:Filter>
   </wfs:Delete>
</wfs:Transaction>
```

**Ejemplo**

Este ejemplo elimina el conjunto de instancias del tipo de *feature* INWATERA_1M que está dentro de la región definida por un polígono especificado en el predicado. El elemento <*Filter*> es usado para restringir el alcance de la operación, y GML es usado para expresar la geometría del polígono.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:Delete typeName="INWATERA_1M">
      <ogc:Filter>
         <ogc:Within>
            <ogc:PropertyName>WKB_GEOM</ogc:PropertyName>
            <gml:Polygon gid="pp9"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:outerBoundaryIs>
                  <gml:LinearRing>
```

```
                    <gml:coordinates>-95.7,38.1 -97.8,38.2 ...</gml:coordinates>
                    </gml:LinearRing>
                 </gml:outerBoundaryIs>
              </gml:Polygon>
           </ogc:Within>
        </ogc:Filter>
     </wfs:Delete>
  </wfs:Transaction>
```

### 11.3 Respuesta

En respuesta a una petición de transacción, un *web feature service* debe generar un documento XML indicando el status de terminación de la transacción. Además, si la petición de transacción incluye operaciones <*Insert*> entonces el *web feature service* debe reportar los identificadores de *feature* de todos los nuevos *features* creados. En el caso de que la transacción falle al ejecutarse, el *web feature service* debe también indicar esto en la respuesta.

La codificación XML de la respuesta a una transacción WFS está definida por el siguiente fragmento de esquema XML:

```
<xsd:element name="WFS_TransactionResponse"
               type="wfs:WFS_TransactionResponseType"/>
   <xsd:complexType name="WFS_TransactionResponseType">
      <xsd:sequence>
         <xsd:element name="InsertResult"
                      type="wfs:InsertResultType"
                      minOccurs="0" maxOccurs="unbounded"/>
         <xsd:element name="TransactionResult"
                      type="wfs:TransactionResultType"/>
      </xsd:sequence>
      <xsd:attribute name="version"
                     type="xsd:string" use="required" fixed="1.0.0"/>
   </xsd:complexType>
   <xsd:complexType name="TransactionResultType">
      <xsd:sequence>
         <xsd:element name="Status" type="wfs:StatusType"/>
         <xsd:element name="Locator" type="xsd:string" minOccurs="0"/>
         <xsd:element name="Message" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   </xsd:complexType>
   <xsd:complexType name="InsertResultType">
      <xsd:sequence>
         <xsd:element ref="ogc:FeatureId" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   </xsd:complexType>
   <xsd:complexType name="StatusType">
      <xsd:choice>
         <xsd:element ref="wfs:SUCCESS"/>
         <xsd:element ref="wfs:FAILED"/>
         <xsd:element ref="wfs:PARTIAL"/>
      </xsd:choice>
   </xsd:complexType>
   <xsd:element name="SUCCESS" type="wfs:EmptyType"/>
   <xsd:element name="FAILED" type="wfs:EmptyType"/>
   <xsd:element name="PARTIAL" type="wfs:EmptyType"/>
```

El elemento <*WFS_TransactionResponse*> contiene cero o más elementos <*InsertResult*> y un elemento <*TransactionResult*>.

El elemento <*InsertResult*> contiene uno o más identificadores de *feature* de las nuevas instancias de *features* creadas. Se reporta un elemento <*InsertResult*> por cada elemento <*Insert*> en la petición. Los resultados de la inserción se reportan en el orden en el cual las operaciones <*Insert*> fueron encontradas en el elemento <*Transaction*>. Adicionalmente, pueden ser correlacionados si se especificó el atributo *handle*.

El resultado total de una petición de transacción se especifica usando el elemento <*TransactionResult*>. El elemento <*TransactionResult*> debe contener un elemento <*Status*> y puede contener elementos <*Locator*> y <*Message*>.

El elemento <*Status*> es usado para indicar el status de terminación de la transacción. La transacción termina con un status de:

- **SUCCESS** – La transacción se completó satisfactoriamente.
- **FAILED** – Se encontró una excepción mientras se procesaban uno o más elementos contenidos en la petición *Transaction*.
- **PARTIAL** – La transacción se concluyó parcialmente y los datos pueden estar en un estado inconsistente. Para sistemas que no soportan transacciones atómicas, este resultado es una posibilidad distinta.

En el caso de que una petición de transacción falle, el elemento <*Locator*> puede ser usado para indicar qué parte de la transacción falló. Si el elemento en el cual ocurrió la falla es etiquetado usando un atributo *handle*, entonce un *web feature service* puede reportar su valor para localizar la falla. De otra manera, un *web feature service* puede intentar identificar la falla relativa al inicio de la petición de la transacción, posiblemente usando números de línea o algún otro mecanismo conveniente.

El elemento <*Message*> es usado para reportar cualquier mensaje de error.

**Ejemplo**

Consideremos una petición de transacción (etiquetada “TX01”) que crea un número de nuevas instancias de *feature*. Las instancias de *feature* son creadas usando tres elementos <*Insert*> etiquetados “STMT1”, “STMT2” y “STMT3”. Una respuesta típica a tal petición puede ser:

```
<?xml version="1.0" ?>
<wfs:TransactionResponse
   version="1.0.0"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:InsertResult handle="STMT1">
      <ogc:FeatureId fid="SOMEFEATURE.4567"/>
      <ogc:FeatureId fid="SOMEFEATURE.4568"/>
      <ogc:FeatureId fid="SOMEFEATURE.4569"/>
   </wfs:InsertResult>
   <wfs:InsertResult handle="STMT2">
      <ogc:FeatureId fid="FEATURE1.4569"/>
   </wfs:InsertResult>
   <wfs:InsertResult handle="STMT3">
      <ogc:FeatureId fid="FEATURE2.389345"/>
   </wfs:InsertResult>
   <wfs:TransactionResult handle="TX01">
      <wfs:Status>
         <wfs:SUCCESS/>
      </wfs:Status>
   </wfs:TransactionResult>
</wfs:TransactionResponse>
```

## 11.4 Excepciones

En el caso de que un *web feature service* encuentre un error al servir una petición *Transaction*, deberá generar una excepción como se describió en la sección 7.7.

En el caso de que un *web feature service* encuentre un error mientras procesa un elemento particular contenido en una petición <*Transaction*>, el *web feature service* debe mantener ese registro del resultado y reportar la falla en un elemento <*WFS_TransactionResponse*> (sección 11.3).

**Ejemplo**

En este ejemplo, la segunda declaración de la petición *Transaction* ha fallado. El WFS puede generar una respuesta como esta:

```
<?xml version="1.0" ?>
<WFS_TransactionResponse
   version="1.0.0"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">

   <TransactionResult handle="TX01">
      <Status><FAILED/></Status>
      <Locator>STMT2</Locator>
      <Message>ORA-00942: table or view does not exist</Message>
   </TransactionResult>
</WFS_TransactionResponse>
```

## 11.5 Ejemplos

Este ejemplo define una transacción compleja, etiquetada “Transaction 01”, que lleva acabo operaciones insertar, modificar y eliminar. Algunos de los tipos de *feature* incluyen propiedades complejas y se usan expresiones XPath en las expresiones de filtrado para hacer referencia inequívoca a las propiedades deseadas. Los comentarios contenidos en este ejemplo explican varias operaciones.

```
<?xml version="1.0" ?>
<wfs:Transaction
   version="1.0.0"
   service="WFS"
   handle="Transaction 01"
   xmlns="http://www.someserver.com/myns"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.someserver.com/myns
                       http://www.someserver.com/wfs/cwwfs.cgi?
                       request=DESCRIBEFEATURETYPE&amp;
                       typename=ELEVP_1M,ROADL_1M,BUILTUPA_1M
                       http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">

   <!-- Create a new instance of feature type ELEVP_1M -->
   <wfs:Insert handle="Statement 1">
      <ELEVP_1M>
         <ID>167928</ID>
         <F_CODE>CA</F_CODE>
         <ACC>2</ACC>
         <ELA>1</ELA>
         <ZV2>152</ZV2>
         <TILE_ID>250</TILE_ID>
         <END_ID>111</END_ID>
         <LOCATION>
            <gml:Point gid="e33"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>-98.5485,24.2633</gml:coordinates>
            </gml:Point>
         </LOCATION>
      </ELEVP_1M>
   </wfs:Insert>

   <!-- Create a new instance of feature type ROADL_1M
        which has complex properties SEGMENT and ROADTYPE. -->
   <wfs:Insert handle="ComplexInsert">
      <ROADL_1M>
         <NAME>Highway 401</NAME>
```

```
      <SEGMENT>
         <DESIGNATION>SEG_A41</DESIGNATION>
         <GEOMETRY>
            <gml:LineString gid="e3"
               srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
               <gml:coordinates>...</gml:coordinates>
            </gml:LineString>
         </GEOMETRY>
      </SEGMENT>
      <ROADTYPE>
         <SURFACE_TYPE>Asphalt</SURFACE_TYPE>
         <NLANES>12</NLANES>
         <GRADE>15</GRADE>
      </ROADTYPE>
   </ROADL_1M>
</wfs:Insert>

<!-- Update the designation of a particular range of segments
     which are now being collapsed into a single segment.  The
     The filter uses an XPath expression to reference the
     DESIGNATION property -->
<wfs:Update typeName="ROADL_1M">
   <wfs:Property>
      <wfs:Name>ROADL_1M/SEGMENT/DESIGNATION</wfs:Name>
      <wfs:Value>SEG_A60</wfs:Value>
   </wfs:Property>
   <ogc:Filter>
      <ogc:PropertyIsBetween>
         <ogc:PropertyName>ROADL_1M/SEGMENT/DESIGNATION</ogc:PropertyName>
         <ogc:LowerBoundary>
            <ogc:Literal>SEG_A60</ogc:Literal>
         </ogc:LowerBoundary>
         <ogc:UpperBoundary>
            <ogc:Literal>SEG_A69</ogc:Literal>
         </ogc:UpperBoundary>
      </ogc:PropertyIsBetween>
   </ogc:Filter>
</wfs:Update>

<!-- Create 2 feature instances of feature type BUILTUPA_1M -->
<wfs:Insert handle="Statement 2">
   <BUILTUPA_1M>
      <PLACEID>4070</PLACEID>
      <NAME>Toronto</NAME>
      <POPULATION>4000000</POPULATION>
      <BNDRY>
         <gml:Polygon gid="g3"
            srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
            <gml:outerBoundaryIs>
               <gml:LinearRing>
               <gml:coordinates>-95.7,38.1 -97.8,-38.2 ...</gml:coordinates>
               </gml:LinearRing>
            </gml:outerBoundaryIs>
         </gml:Polygon>
      </BNDRY>
   </BUILTUPA_1M>
   <BUILTUPA_1M>
      <PLACEID>1725</PLACEID>
      <NAME>Montreal</NAME>
      <POPULATION>2000000</POPULATION>
      <BNDRY>
         <gml:Polygon gid="e4"
            srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
            <gml:outerBoundaryIs>
               <gml:LinearRing>
               <gml:coordinates>-89.8,44.3 -89.9,44.4 ...</gml:coordinates>
               </gml:LinearRing>
            </gml:outerBoundaryIs>
         </gml:Polygon>
      </BNDRY>
   </BUILTUPA_1M>
</wfs:Insert>

<!-- Update the NAME property of the feature instance BUILTUPA_1M.1210 -->
```

```
    <wfs:Update typeName="BUILTUPA_1M">
       <wfs:Property>
          <wfs:Name>NAME</wfs:Name>
          <wfs:Value>SMALLVILLE</wfs:Value>
       </wfs:Property>
       <ogc:Filter>
          <ogc:FeatureId fid="BUILTUPA_1M.1210"/>
       </ogc:Filter>
    </wfs:Update>

    <!-- Update the geometry of the feature BUILTUPA_1M.1725. -->
    <wfs:Update typeName="BUILTUPA_1M">
       <wfs:Property>
          <wfs:Name>BNDRY</wfs:Name>
          <wfs:Value>
             <gml:Polygon gid="g5"
                srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                <gml:outerBoundaryIs>
                   <gml:LinearRing>
                   <gml:coordinates>-89.8,44.3 -89.9,44.4 ...</gml:coordinates>
                   </gml:LinearRing>
                </gml:outerBoundaryIs>
             </gml:Polygon>
          </wfs:Value>
       </wfs:Property>
       <ogc:Filter>
          <ogc:FeatureId fid="BUILTUPA_1M.1725"/>
       </ogc:Filter>
    </wfs:Update>

    <!-- Delete the feature instance BUILTUPA_1M.1013. -->
    <wfs:Delete typeName="BUILTUPA_1M">
       <ogc:Filter>
          <ogc:FeatureId fid="BUILTUPA_1M.1013"/>
       </ogc:Filter>
    </wfs:Delete>

    <!-- Delete all instances of the feature type
         BUILTUPA_1M where:
         1. the geometry is INSIDE a polygonal window
         2. where the POPULATION is between 100 and 2000 -->
    <wfs:Delete typeName="BUILTUPA_1M">
       <ogc:Filter>
          <ogc:And>
             <ogc:Within>
                <ogc:PropertyName>BUILTUPA_1M/BNDRY</ogc:PropertyName>
                <gml:Polygon gid="WINDOW"
                   srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                   <gml:outerBoundaryIs>
                      <gml:LinearRing>
                         <gml:coordinates>0,0 0,5 5,5 5,0 ...</gml:coordinates>
                      </gml:LinearRing>
                   </gml:outerBoundaryIs>
                </gml:Polygon>
             </ogc:Within>
             <ogc:And>
                <ogc:PropertyIsGreaterThanOrEqualTo>
                   <ogc:PropertyName>BUILTUPA_1M/POPULATION</ogc:PropertyName>
                   <ogc:Literal>100</ogc:Literal>
                </ogc:PropertyIsGreaterThanOrEqualTo>
                <ogc:PropertyIsLessThanOrEqualTo>
                   <ogc:PropertyName>BUILTUPA_1M/POPULATION</ogc:PropertyName>
                   <ogc:Literal>2000</ogc:Literal>
                </ogc:PropertyIsLessThanOrEqualTo>
             </ogc:And>
          </ogc:And>
       </ogc:Filter>
    </wfs:Delete>
 </wfs:Transaction>
```

En respuesta a la finalización satisfactoria de esta petición, un *web feature service* puede generar el siguiente documento de respuesta:

```
<?xml version="1.0" ?>
<wfs:TransactionResponse
   version="1.0.0"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-transaction.xsd">
   <wfs:InsertResult handle="Statement 1">
      <ogc:FeatureId fid="ELEVP_1M.1001"/>
   </wfs:InsertResult>
   <wfs:InsertResult handle="ComplexInsert">
      <ogc:FeatureId fid="ROADL_1M.1553"/>
   </wfs:InsertResult>
   <wfs:InsertResult handle="Statement 2">
      <ogc:FeatureId fid="BUILTUPA_1M.509876"/>
      <ogc:FeatureId fid="BUILTUPA_1M.509877"/>
   </wfs:InsertResult>
   <wfs:TransactionResult handle="Transaction 01">
      <wfs:Status>
         <wfs:SUCCESS/>
      </wfs:Status>
   </wfs:TransactionResult>
</wfs:TransactionResponse>
```

# 12 Operación *GetCapabilities*

## 12.1 Introducción

Un *web feature service* debe tener la habilidad de describir sus propias capacidades. Esta sección define un documento XML que debe generar un *web feature service* para definir sus propias capacidades.

El documento de capacidades definido en esta especificación está modelado inmediatamente después de modelar el documento de capacidades definido para servidores de mapas como está definido en el documento OpenGIS© *Web Map Service Implementation Specification*.

## 12.2 Petición

El elemento <*GetCapabilities*> es usado para solicitar el documento de capacidades de un *web feature service.*

Es definido por el siguiente fragmento de esquema XML:

```
<xsd:element name="GetCapabilities" type="wfs:GetCapabilitiesType"/>
   <xsd:complexType name="GetCapabilitiesType">
      <xsd:attribute name="version"
                     type="xsd:string" use="optional"/>
      <xsd:attribute name="service"
                     type="xsd:string" use="required" fixed="WFS"/>
   </xsd:complexType>
```

El atributo *service* esta descrito en la sección 7.8. El atributo *version*, a diferencia del caso normal, no es obligatorio ya que la aplicación cliente puede no tener a priori conocimiento acerca de que versión puede soportar el servidor. El cliente y el servidor necesitarán aplicar las reglas de negociación de versiones descritas en la sección 6.2.4.

## 12.3 Respuesta

### 12.3.1 Esquema de respuesta

El esquema de la respuesta a una petición *GetCapabilities* es normativamente definido usando el esquema XML en la sección A.4.

### 12.3.2 Documento de capacidades

El documento de capacidades esta compuesto de 4 secciones principales:

1. **Sección de servicio**
   La sección de servicio proporciona información acerca del servicio mismo.

2. **Sección de capacidades**
   La sección de capacidades especifica la lista de peticiones que un WFS puede manejar. Hay dos clases de *web feature service*, basados en las capacidades que soportan, y son definidos al inicio de este documento (ver sección i)

3. **Lista de *FeatureType***
   Esta sección define la lista de *features types* (y operaciones sobre cada *feature type*) que están disponibles en el *web feature service*. También es proporcionada información adicional acerca de cada *feature type*, tal como el sistema de referencia espacial (SRS).

4. **Sección de capacidades de filtrado**
   El esquema de la sección de capacidades de filtrado está definido en el documento OpenGIS© *Filter Encoding Implementation Specification*. Esta es una sección opcional. Si existe, entonces el WFS deberá soportar las operaciones ahí anunciadas. Si la sección de capacidades de filtrado no está definida, entonces el cliente deberá asumir que el servidor sólo soporta el conjunto mínimo de operaciones de filtrado como se define en el documento OpenGIS© *Filter Encoding Implementation Specification*.

El atributo *version* indica la revisión de la especificación a la cual aplica este esquema. Su formato es uno, dos o tres enteros separados por puntos: “x”, o “x.y”, o “x.y.z”, en donde el primer número es el más significativo. Así se garantiza que versiones futuras que vayan a ser numeradas de manera monolíticamente incremental, pensadas como espacios, aparezcan en la secuencia.

El atributo *updateSequence* es un número secuencial para dirigir la propagación del contenido del documento de capacidades. Por ejemplo, si un servidor de *features* agrega algún tipo de *feature*, puede incrementar la secuencia de actualización para informar al catálogo del servidor que su versión almacenada previamente, ahora está atrasada. El formato es un entero positivo.

### 12.3.3 Sección de servicio

El elemento <*Service*> contiene los meta-datos del *web feature service*. El siguiente contenido puede ser especificado para el elemento <*Service*>:

| Nombre del elemento | Descripción |
|---|---|
| *Name* | El nombre asignado por el proveedor de servicios a la instancia del *web feature service*. |
| *Title* | Es un título que debe ser entendible al hombre para identificar brevemente este servicio en los menús. |
| *Abstract* | Es una descripción narrativa para mayor información acerca del servidor. |
| *Keyword* | Debe contener palabras cortas para agregar al catálogo de búsqueda. |
| *OnlineResource* | Define el nivel HTTP URL más alto de este servicio. Normalmente es el URL del *homepage* para este servicio. |
| *Fees* | Contiene un bloque de texto indicando la cuota impuesta por proveedor de servicios para poder usar este servicio o para recuperar datos del WFS. La palabra *NONE* está reservada, significa |

| | |
|---|---|
| | *no fees* (sin cuotas). |
| *AccesConstraints* | Contiene un bloque de texto describiendo las restricciones de acceso impuestas por el proveedor de servicios sobre el WFS o para recuperar datos de este servicio. La palabra *NONE* está reservada para indicar que ninguna restricción está impuesta. |

**Tabla 3 – Elementos de la sección de servicio.**

### 12.3.4 Sección de Capacidades

La sección de capacidades es usada específicamente para definir la lista de operaciones que implementa un WFS en particular. Un WFS Básico incluirá las operaciones *GetCapabilities, DescribeFeatureType* y *GetFeature*. Un WFS Transaccional incluirá también la operación *Transaction*, y posiblemente la operación *LockFeature* y/o la operación *GetFeatureWithLock*.

Las capacidades específicas implementadas por un WFS están denotadas por los siguientes elementos:

| **Nombre del elemento** | **Descripción** |
|---|---|
| *GetCapabilities* | El elemento <***GetCapabilities***> es incluido para definir las plataformas de cómputo distribuido para este servicio. |
| *DescribeFeatureType* | El elemento <***DescribeFeatureType***> es usado para definir las plataformas de cómputo distribuido disponible para ese servicio e indicar el lenguaje de descripción de esquemas que puede ser usado para describir el esquema de un *featureType* cuando un cliente solicite tal descripción. *XMLSCHEMA* es el único lenguaje obligatorio que debe estar disponible. La entidad *SCHEMALANGUAGES* puede ser redefinida a lenguajes de vendedores específicos. |
| *Transaction* | El elemento <***Transaction***> es incluido para definir las plataformas de cómputo distribuido para este servicio. |
| *GetFeature* | El elemento <***GetFeature***> es incluido para definir las plataformas de cómputo distribuido para este servicio y enumerar los formatos disponibles para expresar los resultados de una solicitud. La entidad *RESULFORMATS* define el formato de salida obligatoria GML, pero puede ser redefinido para incluir formatos de vendedores específicos adicionales. |
| *LockFeature* | El elemento <***LockFeature***> es incluido para definir las plataformas de cómputo distribuido. |

**Tabla 4 – Operaciones de un *Web Feature Service***

La única plataforma de cómputo distribuida es HTTP, para la cual están definidos dos métodos; *GET* y *POST*. El atributo *onlineResource* indica el prefijo URL para las peticiones HTTP *GET* (todo lo que va antes del signo de interrogación y la petición: http://hostname[:port]/path/scriptname); para peticiones HTTP *POST*, *onlineResource* es el URL completo.

El elemento <*VendorSpecificCapabilities*> puede ser definido para incluir extensiones específicas del vendedor.

### 12.3.5 Sección *FeatureTypeList*

El propósito del elemento <*FeatureTypeList*> es contener una lista de tipos de *feature* que puede servir el WFS y definir los elementos de consulta y de transacción soportados por cada tipo de *feature*. Los elementos posibles de consulta y transacción son:

| Nombre del elemento | Descripción |
|---|---|
| *Insert* | El elemento <***Insert***> indica que el WFS es capaz de crear una nueva instancia de *feature*. |
| *Update* | El elemento <***Update***> indica que el WFS puede cambiar el estado existente de un *feature*. |
| *Delete* | El elemento <***Delete*** > indica que el WFS puede eliminar instancias de un tipo de *feature* de una fuente de datos. |
| *Query* | El elemento <***Query***> indica que el WFS es capaz de ejecutar consultas sobre un tipo de *feature*. |
| *Lock* | El elemento <***Lock***> indica que el WFS es capaz de bloquear instancias de un tipo de *feature*. |

**Tabla 5 – Elementos de consulta y transacción sobre *features*.**

Los elementos de consulta y de transacción pueden ser especificados globalmente para todos los tipos de *feature* o localmente para cada tipo de *feature* específico contenidos en el elemento <*FeatureTypeList*>. Si los elementos de consulta y de transacción son especificados globalmente, son heredados a cada tipo de *feature* contenido en el elemento <*FeatureTypeList*> y puede ser aumentado especificando localmente elementos de consulta o de transacción. Por ejemplo, el elemento <*Query*> puede ser especificado globalmente para todos los tipos de *feature* contenidos en el elemento <*FeatureTypeList*>, pero el elemento <*Update*> puede estar especificado solamente para un pequeño número de tipos de *feature*. Si no son definidos en ninguna parte elementos de consulta o de transacción, entonces el elemento <*Query*> será implicado para todos los tipo de *feature* contenidos en el elemento <*FeatureTypeList*>.

Los siguientes elementos pueden ser usados para describir cada tipo de *feature* contenido en un elemento <*FeatureTypeList*>:

| Elemento | Descripción |
|---|---|
| *Name* | El espacio de nombre calificado del tipo de *feature*. Este elemento es obligatorio. |
| *Title* | Es el título entendible al hombre para identificar brevemente este tipo de *feature* en los menús. |
| *Abstract* | Es una narrativa descriptiva para mayor información acerca del tipo de *feature*. |
| *Keyword* | Contiene palabras cortas para ayudar a la búsqueda en catálogos. |
| *SRS* | Es usado para indicar cual sistema de referencia espacial puede ser usado para expresar el estado de un *feature*. El SRS puede estar indicado usando la forma *Petrotechnical Open Software Corporation* "EPSG:<Código POSC>" o el formato URL definido en la sección 4.3.2 del documento: OpenGIS© *Geography Markup Language (GML) 2.1.2 Implementation Specification.* |
| *Operations* | Define cuales operaciones son soportadas sobre un tipo de *feature*. Cualquier operación definida localmente toma precedencia sobre cualquier operación definida globalmente. |
| *LatLongBoundingBox* | Es usado para indicar los extremos de un rectángulo en el SRS del tipo de *feature* asociado. Su propósito es facilitar búsquedas geográficas indicando dónde existen instancias de un tipo de *feature* particular. Ya que pueden ser especificadas múltiples *LatLongBoundingBoxes*, un WFS puede indicar dónde pueden existir varios clusters de datos. Este conocimiento ayuda a las aplicaciones clientes permitiéndoles saber dónde deberían consultar para tener una alta probabilidad de encontrar datos. |
| *Metadata URL* | Un WFS puede usar cero o más elementos <*MetadataURL*> para ofrecer meta-datos detallados y estandarizados acerca de los datos en un tipo de *feature* particular. El atributo *type* indica el estándar al cual se apega el meta-dato; el atributo *format* indica cómo está estructurado el meta-dato. Dos tipos son definidos actualmente: "TC211"=ISO TC211 19115; |

| | |
|---|---|
| | "FGDC"=FGDC CSDGM. |

**Tabla 6 – Elementos para describir los tipos de *feature*.**

### 12.4 Excepciones

En el caso de que un *web feature service* encuentre un error al servir una petición *GetCapabilities*, debe generar una excepción como se describió en la sección 7.7.

### 12.5 Ejemplos

Este ejemplo muestra cómo puede parecer un documento de capacidades para un *web feature service* básico. Para solicitar un documento de capacidades, el cliente podría enviar la siguiente petición:

```
<?xml version="1.0" ?>
<GetCapabilities
   service="WFS"
   xmlns="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-basic.xsd"/>
```

En respuesta a tal petición, un *web feature service* puede generar un documento como éste:

```
<?xml version="1.0" ?>
<WFS_Capabilities
   version="1.0.0"
   xmlns="http://www.opengis.net/wfs"
   xmlns:myns="http://www.someserver.com/myns"
   xmlns:ogc="http://www.opengis.net/ogc"
   ">

   <!--  The SERVICE section says something about who is providing the  -->
   <!--  service and where one can go to obtain more information about  -->
   <!--  the service.                                                   -->
   <Service>
      <Name>CubeWerx WFS</Name>
      <Title>CubeWerx Web Feature Service</Title>
      <Abstract>Web Feature Server maintained by CubeWerx Inc.</Abstract>
      <OnlineResource>http://www.someserver.com/wfs/cwwfs.cgi?</OnlineResource>
   </Service>

   <!-- The CAPABILITY section defines which WFS operations this        -->
   <!-- service instance supports, what distributed computing platform  -->
   <!-- is supported for each service and what the entry point is for   -->
   <!-- each operation.                                                 -->
   <Capability>
      <Request>
         <GetCapabilities>
            <DCPType>
               <HTTP>
                  <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
               </HTTP>
            </DCPType>
            <DCPType>
               <HTTP>
                  <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
               </HTTP>
            </DCPType>
         </GetCapabilities>
         <DescribeFeatureType>
            <SchemaDescriptionLanguage>
               <XMLSCHEMA/>
            </SchemaDescriptionLanguage>
            <DCPType>
               <HTTP>
                  <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
```

```
            </HTTP>
         </DCPType>
         <DCPType>
            <HTTP>
               <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
            </HTTP>
         </DCPType>
      </DescribeFeatureType>
      <GetFeature>
         <ResultFormat>
            <GML2/>
         </ResultFormat>
         <DCPType>
            <HTTP>
               <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
            </HTTP>
         </DCPType>
         <DCPType>
            <HTTP>
               <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
            </HTTP>
         </DCPType>
      </GetFeature>
      <GetFeatureWithLock>
         <ResultFormat>
            <GML2/>
         </ResultFormat>
         <DCPType>
            <HTTP>
               <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
            </HTTP>
         </DCPType>
         <DCPType>
            <HTTP>
               <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
            </HTTP>
         </DCPType>
      </GetFeatureWithLock>
      <Transaction>
         <DCPType>
            <HTTP>
               <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
            </HTTP>
         </DCPType>
         <DCPType>
            <HTTP>
               <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
            </HTTP>
         </DCPType>
      </Transaction>
      <LockFeature>
         <DCPType>
            <HTTP>
               <Get onlineResource="http://www.someserver.com/wfs/cwwfs.cgi?"/>
            </HTTP>
         </DCPType>
         <DCPType>
            <HTTP>
               <Post onlineResource="http://www.someserver.com/wfs/cwwfs.cgi"/>
            </HTTP>
         </DCPType>
      </LockFeature>
   </Request>
</Capability>

<!-- The FEATURETYPELIST section defines the list of feature types   -->
<!-- that this service instance can operate upon as well as which    -->
<!-- operations are supported on each feature type.                  -->
<FeatureTypeList>
   <Operations>
      <Query/>
   </Operations>
   <FeatureType>
      <Name>myns:BUILTUPA_1M</Name>
```

```
            <SRS>EPSG:4326</SRS>
            <Operations>
               <Insert/>
               <Update/>
               <Delete/>
            </Operations>
            <LatLongBoundingBox minx="-179.1296081543" miny="-53.167423248291"
                                maxx="178.44325256348" maxy="70.992721557617"/>
         </FeatureType>
         <FeatureType>
            <Name>myns:COASTL_1M</Name>
            <SRS>EPSG:4326</SRS>
            <LatLongBoundingBox minx="-179.99942016602" miny="-85.582763671875"
                                maxx="179.9999" maxy="83.627418518066"/>
         </FeatureType>
         <FeatureType>
            <Name>myns:ELEVP_1M</Name>
            <SRS>EPSG:4326</SRS>
            <Operations>
               <Insert/>
               <Update/>
               <Delete/>
            </Operations>
            <LatLongBoundingBox minx="-179.9984893715" miny="-89.83837892767"
                                maxx="179.99234007206" maxy="83.520408603363"/>
         </FeatureType>
         <FeatureType>
            <Name>myns:OCEANSEA_1M</Name>
            <SRS>EPSG:4326</SRS>
            <Operations>
               <Insert/>
               <Update/>
               <Delete/>
            </Operations>
            <LatLongBoundingBox minx="-179.9999" miny="-85.582763671875"
                                maxx="179.99996948242" maxy="89.9999"/>
         </FeatureType>
         <FeatureType>
            <Name>myns:RAILRDL_1M</Name>
            <SRS>EPSG:4326</SRS>
            <Operations>
               <Insert/>
               <Update/>
               <Delete/>
            </Operations>
            <LatLongBoundingBox minx="-165.24467468262" miny="-53.138427734375"
                                maxx="179.60989379883" maxy="78.16796875"/>
         </FeatureType>
         <FeatureType>
            <Name>myns:TREESA_1M</Name>
            <SRS>EPSG:4326</SRS>
            <Operations>
               <Insert/>
               <Update/>
               <Delete/>
            </Operations>
            <LatLongBoundingBox minx="-139.99757385254" miny="25.281270980835"
                                maxx="-52.661720275879" maxy="66.718765258789"/>
         </FeatureType>
      </FeatureTypeList>

      <!-- The FILTER_CAPABILITIES section defines the capabilities of the -->
      <!-- filter supported by this feature instance.  For example, in     -->
      <!-- this case all spatial operator are supported.  Another, simpler -->
      <!-- WFS implementation, may only support the BBOX operator.         -->
      <ogc:Filter_Capabilities>
         <ogc:Spatial_Capabilities>
            <ogc:Spatial_Operators>
               <ogc:BBOX/>
               <ogc:Equals/>
               <ogc:Disjoint/>
               <ogc:Intersect/>
               <ogc:Touches/>
               <ogc:Crosses/>
```

```
            <ogc:Contains/>
            <ogc:Overlaps/>
         </ogc:Spatial_Operators>
      </ogc:Spatial_Capabilities>
      <ogc:Scalar_Capabilities>
         <ogc:Logical_Operators/>
         <ogc:Comparison_Operators>
            <ogc:Simple_Comparisons/>
            <ogc:Like/>
            <ogc:Between/>
            <ogc:NullCheck/>
         </ogc:Comparison_Operators>
         <ogc:Arithmetic_Operators>
            <ogc:Simple_Arithmetic/>
            <ogc:Functions>
               <ogc:Function_Names>
                  <ogc:Function_Name nArgs="1">MIN</ogc:Function_Name>
                  <ogc:Function_Name nArgs="1">MAX</ogc:Function_Name>
                  <ogc:Function_Name nArgs="1">COUNT</ogc:Function_Name>
                  <ogc:Function_Name nArgs="1">DISTINCT</ogc:Function_Name>
               </ogc:Function_Names>
            </ogc:Functions>
         </ogc:Arithmetic_Operators>
      </ogc:Scalar_Capabilities>
   </ogc:Filter_Capabilities>
</WFS_Capabilities>
```

## 13 Codificación par palabra-valor (*Keyword-value pair*)

### 13.1 Introducción

Esta sección describe cómo codificar las operaciones de un WFS usando el estilo estándar CGI de pares palabra-valor. Esto significa que los parámetros consisten en pares de nombre-valor de la forma “nombre=valor” y los pares son separados por el caracter “&”. Esta forma de codificar también es conocida como codificación URL.

#### 13.1.1 Una nota acerca de los ejemplos

En general, la codificación URL requiere que ciertos caracteres, tal como “&”, se omitan cuando no son usados de la manera prevista. En esta sección, sin embargo, tales caracteres no serán omitidos por motivos de claridad.

Además, varios de los ejemplos en esta sección incluyen un parámetro *FILTER* cuyo valor es un filtro XML codificado como se especifica en el documento: OpenGIS© *Filter Encoding Implementation Specification.* Para ser rigurosamente correctos, estos ejemplos deben incluir la información de la localización de los espacios de nombre y de los esquemas en el elemento raíz *<Filter>*, de tal manera que el XML pueda ser validado. De este modo el parámetro:

```
FILTER=<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName>
       <gml:Box><gml><coordinates>10,10 20,20</gml:coordinates></gml:Box>
       </Within></Filter>
```

Debe ser más correctamente especificado como:

```
FILTER=<Filter xmlns="http://www.opengis.net/ogc"
               xmlns:gml="http://www.opengis.net/gml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://www.opengis.net/ogc
                                   ../filter/1.0.0/filter.xsd
                                   http://www.opengis.net/gml
                                   ../gml/2.1/geometry.xsd">
       <Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName>
       <gml:Box><gml><coordinates>10,10 20,20</gml:coordinates></gml:Box>
       </Within></Filter>
```

Sin embargo, para no confundir la información esencial, los *tags* de atributo de localización a esquemas y espacios de nombre, han sido omitidos de los ejemplos de esta sección. Además, las localizaciones a los esquemas mostrados son solamente ejemplos y las localizaciones correctas a los esquemas necesitarían ser sustituidas.

Finalmente, los valores de *FILTER* y otros parámetros (arriba) están discontinuos en varias líneas, nuevamente por motivos de claridad. El valor de un parámetro *FILTER*, en la práctica, debe estar compuesto de una sola cadena larga.

## 13.2 Reglas para los parámetros de las peticiones

### 13.2.1 Orden y uso de los parámetros

Los nombres de los parámetros no deben ser sensibles a mayúsculas y minúsculas, pero los valores de los parámetros sí deben ser sensibles a éstas. En este documento, los nombres de parámetros son mostrados típicamente en mayúsculas para claridad tipográfica, no como requisito.

Los parámetros en una petición pueden ser especificados en cualquier orden.

Un OGC *Web Service* debe estar preparado para encontrar parámetros que no son parte de esta especificación. En términos de producir resultados para esta especificación, un OGC *Web Service* debe ignorar tales parámetros.

### 13.2.2 Lista de parámetros

Los listas que contienen parámetros deben usar la coma (“,”) como el delimitador entre los elementos de la lista. Además, pueden ser especificadas múltiples listas como valor de un parámetro si encerramos cada lista entre paréntesis “(“,”)”.

**Ejemplo 1**

Un ejemplo de una lista de elementos.

```
parameter=item1,item2,item2
```

**Ejemplo 2**

Un ejemplo de múltiples listas de elementos asignadas a un parámetro simple.

```
parameter=(item11,item12,item13)(item21,item22,item23)
```

Los paréntesis también pueden ser usados para delimitar filtros locales cuando se especifica más de un tipo de *feature* en el parámetro *TYPENAME*. El siguiente fragmento URL muestra como se hace:

```
typename=FEAT1,FEAT2&filter=(<Filter>… FEAT1 filter…</Filter>)(<Filter>… FEAT2
filter…</Filter>)
```

## 13.3 Parámetros comunes de las peticiones

### 13.3.1 Parámetro *version*

El parámetro *version* especifica el número de versión del protocolo. El formato del número de versión y la negociación de versiones se describen en la sección 6.2.4.

### 13.3.2 Parámetro *request*

El parámetro *request* indica cuál operación del servicio está siendo invocada. El valor nombre_operación debe ser uno de los ofrecidos por la instancia del OGC *web service*.

### 13.3.3 *Bounding Box*

El parámetro *bounding box*, BBOX, está incluido en esta especificación por conveniencia, como una representación abreviada del muy común filtro *bounding box* el cuál podría ser expresado en una forma mucho más larga usando XML y la codificación de filtrado descrita en el documento OpenGIS© *Filter Encoding Implementation Specification*. Un BBOX aplica a todos los tipos de *feature* listados en la petición.

El *Bounding Box* (BBOX) es un conjunto de cuatro decimales separados por comas, notación científica o valores enteros (si los enteros son proporcionados donde son requeridos puntos flotantes, se asume que el punto decimal va al final del número). Estos valores especifican los rangos de la X mínima, la Y mínima, la X máxima y la Y máxima, en ese orden, expresadas en unidades del SRS del tipo de *feature* que está siendo consultado. Los cuatro valores del *bounding box* indican las orillas exteriores de un rectángulo, como se muestra en la figura 5; X mínima es la orilla izquierda, X máxima la derecha, Y mínima la de abajo, y Y máxima la de arriba.

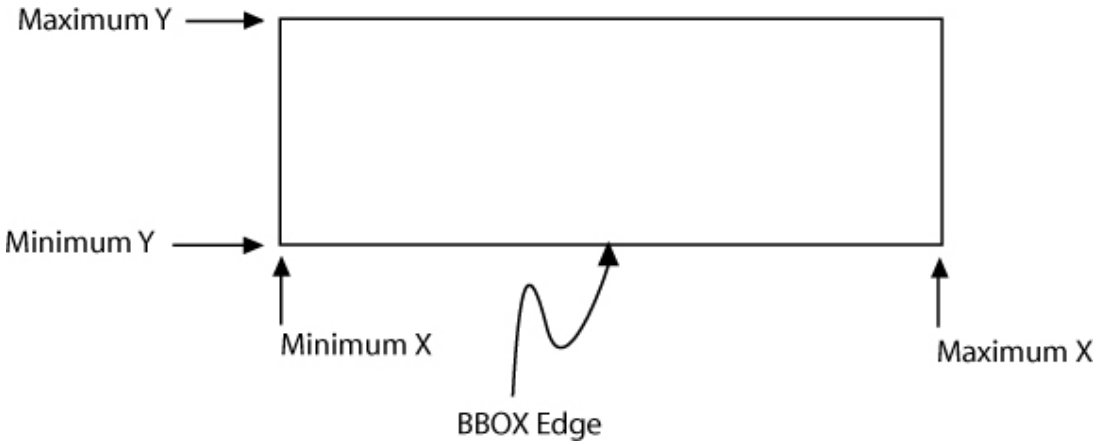


**Figura 5 – Representación de un *bounding box***

Un *bounding box* no debe tener área cero.

Si una petición contiene un *bounding box* inválido (p.e.: uno cuya X mínima sea más grande o igual que la X máxima, ó cuyo Y mínima sea más grande o igual a la Y máxima) el servidor deberá tirar una excepción.

Si una petición contiene un *bounding box* cuya área no la cubre del todo con las *LatLongBoundingBox(s)* anunciadas en el XML de capacidades para el objeto de geo-datos solicitado, el servidor debe regresar un contenido vacío (p.e.: un conjunto de *feature* nulo) para ese elemento. Cualesquiera de los elementos que están parcial o completamente contenidos en el *bounding box* deben ser regresados en el formato apropiado.

El SRS del *bounding box* debe ser el mismo que el SRS del(los) tipo(s) de *feature* en la petición. El SRS de un tipo de *feature* es anunciado por un WFS en el documento de capacidades. Si más de un tipo de *feature* es especificado en una petición, los tipos de *feature* deben estar todos en el mismo SRS y el BBOX debe ser especificado en el SRS común también.

Si los valores del *bounding box* no son definidos en el SRS dado (p.e: latitudes más grandes que 90 grados en EPSG:4326), el servidor debe regresar un contenido vacío para áreas fuera del rango válido del SRS.

En el caso particular de la longitud, el siguiente comportamiento se aplica con respecto al contra-meridiano en los 180 grados de longitud. Hay un deseo legítimo para los mapas que atraviesan el contra-meridiano (por ejemplo, un mapa centrado en el Océano Pacífico). Si Xmin es la longitud más al oeste en grados y Xmax es la más al este, entonces aplica la siguiente restricción:

```
-180 <= Xmin < Xmax < 540
```

**Ejemplo**

Los valores Xmin, Xmax y el alcance correspondiente del *bounding box*:

```
-180,180 = Earth centered at Greenwich
0,360 = Earth with Greenwich at left edge
120,250 = Pacific Ocean
```

### 13.3.4 Parámetros específicos de vendedor

Las peticiones pueden permitir parámetros de vendedores específicos (VSPs) que mejorarán los resultados de una petición. Típicamente, éstos son usados para pruebas privadas de funcionalidades no estándar antes de la posible estandarización. No se requiere ni se espera que un cliente genérico haga uso de estos VSPs.

Un OGC *web service* debe producir un resultado válido incluso si faltan o están mal formados (p.e.: el servicio debe proveer un valor por *default*), ó si los VSPs que se proveen, no son conocidos por el servidor (p.e.: el servicio deberá ignorar parámetros de petición desconocidos).

Un OGC *web service* puede escoger no anunciar algunos o todos de sus VSPs. Si los VSPs son incluidos en el XML de capacidades, el elemento *VendorSpecificCapabilities*, de acuerdo a eso, deberá ser redefinido. Pueden importarse esquemas adicionales que contengan la redefinición del elemento.

Los clientes pueden leer la definición específica del esquema de capacidades y formular peticiones usando cualesquiera VSPs anunciados ahí.

Los vendedores deben escoger nombres de parámetros cuidando evitar conflictos con los parámetros estándares.

## 13.4 Parámetros comunes

La siguiente tabla describe los parámetros comunes a todas las peticiones de WFS. Las tablas subsecuentes pueden redefinir algunas facetas de uno o más parámetros de esta tabla.

| Componente URL | O / M | Default | Descripción |
|---|---|---|---|
| http://server_address/path/script | M | | Prefijo URL de WFS |
| *VERSION* | M* | 1.0.0 | Solicita la versión |
| *SERVICE* | M | WFS | Tipo de servicio |
| *REQUEST* | M | | Nombre de la petición WFS |
| Parámetros adicionales | O | | Como se describen en esta sección |
| Parámetros específicos de vendedor | O | | Parámetros opcionales de vendedores específicos |

O= *Optional* (opcional), M= *Mandatory* (obligatorio)

**VERSION* es obligatorio para todas las operaciones excepto la operación *GetCapabilities*

**Tabla 7 – Parámetros comunes para peticiones WFS**

El parámetro obligatorio *VERSION* especifica el número de versión del protocolo y permite la negociación como se describió en la sección 6.2.4.

El parámetro obligatorio *SERVICE* especifica cual de los tipos de servicios disponibles en la instancia particular del servicio está siendo invocado. El valor **WFS** es usado para indicar que debe ser invocado un *Web Feature Service*.

El parámetro *REQUEST* debe también ser incluido e indica cuál operación se está invocando del *web feature service*. Los posibles valores de los parámetros *REQUEST* son: *DescribeFeatureType, LockFeature, Transaction, GetFeature, GetFeatureWithLock* ó *GetCapabilities*.

Los parámetros adicionales *GET*, como se describen en esta sección, deben ser expresados como pares nombre-valor. Los nombres de los parámetros no deben ser sensibles a mayúsculas y minúsculas, pero sí los valores de esos parámetros. Los parámetros en una petición pueden ser especificados en cualquier orden.

Un WFS debe estar preparado para encontrar parámetros que no son parte de la especificación. Estos son conocidos como parámetros específicos de vendedor. Los parámetros específicos de vendedor permiten a los vendedores especificar parámetros adicionales que mejorarán los resultados de las peticiones. También el WFS debe producir un resultado válido incluso si faltan o están malformados los parámetros específicos de vendedor. Debe declarar los parámetros específicos de vendedor dentro de su XML de capacidades y puede escoger anunciar algunos o todos de estos parámetros específicos de vendedor. Los clientes pueden leer el esquema de capacidades y formular peticiones usando cualquier parámetro específico de vendedor anunciado en él.

### 13.5 Respuesta

La respuesta a cualquier petición codificada usando los pares palabra-valor debe ser idéntica a la respuestas generadas para las peticiones codificadas en XML y descritas en las secciones anteriores de este documento.

### 13.6 Excepciones

El reporte de excepciones para peticiones codificadas usando pares palabra-valor debe ser idéntico al generado por las peticiones usando XML. Referidas a las secciones 7.7 y 11.4 para una discusión detallada del reporte de excepciones.

### 13.7 Operaciones

#### 13.7.1 Introducción

Esta sección describe como formular peticiones WFS usando codificaciones de pares palabra-valor estándar estilo CGI. Se hace uso pesado de ejemplos que ilustran las varias formas posibles. Además, para mayor claridad, cada especificación de un parámetro en los ejemplos es colocado en una línea separada.

### 13.7.2 Operación *DescribeFeatureType*

#### 13.7.2.1 Petición

| Componente URL | O / M | Default | Desripción |
|---|---|---|---|
| *REQUEST=DescribeFeatureType* | M | | Nombre de la petición |
| *TYPENAME* | O | | Una lista de tipos de *features* separadas por coma para describir. Si no se especifica ningún valor esto se interpreta como todos los tipos de *feature*. |
| *OUTPUTFORMAT* | O | XMLSCHEMA | El formato de salida usado para describir el *feature*. Debe ser soportado XMLSCHEMA. Es posible otro formato de salida, como DTD. |

**Tabla 8 – Codificación *DescribeFeatureType***

#### 13.7.2.2 Ejemplos

**Ejemplo 1**

El siguiente ejemplo solicita el esquema de descripción del tipo de *feature* TREESA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=DescribeFeatureType&
       TYPENAME=TREESA_1M
```

**Ejemplo 2**

El siguiente ejemplo solicita un esquema de descripción de los tipos de *feature* INWATERA_1M y BUILTUPA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=DescribeFeatureType&
       TYPENAME=TREESA_1M,BUILTUPA_1M
```

### 13.7.3 Operaciones *GetFeature* y *GetFeatureWithLock*

#### 13.7.3.1 Petición

| Componente URL | O / M | Default | Descripción |
|---|---|---|---|
| *REQUEST=[GetFeature \| GetFeatureWithLock]* | M | | El nombre de la petición WFS |
| *PROPERTYNAME* | O | | Una lista de propiedades puede ser especificada para cada tipo de *feature* que está siendo consultado. Refiérase a la sección 13.2.2 en cómo formar listas de parámetros. Un caracter "*" puede ser usado para indicar que se deben recuperar todas las propiedades. Hay una relación 1:1 entre cada elemento en una lista *FEATUREID* o *TYPENAME* y la lista *PROPERTYNAME*. La ausencia de un valor indica que se deben recuperar todas las propiedades. |
| *FEATUREVERSION=[ALL \|N]* | O | | Si se soportan varias versiones, el parámetro *FEATUREVERSION* indica al WFS cual versión de |

| | | | |
|---|---|---|---|
| | | | *feature* recuperar. Un valor de *ALL* indica que se recuperarán todas las versiones de un *feature*. Un valor entero recupera la N$^{va}$ versión de un *feature.* Ningún valor, indica que se recuperará la última versión del *feature*. |
| *MAXFEATURES=N* | O | | Un entero positivo indica el número máximo de *features* que un WFS regresará en respuesta a una consulta. Si no se especifica ningún valor entonces se presentarán todas las instancias del resultado. |
| *TYPENAME* (si se especifica FEATUREID, este componente es **opcional**) | M | | Una lista de nombres de tipos de *feature* para consultar. |
| *FEATUREID* ( es **mutuamente exclusivo** con *FILTER* y *BBOX*) | O | | Una lista enumerada de instancias de *feature* para recuperar que son identificadas por sus identificadores de *feature.* |
| *FILTER* (**Pre-requisito:** *TYPENAME*) (es **mutuamente exclusivo** con *FEATUREID* y *BBOX*) | O | | La especificación de filtrado describe un conjunto de *features* sobre las cuales operar. El filtro es definido como se especifica en el documento OpenGIS© *Filter Encoding Implementation Specification.* Si se usa el parámetro *FILTER*, se debe especificar un filtro por cada tipo de *feature* listado en el parámetro *TYPENAME*. Los filtros individuales codificados en el parámetro *FILTER* se encierran en paréntesis "(" y ")". |
| *BBOX* (**Pre-requisito:** *TYPENAME*) (es **mutuamente exclusivo** con *FEATUREID* y *FILTER*) | O | | En lugar de un *FEATUREID* ó un *FILTER*, un cliente puede especificar un *bounding box* como se describe en la sección 13.3.3. |

**Tabla 9 – Codificación *GetFeature* y *GetFeatureWithLock***

#### 13.7.3.2 Ejemplos

Muchos de los ejemplos de esta sección incluyen un parámetro *FILTER* cuyo valor es un filtro codificado XML como se especifica en el documento OpenGIS© *Filter Encoding Implementation Specification.* Para ser rigurosamente correcto, estos ejemplos deben incluir información de la localización de los espacios de nombre y del esquema en el elemento raiz *<Filter>*, de tal manera que el XML sea válido. De este modo el parámetro:

```
FILTER=<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName>
       <gml:Box><gml><coordinates>10,10 20,20</gml:coordinates></gml:Box>
       </Within></Filter>
```

debería ser más correctamente:

```
FILTER=<Filter xmlns="http://www.opengis.net/ogc"
               xmlns:gml="http://www.opengis.net/gml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://www.opengis.net/ogc
                                   ../filter/1.0.0/filter.xsd
                                   http://www.opengis.net/gml
                                   ../gml/2.1/geometry.xsd">
       <Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName>
       <gml:Box><gml><coordinates>10,10 20,20</gml:coordinates></gml:Box>
```

```
            </Within></Filter>
```

Por motivos de claridad, sin embargo, las etiquetas de atributo de la localización del espacio de nombre y del esquema han sido omitidas de los ejemplos en esta sección. Además, la localización en los esquemas mostrados es solo una localizacion de ejemplo, la localización correcta del esquema necesitaría ser especificada.

Finalmente, el valor del parámetro *FILTER* (arriba) es discontinuo en varias líneas de nuevo por motivos de claridad. Un parámetro *FILTER* real, tendría una sola línea como argumento.

**Ejemplo 1**

Consulta todas las propiedades de todas las instancias del tipo INWATERA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       TYPENAME=INWATERA_1M
```

**Ejemplo 2**

Consulta algunas propiedades de todas las instancias del tipo INWATERA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID&
       TYPENAME=INWATERA_1M
```

**Ejemplo 3**

Consulta todas las propiedades de la instancia de *feature* identificada por el identificador de *feature* “INWATERA_1M.1013”.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       FEATUREID=INWATERA_1M.1013
```

**Ejemplo 4**

Consulta algunas propiedades de la instancia de *feature* identificada por el identificador de *feature* “INWATERA_1M.1013”.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID&
       FEATUREID=INWATERA_1M.1013
```

**Ejemplo 5**

Consulta todas las propiedades de un conjunto enumerado de instancias de *feature* del tipo INWATERA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       FEATUREID=INWATERA_1M.1013,INWATERA_1M.1014,INWATERA_1M.1015
```

**Ejemplo 6**

Consulta algunas propiedades de un conjunto enumerado de instancias de *feature* de tipo INWATERA_1M. En este ejemplo, son seleccionadas las propiedades WKB_GEOM y TITLE_ID de cada instancia de *feature*.

```
http://www.someserver.com/wfs.cgi&
     SERVICE=WFS&
     VERSION=1.0.0&
     REQUEST=GetFeature&
     PROPERTYNAME=(INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID)
                  (INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID)
                  (INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID)&
     FEATUREID=INWATERA_1M.1013,INWATERA_1M.1014,INWATERA_1M.1015
```

**Ejemplo 7**

Consulta todas las propiedades de un conjunto restringido de instancias de *feature* de tipo INWATERA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       TYPENAME=INWATERA_1M&
       FILTER=<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName>
              <gml:Box><gml><coordinates>10,10 20,20</gml:coordinates></gml:Box>
              </Within></Filter>
```

**Ejemplo 8**

Consulta algunas propiedades de un conjunto restringido de instancias de *feature* de tipo INWATERA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID&
       TYPENAME=INWATERA_1M&
       FILTER=<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName><gml:Box>
              <gml:coordinates>10,1020,20</gml:coordinates></gml:Box></Within></Filter>
```

**Ejemplo 9**

Consulta todas las propiedades de una lista de tipos de *feature*.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       TYPENAME=INWATERA_1M,BUILTUPA_1M
```

**Ejemplo 10**

Consulta algunas propiedades de una lista de *features*. En este caso, los atibutos WKB_GEOM y TITLE_ID son recuperados del tipo de *feature* INWATERA_1M y también son recuperados todos los atributos del tipo de *feature* BUILTUPA_1M.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       PROPERTY=(INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID)(BUILTUPA_1M/*)&
       TYPENAME=INWATERA_1M,BUILTUPA_1M
```

**Ejemplo 11**

Consulta todas las propiedades de un conjunto enumerado de instancias de *feature*.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       FEATUREID=INWATERA_1M.1013,BUILTUP_1M.3456
```

**Ejemplo 12**

Consulta algunas propiedades de un conjunto enumerado de instancias de *feature*. En este caso, se recuperarán los atributos WKB_GEOM y TITLE_ID de la instancia de *feature* "INWATERA_1M.1013". Se recuperará el atributo WKB_GEOM de la instancia de *feature* "BUILTUPA_1M.3456".

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=INWATERA_1M/WKB_GEOM,INWATERA_1M/TILE_ID,BUILTUPA_1M/WKB_GEOM)&
       FEATUREID=INWATERA_1M.1013,BUILTUPA_1M.3456
```

**Ejemplo 13**

Consulta todas las propiedades de las instancias de *feature* de los tipos de *feature* INWATERA_1M y BUILTUPA_1M que caen dentro del *bounding box* especificado.

```
http://www.someserver.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       TYPENAME=INWATERA_1M,BUILTUPA_1M&
       FILTER=(<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM
              <PropertyName><gml:Box><gml:coordinates>10,10 20,20</gml:Box>
              </gml:coordinates></Within></Filter>)(<Filter><Within><PropertyName>
              BUILTUPA_1M/WKB_GEOM<PropertyName><gml:Box><gml:coordinates>10,10
              20,20</gml:Box></gml:coordinates></Within></Filter>)
```

**Ejemplo 14**

Consulta algunas propiedades de un conjunto restringido de instancias de *feature* de tipo INWATERA_1M y BUILTUPA_1M.

```
http://www.SiriusCyberneticsCorp.com/wfs.cgi&
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=GetFeature&
       PROPERTYNAME=(INWATERA_1M/WKB_GEOM,INWATERA_1MTILE_ID)(BUILTUPA_1M/WKB_GEOM)&
       TYPENAME=INWATERA_1M,BUILTUPA_1M&
       FILTER=(<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM|INWATERA_1M/WKB_GEOM
              <PropertyName><gml:Box><gml:coordinates>10,10 20,20</gml:coordinates>
              </gml:Box></Within></Filter>)(<Filter><Within><PropertyName>
              INWATERA_1M/WKB_GEOM<PropertyName><gml:Box><gml:coordinates>10,10
              20,20</gml:coordinates></gml:Box></Within></Filter>)
```

### 13.7.4 Operación *LockFeature*

#### 13.7.4.1 Petición

| Componente URL | O / M | Default | Descripción |
|---|---|---|---|
| *REQUEST=LockFeature* | M | | El nombre de la petición |
| *TYPENAME* (si se especifica *FEATUREID* este | M | | Nombres de uno o más tipos de *feature* cuyas instancias de *feature* serán bloqueadas. |

| | | | |
|---|---|---|---|
| componente es **opcional**). | | | |
| *EXPIRY* | O | | El número de minutos que el bloqueo debe persistir antes de ser liberado. Si ningún valor es especificado el bloqueo persistirá indefinidamente. |
| *LOCKACTION=[ALL \| SOME]* | O | | Especifica como el bloqueo debe ser adquirido. *ALL* indica que se intentará bloquear todos los *features*, de otra manera fallará. *SOME* indica que se intentará bloquear tantos *features* como se pueda. |
| *FEATUREID* (**mutuamente exclusivo** con *FILTER* y con *BBOX*) | O | | Una lista enumerada de identificadores de instancias de *feature* indicando cuales instancias de *feature* serán bloqueadas. |
| *FILTER* (**Pre-requisito:** *TYPENAME*) (**mutuamente exclusivo** con *FEATUREID* y *BBOX*) | O | | Una cadena XML codificada como se describe en el documento OpenGIS© *Filter Encoding Implementation Specification,* indicando sobre cuales *features* se va a operar. Si es usado el parámetro *FILTER*, se debe especificar un filtro para cada tipo de *feature* listado en el parámetro *TYPENAME*. Los filtros individuales en el parámetro *FILTER* se encierran entre paréntesis "(" y ")". |
| *BBOX* (**Pre-requisito:** *TYPENAME)* (**mutuamente exclusivo** con *FEATUREID* y *FILTER*) | O | | En lugar de un *FEATUREID* o *FILTER*, un cliente puede especificar un *bounding box* como se describe en la sección 13.3.3. |

**Tabla 10 – Codificación *LockFeature***

#### 13.7.4.2 Ejemplos

**Ejemplo 1**

El siguiente ejemplo bloquea todas las instancias del tipo de *feature* INWATERA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=LockFeature&
       TYPENAME=INWATERA_1M
```

**Ejemplo 2**

El siguiente ejemplo bloquea el *feature* indentificado por "ROADL_1M.1013".

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=LockFeature&
       FEATUREID=ROADL_1M.1013
```

**Ejemplo 3**

El siguiente ejemplo bloquea todas las instancias de *feature* del tipo de *feature* INWATERA_1M y BUILTUPA_1M.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=LockFeature&
       TYPENAME=INWATER_1M,BUILTUPA_1M
```

**Ejemplo 4**

El siguiente ejemplo bloquea todos los *features* del tipo de *feature* INWATERA_1M y BUILTUPA_1M que caen dentro de una región específica de interés.

```
http://www.someserver.com/wfs.cgi?
     SERVICE=WFS&
     VERSION=1.0.0&
     REQUEST=LockFeature&
     LOCKACTION=ALL&
     TYPENAME=INWATER_1M,BUILTUPA_1M&
     FILTER=(<Filter><Within><PropertyName>WKB_GEOM</PropertyName><gml:Box>
             <gml:coordinates>10,10 20,20</gml:coordinates></gml:Box></Within>
             </Filter>)(<Filter><Within><PropertyName>WKB_GEOM</PropertyName><gml:Box>
             <gml:coordinates>10,10 20,20</gml:coordinates></gml:Box></Within>
             </Filter>)
```

### 13.7.5 Operación *Transaction*

La única operación soportada de la interface de transacción es la operación *DELETE*. Para expresar peticiones *INSERT* o *UPDATE*, lo cual es un poco largo, no es conveniente usar la codificación de pares palabra-valor.

#### 13.7.5.1 Peticion

| Componente URL | O / M | Default | Descripción |
|---|---|---|---|
| *REQUEST=Transaction* | M | | El nombre de la petición WFS. |
| *OPERATION=Delete* | M | | Operación *transaction* a ejecutarse. Actualmente solo *Delete* esta definida. |
| *TYPENAME* (si se especifica *FEATUREID* este componente es **opcional**). | M | | Una lista de tipos de *feature* sobre los cuales aplicar la operación |
| *RELEASEACTION=[ALL \| SOME]* | O | | *ALL* indica que todos los *features* bloqueados deberán ser liberados cuando termine la transacción. Un valor de *SOME* indica que sólo los registros que han sido modificados deberán ser liberados. Los bloqueos restantes se mantienen. |
| *FEATUREID* (**mutuamente exclusivo** con *FILTER* y con *BBOX*) | O | | Una lista de identificadores de *feature* sobre los cuales se aplicará la operación epecificada.<br>Opcional.<br>No default. |
| *FILTER* (**Pre-requisito:** *TYPENAME*) (**mutuamente exclusivo** con *FEATUREID* y *BBOX*) | O | | La especificación de filtrado describe un conjunto de *features* sobre las cuales operar. El filtro es definido como se especifica en el documento OpenGIS© *Filter Encoding Implementation Specification.* Si se usa el parámetro *FILTER*, se debe especificar un filtro por cada tipo de *feature* listado en el parámetro *TYPENAME*. Los filtros individuales codificados en el parámetro *FILTER* se encierran entre paréntesis "(" y ")". |
| *BBOX* (**Pre-requisito:** *TYPENAME)* (**mutuamente exclusivo** con *FEATUREID* y *FILTER*) | O | | En lugar de un *FEATUREID* o *FILTER*, un cliente puede especificar un *bounding box* como se describe en la sección 13.3.3. |

**Tabla 11 – Codificación *Transaction***

#### 13.7.5.2 Ejemplos

**Ejemplo 1**

Elimina la instancia de *feature* identificada por "ROADL_1M.1013".

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=Transaction&
       OPERATION=Delete&
       FEATUREID=ROADL_1M.1013
```

**Ejemplo 2**

El siguiente ejemplo elimina todos los *features* de tipo INWATERA_1M y BUILTUPA_1M que caen dentro del *bounding box* especificado.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=Transaction&
       OPERATION=Delete&
       TYPENAME=INWATER_1M,BUILTUPA_1M&
       FILTER=(<Filter><Within><PropertyName>INWATERA_1M/WKB_GEOM<PropertyName><gml:Box>
              <gml:coordinates>10,10 20,20</gml:coordinates></gml:Box></Within>
              </Filter>)(< Filter><Within><PropertyName>BUILTUPA_1M/WKB_GEOM
              <PropertyName><gml:Box><gml:coordinates>10,10 20,20</gml:coordinates>
              </gml:Box></Within></Filter>)
```

**Ejemplo 3**

El siguiente ejemplo es el mismo que el ejemplo 2, excepto que el parámetro *BBOX* es usado para especificar la restricción espacial sobre el comando *Delete*.

```
http://www.someserver.com/wfs.cgi?
       SERVICE=WFS&
       VERSION=1.0.0&
       REQUEST=Transaction&
       OPERATION=Delete&
       TYPENAME=INWATER_1M,BUILTUPA_1M&
       BBOX=10,10,20,20
```

### 13.7.6 Operación *GetCapabilities*

#### 13.7.6.1 Petición

| Componente URL | O / M | Default | Descripción |
|---|---|---|---|
| *REQUEST=GetCapabilities* | M | | El nombre de la petición WFS. |

**Tabla 12 – Codificación *GetCapabilities***

#### 13.7.6.2 Ejemplos

**Ejemplo 1**

Solicita el documento de capacidades de un WFS.

```
http://www.someserver.com/wfs.cgi?
       VERSION=1.0.0&
       SERVICE=WFS&
       REQUEST=GetCapabilities
```

## Anexo A – Definiciones de esquema XML (normativo)

### A.1 Introducción

Este anexo contiene las definiciones de los esquemas XML normativos de las operaciones WFS, documento de capacidades y excepciones.

El anexo A.2 contiene la definición del esquema XML para mensajes de excepción generados por un *web feature service*. Todos los mensajes de excepciónn generados por un WFS deben ser válidos contra este esquema.

El anexo A.3 contiene la definición del esquema XML de las operaciones básicas que un *web feature service* debe proveer. El conjunto básico de operaciones incluye *GetCapabilities, DescribeFeatureType* y *GetFeature*. Un *web feature service* debe implementar este conjunto de operaciones.

El anexo A.4 contiene la definición del esquema XML de las operaciones opcionales relativas al procesamiento de transacciones. El conjunto de operaciones incluye *GetFeatureWithLock, LockFeature* y *Transaction*. Un *web feature service* puede implementar algunas o todas de estas operaciones.

El anexo A.5 contiene la definición de esquema XML del documento de capacidades que un *web feature service* puede generar en respuesta a una petición *GetCapabilities*. La respuesta *GetCapabilities* de un *web feature service* debe ser válida contra este esquema. Más adelante, si un *web feature service* implementa extensiones de un vendedor específico, este esquema en el anexo A.5 deberá ser extendido en los lugares indicados para que las extensiones del vendedor puedan ser descubiertas por una aplicación cliente que lo desee.

### A.2 OGC-Exception.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/ogc"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

   <xsd:element name="ServiceExceptionReport">
      <xsd:annotation>
         <xsd:documentation>
            The ServiceExceptionReport element contains one
            or more ServiceException elements that describe
            a service exception.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="ServiceException"
                         type="ogc:ServiceExceptionType"
                         minOccurs="0" maxOccurs="unbounded">
               <xsd:annotation>
                  <xsd:documentation>
                     The Service exception element is used to describe
                     a service exception.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:element>
         </xsd:sequence>
         <xsd:attribute name="version" type="xsd:string" fixed="1.2.0"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:complexType name="ServiceExceptionType">
      <xsd:annotation>
         <xsd:documentation>
```

```
            The ServiceExceptionType type defines the ServiceException
            element.  The content of the element is an exception message
            that the service wished to convey to the client application.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="code" type="xsd:string">
               <xsd:annotation>
                  <xsd:documentation>
                     A service may associate a code with an exception
                     by using the code attribute.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="locator" type="xsd:string">
               <xsd:annotation>
                  <xsd:documentation>
                     The locator attribute may be used by a service to
                     indicate to a client where in the client's request
                     an exception was encountered.  If the request included
                     a 'handle' attribute, this may be used to identify the
                     offending component of the request.  Otherwise the
                     service may try to use other means to locate the
                     exception such as line numbers or byte offset from the
                     begining of the request, etc ...
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:schema>
```

### A.3 WFS-Basic.xsd

```
<?xml version="1.0"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/wfs"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

   <!-- ============================================================
        Includes and Imports
        ============================================================ -->
   <xsd:import namespace="http://www.opengis.net/gml"
               schemaLocation="../../gml/2.1/feature.xsd"/>
   <xsd:import namespace="http://www.opengis.net/ogc"
               schemaLocation="../../filter/1.0.0/filter.xsd"/>

   <!-- ============================================================
        REQUEST MESSAGES
        ============================================================ -->
   <xsd:element name="GetCapabilities" type="wfs:GetCapabilitiesType">
      <xsd:annotation>
         <xsd:documentation>
            The GetCapapbilities element is used to request that a Web Feature
            Service generate an XML document describing the organization
            providing the service, the WFS operations that the service
            supports, a list of feature types that the service can operate
            on and list of filtering capabilities that the service support.
            Such an XML document is called a capabilities document.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   <xsd:element name="DescribeFeatureType" type="wfs:DescribeFeatureTypeType">
      <xsd:annotation>
         <xsd:documentation>
```

```
         The DescribeFeatureType element is used to request that a Web
         Feature Service generate a document describing one or more
         feature types.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="GetFeature" type="wfs:GetFeatureType">
   <xsd:annotation>
      <xsd:documentation>
         The GetFeature element is used to request that a Web Feature
         Service return feature instances of one or more feature types.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>

<!-- ==========================================================
     RESPONSE MESSAGES
     ========================================================== -->
<xsd:element name="FeatureCollection"
             type="wfs:FeatureCollectionType"
             substitutionGroup="gml:_FeatureCollection">
   <xsd:annotation>
      <xsd:documentation>
         This element is a container for the response to a GetFeature
         or GetFeatureWithLock (WFS-transaction.xsd) request.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>

<!-- ==========================================================
     COMMON ATTRIBUTES DOCUMENTATION
     ==========================================================
     VERSION:
        The version attribute is used to indicate to which version
        of the Web Feature Service Implementation Specification a
        request conforms.

     SERVICE:
        The service attribute is used to indicate which service
        should process an operation.  This attribute is particularly
        useful in the case where a single server implements multiple
        services (e.g. WMS, WFS, WCS, etc ...).

     HANDLE:
        The purpose of the handle attribute is to allow a client app
        to associate a mnemonic name with a request for error handling
        purposes.  If a "handle" is specified, and an exception occurs,
        a Web Feature Service may use the handle to identify the
        offending element.

     TYPENAME:
        The typeName attribute is used to specify the name of the
        feature type to be queried.  The term "feature type" is a
        term used by convention to refer to the container storing
        feature instances.  It does not mean type in the programmatic
        sense.  The typeName attribute should, instead, be thought
        of as the feature name. -->

<!-- ==========================================================
     TYPES
     ========================================================== -->
<!-- GETCAPABILITIES -->
<xsd:complexType name="GetCapabilitiesType">
   <xsd:annotation>
      <xsd:documentation>
         This type defines the GetCapabilities operation.  In response
         to a GetCapabilities request, a Web Feature Service must
         generate a capabilities XML document that validates against
         the schemas defined in WFS-capabilities.xsd.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:attribute name="version"
                  type="xsd:string" use="optional" fixed="1.0.0"/>
   <xsd:attribute name="service"
```

```
                  type="xsd:string" use="required" fixed="WFS"/>
   </xsd:complexType>
   <!-- DESCRIBEFEATURETYPE -->
   <xsd:complexType name="DescribeFeatureTypeType">
      <xsd:annotation>
         <xsd:documentation>
            The DescribeFeatureType operation allows a client application
            to request that a Web Feature Service describe one or more
            feature types.   A Web Feature Service must be able to generate
            feature descriptions as valid GML2 application schemas.

            The schemas generated by the DescribeFeatureType operation can
            be used by a client application to validate the output.

            Feature instances within the WFS interface must be specified
            using GML2.  The schema of feature instances specified within
            the WFS interface must validate against the feature schemas
            generated by the DescribeFeatureType request.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element name="TypeName" type="xsd:QName"
                      minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation>
                  The TypeName element is used to enumerate the feature types
                  to be described.  If no TypeName elements are specified
                  then all features should be described.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="version"
                     type="xsd:string" use="required" fixed="1.0.0"/>
      <xsd:attribute name="service"
                     type="xsd:string" use="required" fixed="WFS"/>
      <xsd:attribute name="outputFormat"
                     type="xsd:string" use="optional" default="XMLSCHEMA">
         <xsd:annotation>
            <xsd:documentation>
               The outputFormat attribute is used to specify what schema
               description language should be used to describe features.
               The default value of XMLSCHEMA means that the Web Feature
               Service must generate a GML2 application schema that can
               be used to validate the GML2 output of a GetFeature request
               or feature instances specified in Transaction operations.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
   </xsd:complexType>
   <!-- GETFEATURE -->
   <xsd:complexType name="GetFeatureType">
      <xsd:annotation>
         <xsd:documentation>
            A GetFeature element contains one or more Query elements
            that describe a query operation on one feature type.  In
            response to a GetFeature request, a Web Feature Service
            must be able to generate a GML2 response that validates
            using a schema generated by the DescribeFeatureType request.
            A Web Feature Service may support other possibly non-XML
            (and even binary) output formats as long as those formats
            are advertised in the capabilities document.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element ref="wfs:Query" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="version"
                     type="xsd:string" use="required" fixed="1.0.0"/>
      <xsd:attribute name="service"
                     type="xsd:string" use="required" fixed="WFS"/>
      <xsd:attribute name="handle"
                     type="xsd:string" use="optional"/>
      <xsd:attribute name="outputFormat"
```

```
                  type="xsd:string" use="optional" default="GML2">
         <xsd:annotation>
            <xsd:documentation>
               The outputFormat attribute is used to specify the output
               format that the Web Feature Service should generate in
               response to a GetFeature or GetFeatureWithLock element.
               The default value of GML2 indicates that the output is an
               XML document that conforms to the Geography Markup Language
               (GML) Implementation Specification V2.0.

               Other values may be used to specify other formats as long
               as those values are advertised in the capabilities document.
               For example, the value WKB may be used to indicate that a
               Well Known Binary format be used to encode the output.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="maxFeatures" type="xsd:positiveInteger"
                     use="optional">
         <xsd:annotation>
            <xsd:documentation>
               The maxFeatures attribute is used to specify the maximum
               number of features that a GetFeature operation should
               generate (regardless of the actual number of query hits).
            </xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
   </xsd:complexType>
   <xsd:element name="Query" type="wfs:QueryType">
      <xsd:annotation>
         <xsd:documentation>
            The Query element is used to describe a single query.
            One or more Query elements can be specified inside a
            GetFeature element so that multiple queries can be
            executed in one request.  The output from the various
            queries are combined in a wfs:FeatureCollection element
            to form the response to the request.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   <xsd:complexType name="QueryType">
      <xsd:annotation>
         <xsd:documentation>
            The Query element is of type QueryType.
         </xsd:documentation>
      </xsd:annotation>
     <xsd:sequence>
       <xsd:element ref="ogc:PropertyName" minOccurs="0" maxOccurs="unbounded">
          <xsd:annotation>
             <xsd:documentation>
                The PropertyName element is used to specify one or more
                properties of a feature whose values are to be retrieved
                by a Web Feature Service.

                While a Web Feature Service should endeavour to satisfy
                the exact request specified, in some instance this may
                not be possible.  Specifically, a Web Feature Service
                must generate a valid GML2 response to a Query operation.
                The schema used to generate the output may include
                properties that are mandatory.  In order that the output
                validates, these mandatory properties must be specified
                in the request.  If they are not, a Web Feature Service
                may add them automatically to the Query before processing
                it.  Thus a client application should, in general, be
                prepared to receive more properties than it requested.

                Of course, using the DescribeFeatureType request, a client
                application can determine which properties are mandatory
                and request them in the first place.
             </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
       <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1">
          <xsd:annotation>
```

```
            <xsd:documentation>
               The Filter element is used to define spatial and/or non-spatial
               constraints on query.  Spatial constrains use GML2 to specify
               the constraining geometry.  A full description of the Filter
               element can be found in the Filter Encoding Implementation
               Specification.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="handle"
                   type="xsd:string" use="optional"/>
    <xsd:attribute name="typeName"
                   type="xsd:QName" use="required"/>
    <xsd:attribute name="featureVersion"
                   type="xsd:string" use="optional">
       <xsd:annotation>
          <xsd:documentation>
             For systems that implement versioning, the featureVersion
             attribute is used to specify which version of a particular
             feature instance is to be retrieved.  A value of ALL means
             that all versions should be retrieved.  An integer value
             'i', means that the ith version should be retrieve if it
             exists or the most recent version otherwise.
          </xsd:documentation>
       </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>
  <!-- RESPONSE TYPE -->
  <xsd:complexType name="FeatureCollectionType">
     <xsd:annotation>
        <xsd:documentation>
           This type defines a container for the response to a
           GetFeature or GetFeatureWithLock request.  If the
           request is GetFeatureWithLock, the lockId attribute
           must be populated.  The lockId attribute can otherwise
           be safely ignored.
        </xsd:documentation>
     </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureCollectionType">
        <xsd:attribute name="lockId" type="xsd:string" use="optional">
           <xsd:annotation>
              <xsd:documentation>
                 The value of the lockId attribute is an identifier
                 that a Web Feature Service generates and which a
                 client application can use in subsequent operations
                 (such as a Transaction request) to reference the set
                 of locked features.
              </xsd:documentation>
           </xsd:annotation>
        </xsd:attribute>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

### A.4 WFS-Transaction.xsd

```
<?xml version="1.0"?>
<xsd:schema
   targetNamespace="http://www.opengis.net/wfs"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wfs="http://www.opengis.net/wfs"
   elementFormDefault="qualified">

   <!-- ============================================================
        Includes and Imports
        ============================================================ -->
   <xsd:include schemaLocation="WFS-basic.xsd"/>
```

```
<xsd:import namespace="http://www.opengis.net/gml"
            schemaLocation="../../gml/2.1/feature.xsd"/>
<xsd:import namespace="http://www.opengis.net/ogc"
            schemaLocation="../../filter/1.0.0/filter.xsd"/>

<!-- ============================================================
      REQUEST MESSAGES
      ============================================================ -->
<xsd:element name="GetFeatureWithLock" type="wfs:GetFeatureWithLockType">
   <xsd:annotation>
      <xsd:documentation>
         This is the root element for the GetFeatureWithLock request.
         The GetFeatureWithLock operation performs identically to a
         GetFeature request except that the GetFeatureWithLock request
         locks all the feature instances in the result set and returns
         a lock identifier to a client application in the response.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="LockFeature" type="wfs:LockFeatureType">
   <xsd:annotation>
      <xsd:documentation>
         This is the root element for a LockFeature request.
         The LockFeature request can be used to lock one or
         more feature instances.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="Transaction" type="wfs:TransactionType">
   <xsd:annotation>
      <xsd:documentation>
         This is the root element for a Transaction request.
         A transaction request allows insert, update and
         delete operations to be performed to create, change
         or remove feature instances.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>

<!-- ============================================================
      RESPONSE MESSAGES
      ============================================================ -->
<xsd:element name="WFS_LockFeatureResponse"
             type="wfs:WFS_LockFeatureResponseType">
   <xsd:annotation>
      <xsd:documentation>
         The WFS_LockFeatureResponse element contains a report
         about the completion status of a LockFeature request.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="WFS_TransactionResponse"
             type="wfs:WFS_TransactionResponseType">
   <xsd:annotation>
      <xsd:documentation>
         The WFS_TransactionResponse element contains a report
         about the completion status of a Transaction operation.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>

<!-- ============================================================
      COMMON ATTRIBUTE DOCUMENTATION
      ============================================================
      EXPIRY:
         The expiry attribute value is specified in minutes. It
         indicates how long a Web Feature Service should wait to
         receive a request from the client application that locked
         the feature instances.  If the specified time elapses and
         no request has been received by a Web Feature Service that
         references the lockId given to the client application, then
         the locks maintained by the Web Feature Service shall be
         released and the lockId that references the locked features
         shall now be invalid.  If the expiry attribute is not specified,
```

```
         then the feature instances shall be locked indefinitely and the
         intervention of an administrator may be required to release
         the locks. -->

   <!-- ============================================================
         TYPES
         ============================================================ -->
   <!-- GETFEATUREWITHLOCK -->
   <xsd:complexType name="GetFeatureWithLockType">
      <xsd:annotation>
         <xsd:documentation>
            A GetFeatureWithLock request operates identically to a
            GetFeature request expect that it attempts to lock the
            feature instances in the result set and includes a lock
            identifier in its response to a client.  A lock identifier
            is an identifier generated by a Web Feature Service that
            a client application can use, in subsequent operations,
            to reference the locked set of feature instances.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element ref="wfs:Query" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="version"
                     type="xsd:string" use="required" fixed="1.0.0"/>
      <xsd:attribute name="service"
                     type="xsd:string" use="required" fixed="WFS"/>
      <xsd:attribute name="handle"
                     type="xsd:string" use="optional"/>
      <xsd:attribute name="expiry"
                     type="xsd:positiveInteger" use="optional"/>
      <xsd:attribute name="outputFormat"
                     type="xsd:string" use="optional" default="GML2"/>
      <xsd:attribute name="maxFeatures"
                     type="xsd:positiveInteger" use="optional"/>
   </xsd:complexType>
   <!-- LOCKFEATURE -->
   <xsd:complexType name="LockFeatureType">
      <xsd:annotation>
         <xsd:documentation>
            This type defines the LockFeature operation.  The LockFeature
            element contains one or more Lock elements that define
            which features of a particular type should be locked.  A lock
            identifier (lockId) is returned to the client application which
            can be used by subsequent operations to reference the locked
            features.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element name="Lock" type="wfs:LockType" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation>
                  The lock element is used to indicate which feature
                  instances of particular type are to be locked.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="version"
                     type="xsd:string" use="required" fixed="1.0.0"/>
      <xsd:attribute name="service"
                     type="xsd:string" use="required" fixed="WFS"/>
      <xsd:attribute name="expiry"
                     type="xsd:positiveInteger" use="optional"/>
      <xsd:attribute name="lockAction"
                     type="wfs:AllSomeType" use="optional">
         <xsd:annotation>
            <xsd:documentation>
               The lockAction attribute is used to indicate what
               a Web Feature Service should do when it encounters
               a feature instance that has already been locked by
               another client application.

               Valid values are ALL or SOME.
```

```
            ALL means that the Web Feature Service must acquire
            locks on all the requested feature instances.  If it
            cannot acquire those locks then the request should
            fail.  In this instance, all locks acquired by the
            operation should be released.

            SOME means that the Web Feature Service should lock
            as many of the requested features as it can.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
</xsd:complexType>
<xsd:complexType name="LockType">
   <xsd:annotation>
      <xsd:documentation>
         This type defines the Lock element.  The Lock element
         defines a locking operation on feature instances of
         a single type. An OGC Filter is used to constrain the
         scope of the operation.  Features to be locked can be
         identified individually by using their feature identifier
         or they can be locked by satisfying the spatial and
         non-spatial constraints defined in the filter.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="handle"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="typeName"
                  type="xsd:QName" use="required"/>
</xsd:complexType>
<!-- TRANSACTION -->
<xsd:complexType name="TransactionType">
   <xsd:annotation>
      <xsd:documentation>
         The TranactionType defines the Transaction operation.  A
         Transaction element contains one or more Insert, Update
         Delete and Native elements that allow a client application
         to create, modify or remove feature instances from the
         feature repository that a Web Feature Service controls.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element ref="wfs:LockId" minOccurs="0">
         <xsd:annotation>
            <xsd:documentation>
               In order for a client application to operate upon locked
               feature instances, the Transaction request must include
               the LockId element.  The content of this element must be
               the lock identifier the client application obtained from
               a previous GetFeatureWithLock or LockFeature operation.

               If the correct lock identifier is specified the Web
               Feature Service knows that the client application may
               operate upon the locked feature instances.

               No LockId element needs to be specified to operate upon
               unlocked features.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
         <xsd:element ref="wfs:Insert"/>
         <xsd:element ref="wfs:Update"/>
         <xsd:element ref="wfs:Delete"/>
         <xsd:element ref="wfs:Native"/>
      </xsd:choice>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
   <xsd:attribute name="service"
                  type="xsd:string" use="required" fixed="WFS"/>
```

```
      <xsd:attribute name="handle"
                     type="xsd:string" use="optional"/>
      <xsd:attribute name="releaseAction"
                     type="wfs:AllSomeType" use="optional">
         <xsd:annotation>
            <xsd:documentation>
               The releaseAction attribute is used to control how a Web
               Feature service releases locks on feature instances after
               a Transaction request has been processed.

               Valid values are ALL or SOME.

               A value of ALL means that the Web Feature Service should
               release the locks of all feature instances locked with the
               specified lockId, regardless or whether or not the features
               were actually modified.

               A value of SOME means that the Web Feature Service will
               only release the locks held on feature instances that
               were actually operated upon by the transaction.  The lockId
               that the client application obtained shall remain valid and
               the other, unmodified, feature instances shall remain locked.
               If the expiry attribute was specified in the original operation
               that locked the feature instances, then the expiry counter
               will be reset to give the client application that same amount
               of time to post subsequent transactions against the locked
               features.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
   </xsd:complexType>
   <xsd:element name="LockId" type="xsd:string">
      <xsd:annotation>
         <xsd:documentation>
            The LockId element contains the value of the lock identifier
            obtained by a client application from a previous GetFeatureWithLock
            or LockFeature request.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   <xsd:element name="Insert" type="wfs:InsertElementType">
      <xsd:annotation>
         <xsd:documentation>
            The Insert element is used to indicate that the Web Feature
            Service should create a new instance of a feature type.  The
            feature instance is specified using GML2 and one or more
            feature instances to be created can be contained inside the
            Insert element.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   <xsd:complexType name="InsertElementType">
      <xsd:sequence>
         <xsd:element ref="gml:_Feature" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   </xsd:complexType>
   <xsd:element name="Update" type="wfs:UpdateElementType">
      <xsd:annotation>
         <xsd:documentation>
            One or more existing feature instances can be changed by
            using the Update element.  Changing a feature instance
            means that the current value of one or more properties of
            the feature are replaced with new values.  The Update
            element contains  one or more Property elements.  A
            Property element contains the name or a feature property
            who's value is to be changed and the replacement value
            for that property.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   <xsd:complexType name="UpdateElementType">
      <xsd:sequence>
         <xsd:element ref="wfs:Property" maxOccurs="unbounded" />
```

```
      <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1">
         <xsd:annotation>
            <xsd:documentation>
               The Filter element is used to constrain the scope
               of the update operation to those features identified
               by the filter.  Feature instances can be specified
               explicitly and individually using the identifier of
               each feature instance OR a set of features to be
               operated on can be identified by specifying spatial
               and non-spatial constraints in the filter.
               If no filter is specified, then the update operation
               applies to all feature instances.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="Delete" type="wfs:DeleteElementType">
   <xsd:annotation>
      <xsd:documentation>
         The Delete element is used to indicate that one or more
         feature instances should be removed from the feature
         repository.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:complexType name="DeleteElementType">
   <xsd:sequence>
      <xsd:element ref="ogc:Filter" minOccurs="1" maxOccurs="1">
         <xsd:annotation>
            <xsd:documentation>
               The Filter element is used to constrain the scope
               of the delete operation to those features identified
               by the filter.  Feature instances can be specified
               explicitly and individually using the identifier of
               each feature instance OR a set of features to be
               operated on can be identified by specifying spatial
               and non-spatial constraints in the filter.
               If no filter is specified then an exception should
               be raised since it is unlikely that a client application
               intends to delete all feature instances.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="Native" type="wfs:NativeType">
   <xsd:annotation>
      <xsd:documentation>
         Many times, a Web Feature Service interacts with a repository
         that may have special vendor specific capabilities.  The native
         element allows vendor specific command to be passed to the
         repository via the Web Feature Service.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:complexType name="NativeType">
   <xsd:attribute name="vendorId" type="xsd:string" use="required">
      <xsd:annotation>
         <xsd:documentation>
            The vendorId attribute is used to specify the name of
            vendor who's vendor specific command the client
            application wishes to execute.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
   <xsd:attribute name="safeToIgnore" type="xsd:boolean" use="required">
      <xsd:annotation>
         <xsd:documentation>
            In the event that a Web Feature Service does not recognize
```

```
            the vendorId or does not recognize the vendor specific command,
            the safeToIgnore attribute is used to indicate whether the
            exception can be safely ignored.  A value of TRUE means that
            the Web Feature Service may ignore the command.  A value of
            FALSE means that a Web Feature Service cannot ignore the
            command and an exception should be raised if a problem is
            encountered.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
</xsd:complexType>

<!-- define structure to specify a property value -->
<xsd:element name="Property" type="wfs:PropertyType">
   <xsd:annotation>
      <xsd:documentation>
         The Property element is used to specify the new
         value of a feature property inside an Update element.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:complexType name="PropertyType">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string">
         <xsd:annotation>
            <xsd:documentation>
               The Name element contains the name of a feature property
               to be updated.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:element name="Value" minOccurs="0">
         <xsd:annotation>
            <xsd:documentation>
               The Value element contains the replacement value for the
               named property.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
<!-- RESPONSE TYPES -->
<xsd:complexType name="WFS_LockFeatureResponseType">
   <xsd:annotation>
      <xsd:documentation>
         The WFS_LockFeatureResponseType is used to define an
         element to contains the response to a LockFeature
         operation.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element ref="wfs:LockId">
         <xsd:annotation>
            <xsd:documentation>
               The WFS_LockFeatureResponse includes a LockId element
               that contains a lock identifier.  The lock identifier
               can be used by a client, in subsequent operations, to
               operate upon the locked feature instances.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:element name="FeaturesLocked"
                   type="wfs:FeaturesLockedType" minOccurs="0">
         <xsd:annotation>
            <xsd:documentation>
               The LockFeature or GetFeatureWithLock operations
               identify and attempt to lock a set of feature
               instances that satisfy the constraints specified
               in the request.  In the event that the lockAction
               attribute (on the LockFeature or GetFeatureWithLock
               elements) is set to SOME, a Web Feature Service will
               attempt to lock as many of the feature instances from
               the result set as possible.
```

```
                The FeaturesLocked element contains list of ogc:FeatureId
                elements enumerating the feature instances that a WFS
                actually managed to lock.
             </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
       <xsd:element name="FeaturesNotLocked"
                    type="wfs:FeaturesNotLockedType" minOccurs="0">
          <xsd:annotation>
             <xsd:documentation>
                In contrast to the FeaturesLocked element, the
                FeaturesNotLocked element contains a list of
                ogc:Filter elements identifying feature instances
                that a WFS did not manage to lock because they were
                already locked by another process.
             </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeaturesLockedType">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element ref="ogc:FeatureId"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeaturesNotLockedType">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element ref="ogc:FeatureId"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="WFS_TransactionResponseType">
   <xsd:annotation>
      <xsd:documentation>
         The WFS_TransactionResponseType defines the format of
         the XML document that a Web Feature Service generates
         in response to a Transaction request.  The response
         includes the completion status of the transaction
         and the feature identifiers of any newly created
         feature instances.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element name="InsertResult"
                   type="wfs:InsertResultType"
                   minOccurs="0" maxOccurs="unbounded">
         <xsd:annotation>
            <xsd:documentation>
               The InsertResult element contains a list of ogc:FeatureId
               elements that identify any newly created feature instances.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:element name="TransactionResult"
                   type="wfs:TransactionResultType">
         <xsd:annotation>
            <xsd:documentation>
               The TransactionResult element contains a Status element
               indicating the completion status of a transaction.  In
               the event that the transaction fails, additional element
               may be included to help locate which part of the transaction
               failed and why.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="version"
                  type="xsd:string" use="required" fixed="1.0.0"/>
</xsd:complexType>
<xsd:complexType name="TransactionResultType">
   <xsd:sequence>
      <xsd:element name="Status" type="wfs:StatusType">
         <xsd:annotation>
            <xsd:documentation>
               The Status element contains an element indicating the
```

```
                  completion status of a transaction.  The SUCCESS element
                  is used to indicate successful completion.  The FAILED
                  element is used to indicate that an exception was
                  encountered.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="Locator" type="xsd:string" minOccurs="0">
            <xsd:annotation>
               <xsd:documentation>
                  In the event that an exception was encountered while
                  processing a transaction, a Web Feature Service may
                  use the Locator element to try and identify the part
                  of the transaction that failed.  If the element(s)
                  contained in a Transaction element included a handle
                  attribute, then a Web Feature Service may report the
                  handle to identify the offending element.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="Message" type="xsd:string" minOccurs="0">
            <xsd:annotation>
               <xsd:documentation>
                  The Message element may contain an exception report
                  generated by a Web Feature Service when an exception
                  is encountered.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   </xsd:complexType>
   <xsd:complexType name="InsertResultType">
      <xsd:sequence>
         <xsd:element ref="ogc:FeatureId" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="handle" type="xsd:string" use="optional"/>
   </xsd:complexType>
   <xsd:complexType name="StatusType">
      <xsd:choice>
         <xsd:element ref="wfs:SUCCESS"/>
         <xsd:element ref="wfs:FAILED"/>
         <xsd:element ref="wfs:PARTIAL"/>
      </xsd:choice>
   </xsd:complexType>
   <xsd:element name="SUCCESS" type="wfs:EmptyType"/>
   <xsd:element name="FAILED" type="wfs:EmptyType"/>
   <xsd:element name="PARTIAL" type="wfs:EmptyType"/>
   <!-- MISC TYPES -->
   <xsd:complexType name="EmptyType"/>
   <xsd:simpleType name="AllSomeType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="ALL"/>
         <xsd:enumeration value="SOME"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:schema>
```

### A.5 WFS-Capabilities.xsd

```
<?xml version="1.0" ?>
<xsd:schema
   targetNamespace="http://www.opengis.net/wfs"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

   <!-- Comments in this document may impose additional constraints
        beyond those codified in the schema syntax.  A conformant
        Web Feature Server must provide Capabilities XML that
```

```
        (1) validates against this schema
        (2) does not violate the constraints stated in
            comments herein. -->

<!-- ============================================================
     Imports ...
     ============================================================ -->
<xsd:import namespace="http://www.opengis.net/ogc"
            schemaLocation="../../filter/1.0.0/filterCapabilities.xsd" />

<!-- ============================================================
     Global elements and attributes
     ============================================================ -->
<!-- A descriptive narrative for more
     information about this server. -->
<xsd:element name="Abstract" type="xsd:string"/>
<!-- Elements containing text blocks indicating what
     fees or access constraints are imposed by the
     service provider on the service or data retrieved
     from the server. The reserved keyword "NONE"
     indicates no constraint exists. -->
<xsd:element name="AccessConstraints" type="xsd:string"/>
<xsd:element name="Fees" type="xsd:string"/>
<!-- Short words to help catalog searching.
     Currently, no controlled vocabulary has
     been defined. -->
<xsd:element name="Keywords" type="xsd:string"/>
<!-- The top-level HTTP URL of this service.
     Typically the URL of a "home page" for
     the service.  See also the onlineResource
     attributes of <DCPType> children, below.
     Currently, no non-HTTP platforms have been
     specified. -->
<xsd:element name="OnlineResource"/>
<xsd:element name="SRS" type="xsd:string"/>
<!-- A human-readable title to briefly identify
     this server in menus. -->
<xsd:element name="Title" type="xsd:string"/>

<xsd:element name="Query"  type="wfs:EmptyType"/>
<xsd:element name="Insert" type="wfs:EmptyType"/>
<xsd:element name="Update" type="wfs:EmptyType"/>
<xsd:element name="Delete" type="wfs:EmptyType"/>
<xsd:element name="Lock"   type="wfs:EmptyType"/>

<!-- REDEFINE THIS ELEMENT AS NEEDED IN YOUR XML  -->
<xsd:element name="VendorSpecificCapabilities" type="xsd:string"/>

<!-- ============================================================
     Root element
     ============================================================ -->
<!-- The parent element of the Capabilities document includes as
     children a Service element with general information about the
     server, a Capability element with specific information about
     the kinds of functionality offered by the server, a FeatureTypeList
     element defining the list of all feature types available from
     this server and a FeatureCapabilities element describing the
     filter capabilities of the server. -->
<xsd:element name="WFS_Capabilities" type="wfs:WFS_CapabilitiesType"/>

<!-- ============================================================
     Types
     ============================================================ -->
<xsd:complexType name="WFS_CapabilitiesType">
   <xsd:sequence>
      <!-- The Service element provides metadata for
           the service as a whole. -->
      <xsd:element name="Service" type="wfs:ServiceType"/>

      <!-- A Capability lists available request
           types, how exceptions may be reported, and
           whether any vendor-specific capabilities
           are defined.  It also lists all the
           feature types available from this feature
```

```
            server. -->
         <xsd:element name="Capability" type="wfs:CapabilityType"/>
         <xsd:element name="FeatureTypeList" type="wfs:FeatureTypeListType"/>
         <xsd:element ref="ogc:Filter_Capabilities" />
      </xsd:sequence>

      <!-- The version attribute specifies the specification revision
           to which this schema applies.  Its format is one,t two or three
           integers separated by periods: "x", or "x.y", or "x.y.z",
           with the most significant number appearing first.  Future
           revisions are guaranteed to be numbered in monotonically
           increasing fashion, though gaps may appear in the sequence. -->
      <xsd:attribute name="version"
                     type="xsd:string" fixed="1.0.0"/>

      <!-- The updateSequence attribute is a sequence number for
           managing propagation of the contents of this document.
           For example, if a Feature Server adds some data feature
           types it can increment the update sequence to inform
           catalog servers that their previously cached versions
           are now stale. The format is a positive integer. -->
      <xsd:attribute name="updateSequence"
                     type="xsd:nonNegativeInteger" default="0"/>
   </xsd:complexType>

   <xsd:complexType name="ServiceType">
      <xsd:sequence>
         <xsd:element name="Name" type="xsd:string"/>
         <xsd:element ref="wfs:Title"/>
         <xsd:element ref="wfs:Abstract" minOccurs="0"/>
         <xsd:element ref="wfs:Keywords" minOccurs="0"/>
         <xsd:element ref="wfs:OnlineResource"/>
         <xsd:element ref="wfs:Fees" minOccurs="0"/>
         <xsd:element ref="wfs:AccessConstraints" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="CapabilityType">
      <xsd:sequence>
         <xsd:element name="Request" type="wfs:RequestType"/>
         <!-- The optional VendorSpecificCapabilities element lists any
              capabilities unique to a particular server.  Because the
              information is not known a priori, it cannot be constrained
              by this particular schema document.  A vendor-specific schema
              fragment must be supplied at the start of the XML capabilities
              document, after the reference to the general WFS_Capabilities
              schema. -->
         <xsd:element ref="wfs:VendorSpecificCapabilities" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="FeatureTypeListType">
      <xsd:sequence>
         <xsd:element name="Operations"
                      type="wfs:OperationsType" minOccurs="0"/>
         <xsd:element name="FeatureType"
                      type="wfs:FeatureTypeType" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- Available WFS-defined request types are listed here. At
        least one of the values is required, but more than one
        may be given. -->
   <xsd:complexType name="RequestType">
      <xsd:choice maxOccurs="unbounded">
         <xsd:element name="GetCapabilities"
                      type="wfs:GetCapabilitiesType"/>
         <xsd:element name="DescribeFeatureType"
                      type="wfs:DescribeFeatureTypeType"/>
         <xsd:element name="Transaction"
                      type="wfs:TransactionType"/>
         <xsd:element name="GetFeature"
                      type="wfs:GetFeatureTypeType"/>
         <xsd:element name="GetFeatureWithLock"
                      type="wfs:GetFeatureTypeType"/>
```

```
      <xsd:element name="LockFeature"
                   type="wfs:LockFeatureTypeType"/>
   </xsd:choice>
</xsd:complexType>
<xsd:complexType name="GetCapabilitiesType">
   <xsd:sequence>
      <xsd:element name="DCPType"
                   type="wfs:DCPTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="DescribeFeatureTypeType">
   <xsd:sequence>
      <xsd:element name="SchemaDescriptionLanguage"
                   type="wfs:SchemaDescriptionLanguageType"/>
      <xsd:element name="DCPType"
                   type="wfs:DCPTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TransactionType">
   <xsd:sequence>
      <xsd:element name="DCPType"
                   type="wfs:DCPTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GetFeatureTypeType">
   <xsd:sequence>
      <xsd:element name="ResultFormat"
                   type="wfs:ResultFormatType"/>
      <xsd:element name="DCPType"
                   type="wfs:DCPTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="LockFeatureTypeType">
   <xsd:sequence>
      <xsd:element name="DCPType"
                   type="wfs:DCPTypeType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<!-- Available Distributed Computing Platforms (DCPs) are
     listed here.  At present, only HTTP is defined. -->
<xsd:complexType name="DCPTypeType">
   <xsd:sequence>
      <xsd:element name="HTTP" type="wfs:HTTPType"/>
   </xsd:sequence>
</xsd:complexType>

<!-- A list of feature types available from
     this server.  The following table
     specifies the number and source of the
     various elements that are available for
     describing a feature type.

     element             number    comments
     =================   =======   ======================================
     Name                    1     this is the Name of the feature type

     Title                 0/1     an optional Meaningful title for the
                                   feature type  (e.g. "Ontario Roads"
                                   for ROADL_1M")

     Abstract              0/1     optional; no Default

     Keywords              0/1     optional; no Default

     SRS                     1     the SRS that should be used
                                   when specifying the state of
                                   the feature

     Operations            0/1     a list of available operations for
                                   the feature type

     LatLongBoundingBox      1+    bounding box(s) of data
```

```
     MetadataURL           0/1+    optional; no default
-->
<xsd:complexType name="FeatureTypeType">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:QName"/>
      <xsd:element ref="wfs:Title" minOccurs="0"/>
      <xsd:element ref="wfs:Abstract" minOccurs="0"/>
      <xsd:element ref="wfs:Keywords" minOccurs="0"/>
      <xsd:element ref="wfs:SRS"/>
      <xsd:element name="Operations"
                   type="wfs:OperationsType" minOccurs="0"/>
      <xsd:element name="LatLongBoundingBox"
                   type="wfs:LatLongBoundingBoxType"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="MetadataURL"
                   type="wfs:MetadataURLType"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GetType">
   <xsd:attribute name="onlineResource" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- Available HTTP request methods. -->
<xsd:complexType name="HTTPType">
   <xsd:choice maxOccurs="unbounded">
      <!-- HTTP request methods.  The onlineResourc attribute
           indicates the URL prefix for HTTP GET requests
           (everything before the question mark and query string:
           http://hostname[:port]/path/scriptname); for HTTP POST
           requests, onlineResource is the complete URL. -->
      <xsd:element name="Get" type="wfs:GetType"/>
      <xsd:element name="Post" type="wfs:PostType"/>
   </xsd:choice>
</xsd:complexType>

<!-- The LatLongBoundingBox element is used to indicate the edges of
     an enclosing rectangle in the SRS of the associated feature type.
     Its purpose is to facilitate geographic searches  by indicating
     where instances of the particular feature type exist.  Since multiple
     LatLongBoundingBoxes can be specified, a WFS can indicate where
     various clusters of data may exist.  This knowledge aids client
     applications by letting them know where they should query in order
     to have a high probability of finding data. -->
<xsd:complexType name="LatLongBoundingBoxType">
   <xsd:attribute name="minx" type="xsd:string" use="required"/>
   <xsd:attribute name="miny" type="xsd:string" use="required"/>
   <xsd:attribute name="maxx" type="xsd:string" use="required"/>
   <xsd:attribute name="maxy" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- A Web Feature Server MAY use zero or more MetadataURL
     elements to offer detailed, standardized metadata about
     the data underneath a particular feature type.  The type
     attribute indicates the standard to which the metadata
     complies; the format attribute indicates how the metadata is
     structured.  Two types are defined at present:
     'TC211' = ISO TC211 19115;
     'FGDC'  = FGDC CSDGM. -->
<xsd:complexType name="MetadataURLType">
   <xsd:simpleContent>
      <xsd:extension base="xsd:string">
         <xsd:attribute name="type" use="required">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                  <xsd:enumeration value="TC211"/>
                  <xsd:enumeration value="FGDC"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:attribute>
         <xsd:attribute name="format" use="required">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                  <xsd:enumeration value="XML"/>
```

```
                  <xsd:enumeration value="SGML"/>
                  <xsd:enumeration value="TXT"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:attribute>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="OperationsType">
   <xsd:choice maxOccurs="unbounded">
      <xsd:element ref="wfs:Insert"/>
      <xsd:element ref="wfs:Update"/>
      <xsd:element ref="wfs:Delete"/>
      <xsd:element ref="wfs:Query"/>
      <xsd:element ref="wfs:Lock"/>
   </xsd:choice>
</xsd:complexType>
<xsd:complexType name="PostType">
   <xsd:attribute name="onlineResource" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- The ResultFormatType type defines the output formats that the
     web feature server can generate.  The mandatory format "GML2"
     must always be available.  Individual servers can define
     additional elements representing other output formats. -->
<xsd:element name="GML2" type="wfs:EmptyType"/>
<xsd:complexType name="ResultFormatType">
   <xsd:sequence maxOccurs="unbounded">
      <xsd:element ref="wfs:GML2"/>
   </xsd:sequence>
</xsd:complexType>

<!-- The SchemaDescriptionLanguageType type defines the schema languages
     that a feature server is capable of using to describe the schema
     of a feature.  Individual servers can define additional elements
     representing other schema languages but XMLSCHEMA must always
     be defined. -->
<xsd:element name="XMLSCHEMA" type="wfs:EmptyType"/>
<xsd:complexType name="SchemaDescriptionLanguageType">
   <xsd:sequence maxOccurs="unbounded">
      <xsd:element ref="wfs:XMLSCHEMA"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="EmptyType" />
</xsd:schema>
```

## Anexo B – Pruebas de conformidad (normativo)

Las pruebas de conformidad específicas para un *Web Feature Service* no han sido determinadas aún y serán agregadas en una revisión futura de esta especificación.

Por el momento, una implementación de WFS debe satisfacer las siguientes características del sistema para estar conforme con lo mínimo con esta especificación:

1. Se debe soportar las operaciones *GetCapabilities, DescribeFeatureType* y *GetFeature*.
2. El documento XML regresado en respuesta a una petición *GetCapabilities* debe ser válido contra la definición de esquema XML en el anexo A.4. Tal validación puede ser llevada a cabo usando herramientas disponibles comunes de validación XML.
3. En respuesta a la operación *GetFeature*, el WFS debe poder generar GML como uno de sus formatos de salida.
4. El documento XML regresado en respuesta a una petición *GetFeature* debe ser válida contra el esquema generado en respuesta a una petición *DescribeFeatureType*. Tal validación puede ser llevada a cabo usando herramientas disponibles comunes de validación XML.
5. Todas las cláusulas en las secciones normativas de esta especificación que usan las palabras: “debe”, “no debe”, “puede”, “no puede” deben ser satisfechas.