Objektkatalog für das Straßen- und Verkehrswesen

Datenbereitstellungs-Schnittstelle für OKSTRA-Daten auf Basis des OGC Web Feature Service

- *OKSTRA-WFS* -

Version:	1.0.2
Datum:	09.12.2008
Status:	Verabschiedet
Dateiname:	N0112.doc
Verantwortlich:	R. Erstling

OKSTRA-Pflegestelle **http://www.okstra.de/**

interactive instruments GmbH
Trierer Straße 70-72
53115 Bonn

Herr Bernd Weidner
Tel. 0228 91410 74
Fax 0228 91410 90
Email weidner@interactive-instruments.de

Im Auftrag von

Bundesanstalt für Straßenwesen
ZD - OKSTRA
Brüderstraße 53
51427 Bergisch Gladbach

Herr Alfred Stein
Tel. 02204 43 354
Fax 02204 43 673
Email stein@bast.de

0 Allgemeines

0.1 Inhaltsverzeichnis

1 Zweck des Dokuments (informativ)

1.1 Leserkreis

Das Dokument richtet sich an diejenigen Personen und Institutionen, die mit der Implementierung OKSTRA®-konformer Web-Services und deren Nutzung durch Client-Software befasst sind.

1.2 Kernaussagen des Inhalts

Es wird eine Datenbereitstellungsschnittstelle für OKSTRA®-Daten auf der Basis der Spezifikation des Web Feature Service (WFS) des Open Geospatial Consortiums (OGC) definiert. Die Definition erfolgt in zwei Stufen:

- OKSTRA-WFS lesend
- OKSTRA-WFS transaktional

Die zweite Stufe (OKSTRA-WFS transaktional) beinhaltet die erste und sieht auch den schreibenden Zugriff auf OKSTRA-Inhalte vor.

Die folgenden Kapitel sind zum Teil als *informativ* und zum Teil als *normativ* gegekennzeichnet. Nur die normativen Teile sind Bestandteil der Definition. Die informativen Aussagen dienen lediglich zur Veranschaulichung und Begründung des Vorgehens.

2 Gegenstand (normativ)

Dieses Dokument definiert eine Datenbereitstellungsschnittstelle für OKSTRA®-Daten auf der Basis der Spezifikation des Web Feature Service (WFS) des Open Geospatial Consortiums (OGC). Die definierte Datenbereitstellungsschnittstelle ist ein Web-Service zur Bereitstellung von OKSTRA®-Daten mit der Kurzbezeichnung OKSTRA®-WFS. Sie ermöglicht sowohl den lesenden als auch den schreibenden Zugriff. Die Definition erfolgt in zwei Stufen:

1. OKSTRA-WFS lesend
2. OKSTRA-WFS transaktional

Die zweite Stufe (OKSTRA-WFS transaktional) beinhaltet die erste und sieht auch den schreibenden Zugriff auf OKSTRA-Inhalte vor.

Die Definition ist Teil einer Gesamtspezifikation[1] zu „OKSTRA®-konformen Web-Services", also von Web-Services, welche generell OKSTRA®-Daten bereitstellen und verarbeiten. Neben der Datenbereitstellung gehören dazu auch höherwertige Schnittstellenfunktionen, welche über die reine Bereitstellung hinausgehend allgemein verwendbare „Geschäftlogik" des Straßen- und Verkehrsbereichs umfassen. Weiter zählen zur Gesamtspezifikation Schnittstellen für die Autorisierungskontrolle. Höherwertige Schnittstellenfunktionen und Autorisierungskontrolle sind in der vorliegenden Spezifikation nicht enthalten.

Der Zweck der Datenbereitstellungsschnittstelle für OKSTRA®-Daten ist die Kapselung der Verschiedenheiten der existierenden und zukünftigen Speicherungs- und Verarbeitungssysteme für Straßen- und Verkehrsdaten (Straßendatenbanken). Der Zugriff auf diese Systeme erfolgt über die vereinbarte Web-Schnittstelle auf der semantischen Basis des OKSTRA®-XML-Schemas.

Ziel der Definition ist auch Kompatibilität zu Geodateninfrastrukturen. Wenn auch nicht zu erwarten ist, dass das OKSTRA®-Schema als rein deutscher Standard die Basis europaweiter Schemafestlegungen, z.B. im Rahmen von INSPIRE werden wird, so wird es auf der Basis der dort auch eingesetzten Technologie für die Datenbereitstellung (also GML und WFS) leichter fallen, die benötigten Dienste für die Wandlung in die harmonisierten INSPIRE-Datenstrukturen bereitzustellen.

Der OKSTRA-WFS ist weitgehendst als ein Profil – d.h. als spezialisierende Untermenge – der OGC WFS-Spezifikation (Version 1.1.0) definiert. D.h. ein OKSTRA-WFS *ist* (weitgehendst) ein OGC WFS. Es gibt drei geringfügige Ausnahmen von dieser Regel, bei denen die WFS-Spezifikation erweitert und verändert wurde. Sie betreffen die Behandlung des XLink-WFS, Versionierung und die Zulassung eines „optimistischen" Locking-Schemas.

[1] Dies beschreibt einen Planungsstand. Zur Zeit der Schriftlegung dieses Dokuments existiert diese Gesamtspezifikation noch nicht.

3 Regelungen zur Konformität (normativ)

Dieses Kapitel beschreibt, wie für eine Instanz der OKSTRA®-Datenbereitstellungsschnittstelle (OKSTRA-WFS lesend oder OKSTRA-WFS transaktional) die Konformität mit den Definitionen in diesem Dokument und den normativen Bezügen hergestellt wird.

Es wurden bisher noch keine spezifischen Konformitätsprüfungen für die beiden Ebenen

- OKSTRA-WFS lesend und
- OKSTRA-WFS transaktional

entwickelt. Diese müssen aus der Erfahrung mit der vorliegenden Spezifikation in der Praxis entstehen und in die Spezifikation integriert werden. Eine Möglichkeit ist z.B. die Einrichtung einer automatischen Prozedur für eine Konformitätsprüfung. Eine minimale Konformität mit der vorliegenden Spezifikation erfordert die Beachtung der folgenden Punkte:

- Ein *OKSTRA-WFS lesend* muss (außer es ist in der vorliegenden Spezifikation anders definiert) konform sein zu einem OGC WFS basic Version 1.1.0, wie er in der normativen Referenz 4.2, ergänzt durch Referenz 4.1, beschrieben ist. Er muss dessen gesamte obligatorische Funktionalität aufweisen. Unter der obligatorische Funktionalität des OGC WFS basic wird hier die Funktionalität verstanden,
  - welche im Standard durch entsprechende Verbformen wie „shall" oder „must", etc. ausdrücklich als solche bezeichnet wird oder
  - für die keine Möglichkeit vorliegt, deren Fehlen im Capabilities-Response-Dokument zu beschreiben.
- Ein *OKSTRA-WFS lesend* muss alle zutreffenden verbindlichen Regelungen der normativen Kapitel der vorliegenden Spezifikation (außer Kapitel 9) erfüllen, insbesondere alle Regelungen des Kapitels 8. Verbindliche Regelungen werden durch die Verbformen „muss" oder „soll" ausgedrückt. Konjunktivische Formen wie „sollte" dagegen drücken Empfehlungen aus.
- Abweichend von der Regelung in der Spezifikation des OGC WFS basic Version 1.1.0 implementiert ein *OKSTRA-WFS lesend* zusätzlich einen Teil des XLink WFS. Dieser Teil ist für den OKSTRA-WFS abweichend von der Definition des XLink WFS spezifiziert (kein GetGmlObject erforderlich, Anhängen der zusätzlichen Features an das Ende der FeatureCollection). Zu den Details sind 8.1.3.4 und 8.1.3.12 zu beachten.
- Abweichend von der Regelung in der Spezifikation des OGC WFS basic Version 1.1.0 implementiert ein *OKSTRA-WFS lesend* das Attribut featureVersion am Query-Element im Sinne der Beachtung von Stichtagen. Zu den Details ist 8.1.3.9 zu beachten.
- Ein *OKSTRA-WFS transaktional* muss (außer es ist in der vorliegenden Spezifikation anders definiert) konform sein zu einem OGC WFS transactional Version 1.1.0 wie er in der normativen Referenz 4.2, ergänzt durch Referenz 4.1, beschrieben ist. Er muss dessen gesamte obligatorische Funktionalität aufweisen. Die obligatorische Funktionalität des OGC WFS transactional ist wie oben beim OGC WFS basic zu verstehen.
- Ein *OKSTRA-WFS transaktional* muss zusätzlich konform sein zu einem *OKSTRA-WFS lesend*.
- Ein *OKSTRA-WFS transaktional* muss alle zutreffenden verbindlichen Regelungen der normativen Kapitel der vorliegenden Spezifikation erfüllen, insbesondere alle Regelungen der Kapitel 8 und 9. Verbindliche Regelungen werden durch die Verbformen „muss" oder „soll" ausgedrückt. Konjunktivische Formen wie „sollte" dagegen drücken Empfehlungen aus.
- Abweichend von der Regelung in der Spezifikation des OGC WFS transactional Version 1.1.0 darf ein *OKSTRA-WFS transaktional* „optimistisches" Locking implementieren. Wenn davon Gebrauch gemacht wird, sind 8.1.1.3, 9.1.4 und 9.1.5 zu beachten.

4 Normative Bezüge (normativ)

Die *Datenbereitstellungs-Schnittstelle für OKSTRA®-Daten auf der Basis des OGC Web Feature Service*, kurz *OKSTRA®-WFS*, beruht auf den im Folgenden aufgeführten Standards.

Soweit die hier gelisteten Standardisierungs-Dokumente mit einem Veröffentlichungsdatum oder einer Version versehen sind, so gelten die im vorliegenden Dokument definierten Spezifikationen nur für die so bezeichnete Ausgabe des jeweiligen Standards und nicht für spätere Fortschreibungen. Nutzer der vorliegenden Spezifikation für den *OKSTRA®-WFS* werden jedoch dazu ermutigt, zu untersuchen, inwieweit die neuesten Ausgaben der bezeichneten Standards anwendbar sind, ohne dass die vorliegende Spezifikation Änderungen erfahren müsste.

Soweit die gelisteten Standardisierungs-Dokumente ohne Version oder zeitliche Festlegung existieren, gelten immer die neuesten Versionen.

4.1 OGC-FE, 2005

"OpenGIS® Filter Encoding Implementation Specification", Version 1.1.0, Open Geospatial Consortium (OGC), 2005-05-03, Ref.No OGC 04-095

4.2 OGC-WFS, 2005

"OpenGIS® Web Feature Service (WFS) Implementation Specification", Version 1.1.0, Open Geospatial Consortium (OGC), 2005-05-03, Ref.No OGC 04-094

4.3 OGC-URN-Id, 2007

"Definition identifier URNs in OGC namespace", Best Practice Paper - Version 1.1.2, Open Geospatial Consortium (OGC), 2007-09-28, Ref.No OGC 07-92r1

4.4 EPSG, 2007

"EPSG Geodetic Parameter Registry", Version: 6.14, OGP Surveying and Positioning Committee, 2007, http://www.epsg-registry.org/

4.5 OKSTRA®, 2007

"Objektkatalog für das Straßen- und Verkehrswesen", Version: 1.012, Bundesanstalt für Straßenwesen / OKSTRA®-Pflegestelle, 2007, http://www.okstra.de

5 Glossar (normativ)

Dieses Kapitel definiert Begriffe, die in dieser Spezifikation verwendet werden.

Capabilities

Ein Client erfragt ein Capabilities-Dokument von einem Web-Service über eine spezielle getCapabilites-Operation. Ein solches Dokument enthält Metadaten, die den Service beschreiben und spezifische Charakteristika des betroffenen Service.

Client, Web-Client

Softwarekomponente, welche einen Web-Service benutzt.

Feature

Ein Feature ist im Zusammenhang mit dem OKSTRA-WFS ein OKSTRA-Objekt, bzw. im Falle von Historisierung eine OKSTRA-Objektversion, d.h. ein Objekt mit einem bestimmten Lebenszeitintervall. Ein Feature ist eine Ausprägung eines Feature-Types.

Feature-Type

Ein Feature-Type definiert im vorliegenden Zusammenhang die gemeinsame Struktur von OKSTRA-Objekten. Alle Features besitzen einen Feature-Type.

Koordinatenreferenzsystem

Eine Definition bezüglich der Interpretation von Koordinaten (Vektoren von Zahlen) als Positionen in der Welt.

Operation

Angabe einer Aktion oder einer Abfrage an einen Service, welcher dieser ausführen soll.

Register

Ein Register ist der (Metadaten-)Inhalt einer Registry. Oft wird das Wort verwendet, um einen abgegrenzten Teil dieses Inhalts zu beschreiben.

Registry

Eine Registry ist eine Komponente (ein Web-Service), welche die Daten eines Registers zur Verfügung stellt.

Request

Ein Client ruft eine Operation eines Web-Service auf. Das übermittelte Datenpaket wird als Request bezeichnet.

Response

Das Resultat einer Operation. Der Service gibt den Response an den Client zurück.

Service, Web-Service

Eine Komponente, die über das Netz erreichbar ist und eine festgelegte Schnittstelle von Operationen bereitstellt, die von Clients aufgerufen werden können.

6 Überblick über OGC WFS (informativ)

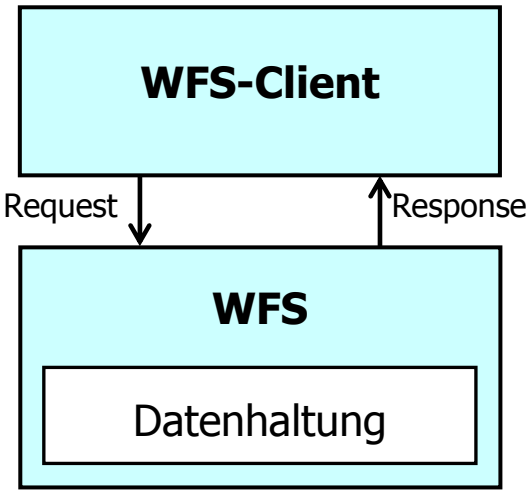
Dieses Kapitel gibt einen kurzen Überblick über OGC WFS 1.1 und Filter Encoding.

Der OGC Web Feature Service (WFS) stellt eine Schnittstelle zum Zugriff und für Veränderungsoperationen auf Daten mit Raumbezug zur Verfügung. Der Zugriff erfolgt wie bei allen Web-Services über http, das Protokoll des World-Wide-Web.

Die Schnittstelle des WFS dient zur Beschreibung, Abfrage, Erzeugung, Fortschreibung und Lösung von Daten, d.h. es handelt sich um das Äquivalent zu einer Datenbankschnittstelle (vergleichbar zu SQL, aber nicht ganz so leistungsfähig) für den Betrieb als Web-Service. Die Strukturierung der Daten erfolgt dabei gemäß einem Applikationsschema in OGC Geography Markup Language (GML), einer XML Sprache für räumliche Daten.

Für den OKSTRA® liegt eine solche Strukturierung in Form eines GML-Applikationsschemas vor, es nennt sich OKSTRA® XML. OKSTRA®-Daten sind daher für die Abfrage und Pflege durch einen Web Feature Server (eine Software, die einen Web Feature Service realisiert) geeignet. Die OKSTRA-Objekte werden in diesem Zusammenhang als „Features" bezeichnet.

Sowohl die Abfragesprache, welche durch den OGC Filter Encoding Standard gegeben ist, als auch die zurückgebrachten Abfrageergebnisse (die ermittelte Menge von „Features") beziehen sich auf die Strukturierung der Daten in GML, bzw. sind in dieser Form strukturiert. Dies ist unabhängig davon, wie die konkrete Haltung der Daten „hinter" oder „unter" oder „im" WFS beschaffen ist. Das bedeutet, der WFS kapselt und abstrahiert die tatsächliche Natur dieses Speichers und liefert nach außen nur eine Sicht auf die Daten als das GML-Applikationsschema, also OKSTRA® XML.



Abfragen (Queries) folgen dem Filter Encoding Standard. Er erlaubt es, wie in anderen Abfragesprachen, z.B. SQL, Datenelemente anzusprechen und durch Vergleiche und logische Kombinationen zu selektieren. Eine Besonderheit beim Filter Encoding sind die Räumlichen Operatoren, welche Bedingungen auf geometrischen Eigenschaften beschreiben. Sie können zusammen mit „normalen" Vergleichen eingesetzt werden, so dass es z.B. möglich wird, bestimmte Objekte in der Nähe von anderen Objekten zu ermitteln.

Der Zugriff auf die Datenelemente erfolgt beim Filter Encoding natürlich in Bezug auf das GML-Applikationsschema, d.h. durch Bezugnahme auf OKSTRA XML. Infolge der starken Strukturierung der OKSTRA-Daten ist es dabei wesentlich, dass dabei assoziative Verknüpfungen der OKSTRA-Objekte einfließen können.

Ein Web Feature Service umfasst die folgenden Operationen:

Web Feature Service	
Operation	Beschreibung
GetCapabilities	Die Fähigkeiten eines WFS werden in einem umfangreichen XML-Dokument beschrieben.
DescribeFeature	Durch diese Operation wird das GML-Schema für die angegebenen Feature-Types abgefragt. Durch das Schema weiß der Client, welche Eigenschaften ein Feature hat.
GetFeature	Die Operation erfragt die Inhalte eines oder mehrerer Features. Die Auswahl der Features erfolgt anhand des Feature-Types und (optional) eines Filter-Ausdrucks, der es gestattet, Features nach inhaltlichen Kriterien zu selektieren. Die Features werden ausgedrückt in GML (Geography Markup Language) zurückgebracht.
GetGmlObject	Diese Operation erfragt identifizierbare GML-Objekte über ihren Identifier. Sie hat für das hier zu beschreibende Profil keine Bedeutung.
Transaction	In einer Transaction können beliebig viele Features erzeugt, verändert oder gelöscht werden. Eine Transaction wirkt als Einheit, d.h. ganz oder gar nicht. Sollen Features mit „Locks" an einer Transaction teilnehmen, so muss die zugehörige „Lock-Id" angegeben werden.
LockFeature	Durch diese Operation werden „Locks" auf durch Filter selektierten Features genommen. Features, auf denen ein Lock besteht, können nicht durch andere Clients verändert werden. Es können auch keine weiteren Locks für diese genommen werden.
GetFeatureWithLock	Wie GetFeature. Die Operation nimmt aber zusätzlich ein „Lock" auf den ausgewählten Features, das sicherstellt, dass diese bis zu einer nachfolgenden Transaction-Operation, die sich auf das Lock bezieht, nicht durch andere Clients verändert werden kann.

Web Feature Server, die nur die Operationen „GetFeature", „DescribeFeature" und „GetCapabilities" unterstützen, sind zulässig. Sie eignen sich nur für lesenden Zugriff und werden „Basic WFS" genannt.

Ein WFS, der auch schreiben kann und deshalb die Operation „Transaction" unterstützt, heißt „Transactional WFS".

Daneben gibt es noch die Stufe „XLink WFS". Für diese ist die Operation „GetGmlObject" relevant. Diese Ausbaustufe ist für den OKSTRA-WFS nur zum Teil wichtig.

7 Entscheidungsgrundlagen (informativ)

Aufgrund der Struktur des OKSTRA® und der bekannten Anforderungen ergeben sich einige Problemfelder, welche an dieser Stelle analytisch behandelt werden. Das Kapitel dient der Begründung der in den beiden folgenden normativen Kapiteln getroffenen Festlegungen.

Hinweis zur Terminologie

Bezüglich des Begriffs „Objekt" gibt es im Zusammenhang mit der OKSTRA-Historisierung eine gewisse sprachliche Unschärfe und eine Abweichung von der normalen Verwendung im IT-Bereich.

Unter einem „Objekt" wird beim OKSTRA die gesamte sich zeitlich in ihren Eigenschaften verändernde Einheit verstanden (etwa eine bestimmte Straße während ihres Bestehens), während eine der Ausprägungen zu einer Zeit (oder in einem Zeitintervall) als „Version" bezeichnet wird.

Nun ist es aber so, dass die eigentlichen im Schema modellierten Einheiten eben die Versionen sind – die Objekte im obigen Sinne kommen darin gar nicht vor. Dies ist im Gegensatz zum normalen Gebrauch bei objektorientierter Modellierung, wo die Instanzen von Objektarten eben Objekte genannt werden und nicht Versionen.

Um verständlich zu bleiben, werden wir im folgenden unter „Objekt" die normale Bedeutung im IT-Bereich verstehen, also die Ausprägung (die Instanz) einer Objektart. Nur wenn wir über Historisierung reden, wenden wir die genauen Begriffe an.

7.1 Anforderungen an Filterausdrücke

Dieser Abschnitt analysisiert die Anforderungen, die sich durch das OKSTRA-Schema bezüglich der WFS-Filtersprache ergeben.

7.1.1 Die Filtersprache

Die von der WFS-Spezifikation vorgesehene Filtersprache, dargelegt in der Filter Encoding Spezifikation mit einigen Erweiterungen durch die WFS-Spezifikation, reicht im Großen und Ganzen aus, um die Anforderungen an wirkungsvolle Queries gegen einen OKSTRA-Bestand zu befriedigen.

Vorhanden sind im aktuellen Filter Encoding Standard[2]:

- Zugriffe auf Eigenschaften der Objekte,
- Konstanten,
- Ausdrücke, gebildet aus den Grundrechenarten und Funktionsaufrufen,
- Vergleiche (=, <>, <, <=, >, >=), einschließlich Intervallabfrage (Between), Mustervergleich (Like) und NULL-Vergleich (IsNull),
- Räumliche Operatoren (BBOX, Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains, DWithin, Beyond),
- Und, Oder, Nicht
- Spezielle Abfragen nach Objektidentifikatoren

[2] In der nächsten Version des Filter Encoding kommen weitere hinzu, z.B. zeitliche Operatoren.

Die Funktionsaufrufe sind im Standard nicht weiter ausspezifiziert und können deshalb für die nötigen Erweiterungen ausgestaltet werden. Hierauf wird bei der Betrachtung der einzelnen Problemfelder eingegangen werden.

Erfahrungsgemäß werden alle Elemente der Filtersprache tatsächlich benötigt. Allenfalls bei den räumlichen Operatoren können evtl. einige seltener benötigte entfallen. Benötigt werden auf alle Fälle BBOX, Disjoint, Within, Intersects, Contains, und DWithin, da sie verschiedene Varianten des Enthaltenseins, bzw. Nichtenthaltenseins in Bezug auf Geometrievorgaben ausdrücken. Zusätzlich ist Overlaps eine häufig eingesetzte Relation und sollte vorhanden sein.

Anforderung

Die Filtersprache soll mit Ausnahme der Geometrie-Operatoren vollumfänglich unterstützt werden. Mindestens folgende Geometrie-Operatoren sollen vorhanden sein: BBOX, Disjoint, Within, Intersects, Contains, DWithin.

7.1.2 Eigenschaftszugriffe

Wirkliche Erweiterungen gegenüber der Spezifikation sind im Zusammenhang mit den Zugriffsmöglichkeiten auf die Eigenschaften der Objekte erforderlich.

Der Grund liegt darin, dass das OKSTRA-Schema, der administrativen Wirklichkeit folgend, in den meisten Bereichen relativ komplex ist. Es gibt eine Vielzahl von Objektarten und relationale Bezüge zwischen diesen, welche zu beachten sind.

Bei der Abfassung von Queries bezüglich einer Objektart (FeatureType), wie es durch die WFS-Operation GetFeature geschieht, besteht deshalb nur in den einfachsten Fällen die Möglichkeit, sich auf die direkten Eigenschaften der erfragten Objektart zu beschränken.

Einfaches Beispiel:

> Finde alle Abschnitte, die zu Autobahnen gehören.

Um herauszufinden, zu welcher Straßenklasse ein *Abschnitt* gehört, muss man vom *Abschnitt* ausgehend der Relation *gehoert_zu_Strasse*, in der *Strasse* der Relation *hat_Strassenbezeichnung* und in der *Strassenbezeichnung* der Relation *Strassenklasse* folgen. In der *Strassenklasse* ist die Eigenschaft *Kennung* auszuwerten (für Autobahnen = „A").

Das Query-Konstrukt in der WFS GetFeature-Operation sieht folgendermaßen aus[3]:

```
<wfs:Query typeName='Abschnitt'>
 <ogc:Filter>
  <ogc:PropertyIsEqualTo>
  <ogc:PropertyName>gehoert_zu_Strasse/Strasse/hat_Strassenbezeichnung/
   Strassenbezeichnung/Strassenklasse/Strassenklasse/Kennung</ogc:PropertyName>
  <ogc:Literal>A</ogc:Literal>
  </ogc:PropertyIsEqualTo>
 </ogc:Filter>
</wfs:Query>
```

Die beim Element <ogc:PropertyName> verwendete Xpath-Notation ist zwar im Filter-Encoding- und im WFS-Standard vorgesehen, aber es ist in der aktuellen Version 1.1 der Standards nicht explizit spezifiziert, dass damit die Eigenschaften anderer Objektarten angesprochen werden können.

[3] Das Beispiel vernachlässigt die Historisierung. Siehe zu diesem Thema die Erörterung weiter unten.

Dies muss jedoch auf alle Fälle gefordert werden, um OKSTRA-Objekte sinnvoll inhaltlich erfragen zu können. Der AFIS-ALKIS-ATKIS-Standard der deutschen Vermessungsverwaltungen verwendet ebenfalls diese Regelung.

Anforderung

Die Xpath-Notation für Eigenschaftszugriffe (PropertyName) soll es gestatten, Relationen zu anderen Objektarten (FeatureTypes) zu verfolgen, auch wenn diese im OKSTRA XML Schema durch eine Referenz (xlink:href) ausgedrückt sind. Ein solcher Zugriff repräsentiert in seiner Gesamtheit die Menge der so vom Objekt (Feature) aus erreichten Eigenschaftswerte.

Folgeversionen des WFS werden aller Voraussicht nach diese erweiterte Interpretation der Xpath-Notation für den Eigenschaftszugriff in einer abgewandelten Syntax unterstützen.

7.1.3 Behandlung multipler Eigenschaften

Unabhängig davon, ob man gestattet, bei Eigenschaftszugriffen Relationen zu anderen Objekten zu verfolgen (aber besonders dann) muss man mit dem Problem umgehen, dass die angesprochenen Eigenschaften multipel vorliegen. Es ist zu klären, wie z.B. ein multipler Wert in einem Vergleich wirkt.

Zwar sieht der WFS-Standard hier eine Prädikatnotation vor, die es gestattet, das erste, zweite oder n-te Element einer Menge von Eigenschaften zu selektieren. Dies ist jedoch im Falle des OKSTRA oft nicht sinnvoll einsetzbar, da viele der multiplen Eigenschaften keine festgelegte Anordung aufweisen. In einigen Fällen ist das aber doch so, und dort ist der Einsatz der Positionsnotation sehr hilfreich. Außerdem ist die Notation Teil des Standards und sollte deshalb unterstützt werden.

Ein Beispiel für den nützlichen Einsatz dieser Positionsprädikate ist z.B. die Selektion eines Teilabschnitts in einer Strecke.

Anforderung

Die vom WFS-Standard geforderten Positionsprädikate in Xpath-Ausdrücken werden unterstützt.

Wir schlagen zusätzlich vor, bei multiplen Eigenschaftszugriffen die verwendende Operation im Sinne einer Existenzquantifizierung zu interpretieren. Die Lesart bei einem Vergleich wäre dann: Es gibt einen der multiplen Werte, für den der Vergleich zutrifft. Der AFIS-ALKIS-ATKIS-Standard der deutschen Vermessungsverwaltungen verwendet ebenfalls diese Regelung.

Bei Geometrie-Operatoren sollte man das genauso handhaben. Im Allgemeinen entspricht dies dem geometrischen Vergleich mit der Vereinigung der Einzelgeomentrien der Menge. Dies trifft aber nicht für alle Operationen zu, z.B. kann Contains auf die Vereinigung zutreffen aber auf keine der Einzelgeometrien.

Anforderung

Wenn Eigenschaftzugriffe (PropertyName) zu einer Menge von Eigenschaften führen, so ist diese Menge in der umgebenden Operation im Sinne einer Existenzquantifizierung zu interpretieren.

Die nächste WFS-Version wird es gestatten, das Verhalten bei Mengen explizit zu steuern.

7.1.4 Prädikate in Eigenschaftszugriffen

Für Teilbereiche des OKSTRA sind noch weitergehende Anforderungen an die Eigenschaftszugriffe zu stellen. Sie benötigen über die oben geforderte Interpretation der Xpath-Ausdrücke hinaus die Anwendbarkeit von zusätzlichen Prädikaten in den einzelnen Pfadpositionen.

Ein Beispiel aus dem Bereich Verkehrsdaten:

Es seien Zählstellen (im Beispiel der Objektart *automatische_Dauerzaehlstelle*) zu erfragen, welche für ein bestimmtes *Bezugsjahr* und für die *Fahrzeugart* Pkw einen DTV-Wert von 100000 überschreiten.

Dies kann eigentlich nur durch ein Query folgender Form sein:

```
<wfs:Query typeName='automatische_Dauerzaehlstelle'>
 <ogc:Filter>
  <ogc:PropertyIsGreaterThan>
   <ogc:PropertyName>zu_DTV/DTV[Bezugsjahr=2005 and Fahrzeugart/Fahrzeugart/Kennung='Pkw']/
    Fahrzeuge_pro_24h</ogc:PropertyName>
   <ogc:Literal>100000</ogc:Literal>
  </ogc:PropertyIsGreaterThan>
 </ogc:Filter>
</wfs:Query>
```

Der Versuch, dies mit konventionellen Mitteln anders auszudrücken, etwa durch ein AND-Konstrukt, bei dem der Vergleich und das Prädikat verknüpft werden, bewirkt nicht dasselbe.

```
<wfs:Query typeName='automatische_Dauerzaehlstelle'>
 <ogc:Filter>
  <ogc:And>
   <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>zu_DTV/DTV/Bezugsjahr</ogc:PropertyName>
    <ogc:Literal>2005</ogc:Literal>
   </ogc:PropertyIsEqualTo>
   <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>zu_DTV/DTV/Fahrzeugart/Fahrzeugart/Kennung</ogc:PropertyName>
    <ogc:Literal>Pkw</ogc:Literal>
   </ogc:PropertyIsEqualTo>
   <ogc:PropertyIsGreaterThan>
    <ogc:PropertyName>zu_DTV/DTV/Fahrzeuge_pro_24h</ogc:PropertyName>
    <ogc:Literal>100000</ogc:Literal>
   </ogc:PropertyIsGreaterThan>
  </ogc:And>
 </ogc:Filter>
</wfs:Query>
```

Dieser Filterausdruck löst die gegebene Selektionsaufgabe nicht, da die drei Komponenten des *And* voneinander unabhängige Ergebnisse bezüglich der DTV-Objekte liefern[4].

Anforderung

Die Xpath-Ausdrücke in Eigenschaftzugriffen (PropertyName) sollen Prädikate in Sinne der Xpath-Syntax unterstützen. Die Unterstützung ist erforderlich auf den Pfadpositionen, welche den OKSTRA-Objekten (Features) entsprechen. Die Ausdruckssyntax soll umfassen: Xpath-Eigenschaftszugriffe, Konstanten, Vergleiche, and, or, not().

Prädikate erleichtern im Übrigen die Behandlung multipler Eigenschaftszugriffe und sind unbedingt erforderlich, wenn die Historisierung im weiter unten vorgeschlagenden Sinne gelöst werden soll.

7.2 Implizite geometrische Eigenschaften

Objekte, welche dem Straßennetz zugeordnet sind, erhalten diesen Netzbezug typischerweise durch „Lineare Referenzierung", im OKSTRA als Stationierung bezeichnet. Stationierte Objekte treten als OKSTRA-Objektarten auf, welche Bezüge auf die Abschnitte oder Äste besitzen und gleichzeitig angeben, wo entlang der Geometrie des referierten Abschnitts oder Asts sie positioniert sind.

Bei punktförmigen Bezügen ist das einfach und wird über das Straßenpunkt-Konstrukt hergestellt.

Bei linearen Bezügen, gibt es die erste und einfache Variante, die sich nur auf einen Abschnitt oder Ast bezieht. Hier kommt das Teilabschnitt-Konstrukt zum Tragen, das zum Abschnitt oder Ast einfach zwei Stationen angibt. Die zweite Variante bezieht sich auf die lineare Folge zusammenhängender Teilabschnitte, die zu einer sog. Strecke zusammengefasst sind.

Alle diese Konstruktionen sind in OKSTRA XML definiert und können in Filterausdrücken abgefragt werden.

Durch die Stationierung erhält das stationierte Objekt implizit eine Geometrie, die (bis auf mögliche seitliche Auslenkung) mit Teilen der Netzgeometrie übereinstimmt. Es ist wünschenswert, diese implizit definierte Geometrie bei Queries nutzbar zu machen. Dies soll in diesem Abschnitt diskutiert werden. Um ein Beispiel zu geben: Man will alle Verkehrszeichentraeger-Objekte (das sind Punktobjekte) in einer vorgegebenen Fläche finden.

Implizite Geometrie kommt aber im OKSTRA nicht nur infolge der Stationierung vor. Z.B. wird durch die Abschnitte und Äste implizit eine Geometrie der Strassenobjekte gebildet. Weitere Beispiele für implizite Geometrie im OKSTRA finden sich insbesondere im Schema „Entwurf" (Achselement, Deckenbuch etc.).

[4] Über die Einzelkomponenten hat man zunächst drei Ergebnismengen von automatischen Dauerzählstellen:

1. Solche, die DTVs mit dem Bezugsjahr 2005 besitzen.
2. Solche, die DTVs für die Fahrzeugart „PkW" besitzen.
3. Solche, die DTVs mit einem „Fahrzeuge_pro_24h"-Wert von mehr als 100000 besitzen.

Über den AND-Operator gelangen diejenigen automatischen Dauerzählstellen in die Ergebnismenge des gesamten Filters, die in allen Ergebnismengen der einzelnen Komponenten auftreten. Leider ist damit nicht sichergestellt, dass alle geforderten Eigenschaften im selben DTV auftreten – sie können auch auf unterschiedliche DTVs verteilt sein. Insofern liefert diese Query potenziell zu vielle Dauerzählstellen zurück.

Es sind hier im Grunde drei Alternativen denkbar, die leider alle ihre Vor- und Nachteile aufweisen.

7.2.1 Alternative 1: Eine Filter-Funktion zur Berechnung der Geometrie

Wie in 7.1.1 bereits angemerkt, besitzt die Filtersprache ein Konstrukt für sog. Functions. Diese sind im Filter Encoding Standard nicht festgelegt, ein WFS-Profil kann jedoch davon Gebrauch machen und spezielle *Functions* definieren und fordern.

Es wäre in diesem Falle eine *Function* zu definieren, welche bei Versorgung mit dem entsprechenden Stationierungskonstrukt (also Straßenpunkt oder Teilabschnitt oder Strecke) und – im Falle der Historisierung – einem zusätzlichen Bezugsdatum die Berechnung der gewünschten Geometrie vornimmt und als Wert der *Function* die betreffende Geometrie zurückbringt.

Die zurückgebrachte Geometrie sollte dann wie eine normale Geometrieeigenschaft in räumlichen Operatoren eingesetzt werden.

Leider ist das bei der gegenwärtigen Definition des Filter Encoding nicht möglich, da bei den räumlichen Operatoren statt eines Audrucks nur ein Eigenschaftszugriff (PropertyName) erlaubt ist. Es wäre deshalb erforderlich, das Filter Encoding Schema zu erweitern und – wegen dessen enger Verknüpfung mit dem WFS Schema – auch dieses.

Vom Ansatz sähe das etwa so aus:

```
<okwfs:Query typeName='Verkehrszeichentraeger'>
 <okfe:Filter>
  <okfe:BBOX>
   <ogc:Function name='PunktAusStrassenpunkt'>
    <ogc:PropertyName>bei_Strassenpunkt</ogc:PropertyName>
    <ogc:Literal>2007-12-01</ogc:Literal>
   </ogc:Function>
   <gml:Envelope>
    <gml:lowerCorner> … </gml:lowerCorner>
    <gml:upperCorner> … </gml:upperCorner>
   </gml:Envelope>
  </okfe:BBOX>
 </okfe:Filter>
</okwfs:Query>
```

Dabei wurde das Namespace-Kürzel „okwfs" für das modifizierte WFS-Schema gewählt und „okfe" für das modifizierte Filter Encoding Schema.

Wegen der erforderlichen Erweiterungen der OGC-Spezifikationen (in daraus abgeleiteten Schemas) sollte man diese Alternative nur verfolgen, wenn solche schwerwiegenden Eingriffe aufgrund anderer Anforderungen nötig werden.

7.2.2 Alternative 2: Eine Xpath-Funktion zur Berechnung der Geometrie

Die Schwierigkeit aus Alternative 1 kann man vermeiden, indem man die Berechnung der Geometrie in den Xpath-Ausdruck des Eigenschaftszugriffs (PropertyName) verlagert.

Dies könnte etwa so aussehen:

```
<wfs:Query typeName='Verkehrszeichentraeger'>
 <ogc:Filter>
  <ogc:BBOX>
   <ogc:PropertyName>punktAusPunktobjekt(.,'2007-12-01')</ogc:PropertyName>
   <gml:Envelope>
    <gml:lowerCorner> … </gml:lowerCorner>
    <gml:upperCorner> … </gml:upperCorner>
   </gml:Envelope>
  </ogc:BBOX>
 </ogc:Filter>
```

```
</wfs:Query>
```

Der Vorteil der Lösung ist, dass weder das Filter Encoding Schema noch das WFS-Schema erweitert werden muss. Es ist nur eine textliche Verschärfung der Spezifikationen erforderlich. Zudem hat die Lösung den Vorteil, dass sie auch zur Geometriebereitstellung in Funktionen zur Kartenerzeugung eingesetzt werden kann, z.B. in einen Styled Layer Descriptor.

Die Lösung erfodert die geringsten Änderungen an den Basisspezifikationen und ist deshalb anzustreben. Allerdings ist die Realisierung aufwändig, so dass sie nicht verpflichtend gemacht werden sollte.

Empfehlung

Die XPath-Ausdrücke in Eigenschaftzugriffen (PropertyName) sollen Funktionen zur Berechnung der Geometrie aus OKSTRA-Stationierungsangaben enthalten.

7.2.3 Alternative 3: Das Einrichten der Geometrie durch Erweiterung des OKSTRA

Stationierte Objekte könnten durch Erweiterung der entsprechenden Stationierungskonstrukte durch eine optionale Geometrie-Eigenschaft aufgerüstet werden.

Diese Geometrie-Eigenschaft würde im OKSTRA optional eingerichtet, aber für die Verwendung des OKSTRA für OKSTRA-konforme Web-Services würde man fordern, dass diese in den stationierten Objekten enthalten sind. Sie könnte in Queries beliebig eingesetzt werden.

Grundsätzlich wäre dies als berechnete Eigenschaft aufzufassen. D.h. sie wäre bei lesenden Operationen sichtbar, würde aber bei schreibenden Operationen ignoriert.

Der Nachteil dieser Lösung ist natürlich die erforderliche OKSTRA-Erweiterung, was bedeutet, dass die Änderung für alte OKSTRA-Versionen nicht zur Verfügung steht.

Auf der positiven Seite steht, dass wirklich eine konkrete Geometrie zur Verfügung steht, die auch im OKSTRA XML Response enthalten ist und z.B. für die Kartierung benutzt werden kann.

7.3 Der XLink-WFS

Der z.Z. gültige WFS-Standard (1.1.0) beinhaltet eine optionale Fähigkeit, welche bei Abfragen die Bereitstellung von zusätzlichen Objekten vereinfacht, welche mit dem eigentlichen Abfrageresultat relational verknüpft sind. Die zusätzliche Bereitstellung kann pauschal gefordert werden, z.B. bis zur Relationsverfolgungstiefe n – oder – einzeln pro Eigenschaft (Property) angefordert werden.

Leider ist die Definition dieser optionalen Befähigung – genannt XLink-WFS – im augenblicklichen Zustand des Standards unzureichend. Es wird gefordert, dass die gefundenen Objekte immer in die referierenden Eigenschaften (Properties) „geschachtelt" eingesetzt werden. Dies kann jedoch fallweise zu Ergebnisdokumenten führen, welche nicht mit dem Schema validieren, z.B. weil die geforderte Schachtelung nicht gestattet ist.

Es ist zu erwarten, dass in zukünftigen Versionen des Standards dieser Definitionsfehler behoben sein wird und dass die „zusätzlich" ermittelten Objekte in irgendeiner Form an den Response der Operation angehängt werden. In dieser Form wäre der XLink-WFS in der Tat eine empfehlenswerte und wichtige Vereinfachung für die Handhabung eines stark relational verknüpften Schemas wie dem OKSTRA.

Verbunden mit der automatischen Bereitstellung relational verknüpfter Objekte ist die Zusatzoperation *GetGmlObject*. Sie ermöglicht es, einzelne Objekte per Identifier zu holen. Da hier die automatische Bereitstellung relational verknüpfter Objekte ebenfalls nicht sauber definiert wurde, aber in diesem Fall nicht so leicht geheilt werden kann, sollte eine Forderung nach Implementierung von GetGmlObject nicht gestellt werden[5].

Anforderung

Die Angabe von *traverseXlinkDepth* in der Operation *GetFeature* und bei einzelnen Properties soll generell möglich sein, wobei nur datenbanklokale Relationen aufgelöst werden müssen. Die zusätzlich ermittelten Objekte werden nicht eingebettet, sondern am Ende des GetFeature-Response angefügt.

Es ist nicht erforderlich, die *GetGmlObject*-Operation zu implementieren.

7.4 Umgang mit der OKSTRA®-Historisierung

Da die Historisierung der OKSTRA-Objekte optional ist, ergibt sich die grundsätzliche Fragestellung, ob die OKSTRA-konformen Web-Services und insbesondere die Datenbereitstellungs-Schnittstelle über OGC WFS, die Historierung überhaupt unterstützen soll oder nicht. Hier wurde im Vorfeld aufgrund des Anforderungsprofils eine Entscheidung zugunsten der Unterstützung der Historisierung getroffen.

Wenn Historisierung im Gesamtkonzept der OKSTRA-konformen Web-Services unterstützt werden soll, so hat man die Wahl, auf welcher Ebene dies erfolgen soll. Dieses Dokument behandelt nur die Datenbereitstellung auf der Basis des OGC WFS – es ist also zu festzulegen, in welcher Form die Historienunterstützung im WFS definiert werden kann.

Hierfür werden zwei alternative Herangehensweisen skizziert.

Die erste Alternative unterstützt die Historisierung rein auf der Basis des OKSTRA-Schemas, kein besonderer Komfort ist vorgesehen. In diesem Falle müsste eine Komfortunterstützung in einer höheren Schnittstellenschicht erfolgen.

Die zweite Alternative skizziert eine komfortable Historienunterstützung für die Datenbereitstellung, hier gibt es einen breiten Spielraum, der bis zu erheblichen Erweiterungen an der WFS-Schnittstelle führen kann (der damit kein WFS mehr wäre).

Zur Einführung dient eine kurze Zusammenfassung der Elemente der OKSTRA-Historisierung.

7.4.1 Die Historisierung im OKSTRA

Die meisten Objekte des OKSTRA unterliegen optional der Historisierung. D.h. sie (die Versionen) besitzen qua Ableitung von *historisches_Objekt* optionale, tagesscharfe Gültigkeitsgrenzen, *gueltig_von*, *gueltig_bis*.

Neben den Gültigkeitsgrenzen in den historischen Objekten des OKSTRA wird die Historisierung noch durch eine Reihe von Hilfsobjekten unterstützt. Die Zusammenfassung der verschiedenen his-

[5] In der nächsten Version des WFS wird die Operation durch eine allgemeinere Funktionalität names *GetPropertyValue* ersetzt werden.

torischen Objektversionen, die alle nur Varianten ein und desselben zeitveränderlichen Modellgegenstands (im OKSTRA „Objekt" genannt) sind, wird durch eine lineare Verkettung der historischen Objektversionen beschrieben. Zeitveränderliche Objekte als solche sind im OKSTRA nicht direkt repräsentiert.

Ereignis-Objekte können Erzeugung und Löschung von historischen Objektversionen, die zu einer Maßnahme gehören, dokumentieren. Die oft komplexen übergänge bei Veränderungen in der Stationierung können durch *identisches_Netzteil* und *Teilabschnitt_IdNT* beschrieben werden.

Die Relationen zwischen den Objekten des OKSTRA unterliegen nur implizit der Historisierung. Sie tragen selbst keine Gültigkeitsgrenzen, sondern gewinnen diese aus den historischen Objektversionen, welche sie miteinander verknüpfen. D.h. die impliziten Gültigkeitsgrenzen einer Relationsinstanz sind durch den Schnitt der Gültigkeitsintervalle der beiden beteiligten historischen Objekte bestimmt.

7.4.2 Alternative 1: Einfache Historienunterstützung auf Basis des OKSTRA

Als einfachste Lösung bietet es sich an, die Historisierung des OKSTRA unmittelbar durch Verwendung der dafür vorgesehenen Objekteigenschaften, also insbesondere der Gültigkeitsgrenzen, *gueltig_von*, *gueltig_bis* umzusetzen.

Bei lesenden Aufrufen, muss unter Verwendung dieser Grenzen durch entsprechende Vergleiche dafür gesorgt werden, dass nur die Objekte zum gewünschten Stichtag erfragt werden.

Dies ist leicht möglich, beschreibt das Problem aber zur num Teil. Denn es ist dabei zu bedenken, dass Relationen zu Objektversionen bestehen können, die nicht für den Stichtag gültig sind. Will man über Xpath-Ausdrücke auf die Inhalte solcher relational verknüpfter Objekte zugreifen (siehe 7.1.2), so ist sicherzustellen, dass hier keine stichtags-ungültigen Objektversionen erfasst werden.

Das geht nur mit den in 7.1.4 geforderten Prädikaten.

Das Beispiel aus 7.1.2 muss bei einer Abfrage zum Stichtag folgendermaßen abgefragt werden:

```
<wfs:Query typeName='Abschnitt'>
 <ogc:Filter>
  <ogc:And>
   <ogc:PropertyIsBetween>
    <ogc:Literal>2005-03-25</ogc:Literal>
    <ogc:LowerBoundary>
     <ogc:PropertyName>gueltig_von</ogc:PropertyName>
    </ogc:LowerBoundary>
    <ogc:UpperBoundary>
     <ogc:PropertyName>gueltig_bis</ogc:PropertyName>
    </ogc:UpperBoundary>
    </ogc:PropertyIsBetween>
    <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>gehoert_zu_Strasse/
     Strasse['2005-03-25' >= gueltig_von and gueltig_bis >= '2005-03-25']/hat_Strassenbezeichnung/
     Strassenbezeichnung/Strassenklasse/Strassenklasse/Kennung</ogc:PropertyName>
    <ogc:Literal>A</ogc:Literal>
   </ogc:PropertyIsEqualTo>
  </ogc:And>
 </ogc:Filter>
</wfs:Query>
```

Diesen Ansatz kann man verbessern, indem man das Attribut <wfs:Query featureVersion="xxx"> einsetzt, um Stichtage vorzugeben. Dieses Attribut ist zwar in der textlichen Beschreibung des WFS als Versionsnummer[6] festlegt, syntaktisch ist es aber ein String.

[6] In der nächsten WFS-Version wird diese Einschränkung allerdings nicht mehr bestehen.

Es würde dann durch einen OKSTRA-WFS dieser Stichtag berücksichtigt und automatisch zur Filterung aller historischen Objekte und der Relationen dazwischen verwendet. Bei Weglassen würde nur der aktuelle Stand berücksichtigt und bei einem Sonderwert „ALL" würden, wie auch im WFS-Standard vorgesehen, alle Varianten berücksichtigt und die oben beschriebene Filterlogik würde greifen. Es ist allerdings klar, dass das mit einem WFS „aus dem Regal" nicht zu machen ist.

Bei schreibenden Aufrufen steht ein solches einfaches Vehikel zur Stichtagssteuerung nicht zur Verfügung. Hier muss der der Client sicherstellen, dass die vom OKSTRA geforderten Invarianten eingehalten werden. D.h. die Gültigkeitsbereiche der Versionsvorgänger sind durch Update einzuschränken und die neue Objektversion ist entsprechend durch Insert-Operation einzufügen. Ereignis-Objekte und identische Netzteile sind zu pflegen.

Zumindest bei den schreibenden Operationen erscheint es lohnend, einen entsprechenden Fach-Service, der dies bewerkstelligen kann, zu definieren.

Obwohl die Handhabung von Historisierung auf der Basis des OKSTRA zumindest für das Schreiben mühsam erscheint, so ist so doch möglich und relativ klar definiert. Es handelt sich auch um eine Möglichkeit, die immer zur Verfügung stehen stehen wird (und muss), obwohl in der WFS-Schicht oder darüber bequemere Möglichkeiten eingerichtet werden. In diesem Sinne kann zur Zeit eigentlich nur diese Variante ausgewählt werden.

Anforderung

Die Unterstützung der OKSTRA-Historie erfolgt ohne zusätzliche Erweiterungen der WFS-Schnittstelle. Erforderlich sind die an anderer Stelle geforderten „relations-übergreifenden" Xpath-Ausdrücke und Prädikate an den Objekten. Das Attribut featureVersion bei GetFeature/Query wird zur Stichtagsauswahl eingesetzt.

Die Unterstützung weitergehender Komfort-Operationen erfolgt über der Datenbereitstellungsschicht.

7.4.3 Alternative 2: Historienunterstützung mit Komfort

Eine komfortable Form der Historienunterstützung würde

- es bei lesenden Zugriffen erlauben, den Datenbestand im WFS auf einfache Weise auf einen bestimmten Zeitpunkt oder ein Zeitintervall einzuschränken, und
- bei schreibenden Zugriffen für ein gegebenes Änderungsdatum oder Änderungsintervall, die automatische Aufteilung von historischen Objekten vornehmen. Hierbei sollten bestehende Ereignis-Objekte automatisch fortgeschrieben werden.

Tatsächlich ist eine solche komfortable Historienunterstützung sehr schwer durch ein Profil des OGC WFS zu erreichen.

Das Lesen kann wie in der Alternative 1 erreicht werden. Dies ließe sich evtl. auch für Zeiträume erweitern.

Beim Schreiben liegen die Anforderungen an die WFS-Schnittstelle noch viel höher.

Neue Objekte ohne Bezug auf ältere Objekte werden durch Insert erzeugt.

Durch eine Update-Operation mit einem übergebenen Bezugsdatum (Erweiterung der Schnittstelle) wird automatisch die Filterung der Objekte unterstützt und die Aufteilung der betroffenen Objekte

bewirkt, falls das erforderlich ist. Die neuen Objekte müssten durch ein *UpdateResult* analog zum bestehenden *InsertResult* nachgewiesen werden.

Durch eine Delete-Operation mit einem übergebenen Bezugsdatum (Erweiterung der Schnittstelle) würde auch hier automatisch die Filterung der Objekte unterstützt und die qualifizierenden Objekte werden beendet.

Es ist zu bedenken, dass neben der beschriebenen Komfortschnittstelle auch noch die normale in 7.4.2 beschriebene Schnittstelle existieren muss, um Korrekturen an Objekten ohne Historienbildung zu gestatten.

7.5 Transaktionen und Locks

Dieser Abschnitt diskutiert die Fragestellungen zur Profilbildung des WFS bezüglich der gesicherten Durchführung schreibender Operationen. In einem Nutzungsumfeld, das die Datenbereitstellung konkurrierend lesend und schreibend durch mehrerer Clienten benutzt, muss sichergestellt werden, dass von keiner Partei inkonsistente Zwischenzustände gesehen werden können oder sich durch Fehlersituationen permanent manifestieren können. Gewöhnlich wird dies durch die Stichworte Serialisierung und *atomicity, consistency, isolation, durability* – kurz ACID adressiert.

In der Datenbanktechnologie wird dies überlicherweise durch die Verwendung von Transaktionen und Locks (Sperren) gewährleistet. Der WFS-transactional unterstützt diese Konzepte – zu einem gewissen Umfang. Es ist in diesem Kapitel zu analysieren, wie weit man mit dem Standardverhalten kommt und was man verschärfend in der Schnittstelle bereitstellen kann.

7.5.1 Der Leistungsumfang der WFS-Schnittstelle

Die WFS-Spezifikation besitzt nur relativ schwach ausgeprägte Hilfsmittel zur Serialisierung konkurrierender Fortschreibungsoperationen, nämlich:

- Die atomare (also ganz-oder-gar-nicht) Ausführung zusammengesetzter Insert/Update/Delete-Operationen. (Deshalb heißt diese WFS-Operation auch „Transaction".)
- Die explizite Isolation der Zugriffe auf einzelne Features durch „Locks"

Mit diesen Hilfsmitteln ist die gesicherte Fortführung von Datenbeständen im Große und Ganzen möglich.

Programmiertechnisch ist die Nutzung der vorhandenen Serialisierungshilfsmittel allerdings recht aufwändig: Durch das Fehlen operationsübergreifender Transaktionen muss die Programmierung eines transaktionssicher ändernden Dienstes so erfolgen, dass zuerst alle beteiligten Features unter Vergabe eines (!) Locks (also durch ein *GetFeatureWithLock* oder ein *LockFeature*) geholt und gesperrt werden müssen. Sobald dies erfolgreich geschehen ist, können die vorgenommenen Veränderungen im Prinzip durch eine oder mehrere *Transaction*-Operationen unter Bezugnahme auf dieses Lock zurückgeschrieben werden.

Mehrere *Transaction*-Operationen dürfen es aber nur sein, wenn bereits nach der ersten solchen ein konsistenter Zustand erreicht ist. Ansonsten muss der gesamte Schreibvorgang in einer *Transaction* zusammengefasst werden.

Die durch den WFS-Standard definierten Locks werden teilweise kritisch bewertet, da sie möglicherweise durch zu restriktives Verhalten, die Nutzung der Datenhaltung zu stark einschränken. Es soll daher auch möglich sein, eine „optimistische" Locking-Strategie zu implementieren. Bei dieser

wird beim Lesen zum Zwecke des späteren Schreibes die Zeit an den Objekten vermerkt und dem Client mitgeteilt. Beim Schreiben wird geprüft, ob dieser Zeitvermerk unverändert geblieben ist – ansonsten wird der Schreibvorgang zurückgewiesen. Siehe hierzu auch den folgenden Abschnitt 7.5.2.

Eine Entscheidung fällt angesichts der Beschränkungen des vorhandenen Konzepts nicht leicht. Da allerdings alle weitergehenden Definitionen schwerwiegende Erweiterungen des WFS-Standards darstellen, wird empfohlen, erstmal bei der Standardfunktionalität (erweitert um „optimistische" Locks) zu bleiben.

Anforderung

Die Datenbereitstellungsschnittstelle über WFS soll in ihrer schreibenden Ausbaustufe *Transactions* und *Locks* im vollen Umfang unterstützen. Es soll dabei auch möglich sein, eine „optimistische" Locking-Strategie umzusetzen.

Ein weitergehendes Profil – wenn es denn gebraucht wird – sollte auf alle Fälle als dritte Ausbaustufe definiert werden, etwa „OKSTRA – Datenbereitstellung, verteilt-transaktional".

7.5.2 „Optimistisches" Locking und der WFS

Die WFS-Spezifikation verlangt, dass zu einer Zeit höchstens ein Lock für ein Feature existieren darf und dass Veränderungen an Features, für die ein Lock gesetzt wurde, nur durch die Transactions möglich sind, welche die LockId des Locks kennen. Für einen Clientprozess ergibt sich, dass Features, welche er durch die entsprechenden Operationen (LockFeature bzw. GetFeatureWithLock) mit Locks belegt hat, nur von ihm (mittels der Operation Transaction) verändert werden können, denn kein zweiter Clientprozess kann dieselben Features mit Locks belegen oder verändern. Jeder Versuch, ein zweites Lock auf einem Feature zu nehmen oder ein „gelocktes" Feature illegal zu verändern, würde nach der WFS-Spezifikation abgewiesen.

Es handelt sich bei den Locks der WFS-Spezifikation also um wirkliche Sperren. Der WFS oder die Datenhaltung müssen sich merken, welche Features in diesem Sinne gesperrt sind und diese Sperren während der definierten Lebenszeit der Locks aufrecht erhalten. Diese aktive Form des Lockings, die davon ausgeht, dass mehrere Clientprozesse parallel dieselben Features verändern wollen und ebendies verhindern, wird oft als „pessimistisches" Locking bezeichnet, besonders im Gegensatz zum im Folgenden beschriebenen „optimistischen" Ansatz.

Es gibt die Möglichkeit, die Steuerung der konkurrierenden Zugriffe auf ein und dasselbe Feature mit einer „optimistischen" Strategie umzusetzen. Dieser Fall geht davon aus, dass es relativ selten passiert, dass zwei verschiedene Clientprozesse zeitgleich dieselben Daten editieren wollen – deshalb die Bezeichnung.

Es werden bei dieser Strategie keine harten Sperren gesetzt, vielmehr wird beim Locking ein Zeitstempel der zu sperrenden Features zurückgeliefert. Dieser Zeitstempel gibt an, wann zuletzt eines der betroffenen Feature verändert wurde. Die Lock-Operationen können in diesem Fall mehrmals durch unterschiedliche Clients und für dieselben Features aufgerufen werden. Die Umsetzung setzt voraus, dass der WFS oder die Datenhaltung intern für jedes Feature den Zeitpunkt der letzten Änderung pflegt. Die Konfliktbehandlung erfolgt erst innerhalb des Transaction Requests. Hier muss getestet werden, ob ein zu aktualisierendes Objekt nach Aufruf der Lock-Operation bereits durch einen zweiten Clientprozess verändert wurde (ob der Zeitstempel des Features nach dem

Zeitstempel der in der LockId übergeben wird liegt). In diesem Fall schlägt die Transaktion fehl. Nur wenn zwischenzeitlich kein zweiter Prozess dieselben Daten verändert hat, kann die Transaktion ausgeführt werden.

Die Abbildung dieser „optimistischen" Strategie auf den WFS kann formal mit denselben Operationen arbeiten, wie im WFS-Standards vorgesehen. Die Operationen LockFeature bzw. GetFeature-WithLock bringen eine LockId zurück, die im „optimistischen" Fall ein Zeitstempel ist. Diese LockId wird dann bei der Transaction-Operation an den WFS übergeben.

Der Unterschied liegt darin, ob die Konfliktbehandlung bereits in der Lock-Operation erfolgt oder erst dann, wenn die Daten verändert werden. Ein Client, der einen „optimistischen" Server versorgt, kann nicht fest damit rechnen, dass ein Feature, das er gelockt hat, ihm zur Verfügung steht. Es kann passieren, dass die Veränderung abgewiesen wird, obwohl er ein Lock besitzt. Der mögliche Konflikt wird jedoch erkannt.

Clients müssen deshalb auf diese späte Konfliktbehandlung eingehen können. Dies ist allerdings keine wirklich zusätzliche Problematik, weil auch bei einer „pessimistischen" Strategie ein Veränderungsversuch durch Transaction fehlschlagen kann, z.B. wegen Zeitüberschreitung des Locks oder wegen eines Systemfehlers. Ein korrekter Client muss also letztlich auch in diesem Fall darauf vorbereitet sein, den gesamten Zyklus von Datenbeschaffung-Locking-Veränderung-Transaction zu wiederholen – auch wenn die Fälle beim „pessimistischen" Verfahren naturgemäß unwahrscheinlicher sind.

Wegen der geringen Unterschiede und der davon erwarteten Vorteile im Verhalten, bietet es sich an, für den OKSTRA-WFS auch die „optimistische" Strategie zuzulassen. Es ist aber festzuhalten, dass diese Art der Implementierung nur formal unter Verwendung der WFS-Operationen arbeitet. Sie entspricht nicht im vollen Umfang dem Wortsinn der WFS-Spezifikation.

7.5.3 Operationsübergreifende Transaktionen

Die Definition operationsübergreifender Transaktionen wäre für die effektive Umsetzung von fachlichen Diensten „über" dem WFS sehr wünschenswert. Sie würde die Programmierung stark vereinfachen, da das Verwendungsmuster aus jeweils einem Paar *Lock + Transaction* viel flexibler gestaltet werden könnte.

Technisch würde man das durch zwei zusätzliche Operationen machen, nämlich

1. *TransactionBegin*, welche einen *TransactionsIdentifier* zurückbringt, und
2. *TransactionEnd* mit den Optionen *Commit* und *Abort* für einen *TransactionIdentifer*.

Der *TransactionIdentifier* wäre zusätzlich optionaler Input in die erweiterte *Transaction*-Operation.

Alternativ könnte man die beiden Operationen *TransactionBegin* und *TransactionEnd* auch als Optionen in die erweiterte *Transaction*-Operation aufnehmen.

Eins ist an dieser Stelle noch zu diskutieren:

Der WFS-Standard bietet auch die Operation *Native* an, die dazu verwendet werden kann, Basis-Funktionalität der zugrundeliegenden Datenbank anzusprechen. Das kann im Prinzip eingesetzt werden. Wir geben in diesem Falle aber die einheitliche Schnittstelle auf, da die zugrundeliegende Datenbank sicher durchaus verschieden ausfallen wird. Von diesem Weg ist deshalb eher abzuraten.

7.5.4 Verteilte Transaktionen

Bei serverübergreifenden Fortführungen weiten sich die beschriebenen Sachverhalte zu ernsthaften und nicht-trivialen Problemen aus, die durch Nutzung der WFS-Schnittstelle alleine nicht in den Griff zu kriegen sind.

Bei Durchführung einer komplexen Operation auf mehreren Servern kann nämlich die sichere Logik des Holens und Sperrens und anschließenden Veränderns der gesperrten Features durch eine *Transaction*-Operation nicht mehr funktionieren, weil für jeden Server eine extra Fortschreibungs-Operation durchgeführt werden muss.

Das Problem zeigt sich besonders auf zwei Arten:

1. Geht die erste *Transaction*-Operation auf dem ersten Server glatt und versagt die zweite, so bleibt das Gesamtsystem in einem inkonsistenten Zustand, der nicht zurückgefahren werden kann. Dies ist ein schlimmer Fall, weil die Datenbasis dauerhaft in einem insgesamt inkonsistenten und deshalb evtl. unbrauchbaren Zustand verbleibt.
2. Verbunden damit ist auch eine „unsaubere Sichtbarkeit" der nach der erfolgreichen *Transaction* auf dem ersten Server fortgeführten Features. Denn die dazu gehörenden Fortführungen auf Server 2 sind noch nicht ausgeführt, auch wenn dies schließlich erfolgreich geschehen sollte. Dies erzeugt bis zur Fertigstellung der Operation auf Server 2 temporär denselben inkonsistenten Zustand wie im ersten Fall.

Durch explizit kontrollierbare, also operationsüberspannende Transaktionen würden sich diese Effekte vermeiden lassen. Man würde in diesem Falle die Transaktion auf Server 1 offen lassen und „zurückrollen", falls die Transaktion auf Server 2 fehlschlägt. In diesem Falle würde der Client einen Teil der Rolle des Koordinators in einem verteilten Transaktionsprotokoll übernehmen.

Die für die Durchführung von verteilten Transaktionen erforderlichen Steuerungs- und Kommunikationsaufgaben sind leider i.d.R. kein standardisierter Bestandteil von Datenbankmanagementsystemen, sodass sich das Konzept des WFS einfach dieser Fähigkeiten bedienen könnte.

Soll eine solche Lösung tatsächlich umgesetzt werden, so muss die WFS-Schnittstelle um die Operationen eines des 2-Phasen-Commit-Protokolls (oder einer dessen Varianten) ergänzt werden. Dies dem Client zu überlassen, ist wahrscheinlich nicht anzuraten, da das System auch gegen den Ausfall des Clients geschützt werden muss.

7.6 Normalisierung des OKSTRA

Für Darstellung einiger Sachverhalte bietet der OKSTRA die Möglichkeit alternativer Repräsentierungen. Dies ist für seine Rolle als Austauschformat zulässig, aber sehr hinderlich, wenn es darum geht, Abfragen gegen einen Datenbestand, der einem über das OKSTRA XML Schema entgegentritt: Es wäre bei Abfragen erforderlich, die Abfrage für alle Alternativen zu formulieren, weil man nicht weiß, welcher Alternative der Datenbestand folgt. Ähnliches gilt für transaktionale (fortschreibende) Operationen, die sich naturgemäß an der vorhandenen Repräsentierung ausrichten müssen.

Es ist also nötig sicherzustellen, dass ein Sachverhalt nur auf eine Art dargestellt werden kann.

In folgenden Bereichen könnte/sollte eine Normalisierung vorgenommen werden:

1. **Schlüsseltabellen:** Die OKSTRA-Schlüsseltabellen können nach dem OKSTRA XML Schema wahlweise über xlink-Verweise referenziert oder jeweils an Ort und Stelle eingebettet

werden. Dies kann man imgunde offen lassen, da der Einsatz in Filterausdrücken in beiden Varianten dieselbe Syntax aufweist.

2. **Abstrakte Verweise:** Ein abstrakter Verweis kann nach dem OKSTRA XML Schema entweder einen xlink-Verweis auf die referenzierte Instanz oder einen konzeptionellen Schlüssel (d.h. einen String) enthalten, der die referenzierte Instanz in fachlicher Hinsicht eindeutig identifiziert. Hier sollte ein Ansatz gefunden werden, der der Verweistechnik in XML-Dokumenten besser gerecht wird.

3. **Bereichsobjekte:** Die *Bereichsobjekte* des OKSTRA (insbesondere die *Strassenbaudienststellen* und *Verwaltungsbezirke*) werden im Straßennetz auf *Netzbereichen* verortet. Diese *Netzbereiche* können im OKSTRA aus *Teilabschnitten* sowie deren Aggregationen (*Strecken* oder anderen *Netzbereichen*) zusammengesetzt werden. Es hat sich herausgestellt, dass sich in diesem Punkt kein normalisiertes OKSTRA-Profil finden lässt, das allen Implementierungen[7] gerecht wird. Es muss daher durch den OKSTRA-WFS das OKSTRA-konform interpretierte, tatsächlich in den Straßendatenbanken vorhandene Modell angeboten werden. Das verwendete Modell ist in den Capabilities anzuzeigen.

4. **Streckenobjekte:** Die *Streckenobjekte* des OKSTRA können sowohl auf einem *Teilabschnitt* als auch auf einer Aggregation von *Teilabschnitten* (einer *Strecke*) verortet werden. Als dritte Möglichkeit kommt noch die Verortung auf einem *Strassenelement* in Betracht. Im Sinne einer Normalisierung sollte stets eine *Strecke* angegeben werden (die aus 1 … n *Teilabschnitten* bestehen kann). Die Verortung auf dem *Strassenelement* sollte nicht berücksichtigt werden, da bislang keine diesbezüglichen Daten existieren.

5. **Teilabschnitte / Strecken / Netzbereiche:** Die Instanzen dieser Objektarten können im OKSTRA zur Verortung einer beliebigen Menge von Fachobjekt-Instanzen dienen. Im Sinne einer Normalisierung wäre es jedoch sinnvoll, immer nur eine einzige Instanz eines Fachobjektes auf diesen Objektarten zu verorten, weil es sonst (bei im Grunde gleichem Dateninhalt) zu unterschiedlich strukturierten Netzverortungen kommen kann. Beispiel: Der Sachverhalt, dass zwei Fachobjekte im gleichen Stationsbereich auf demselben *Abschnitt* liegen, kann nach dem OKSTRA entweder dadurch angegeben werden, dass die beiden Fachobjekte auf denselben *Teilabschnitt* verweisen oder dadurch, dass jedes Fachobjekt auf einen eigenen *Teilabschnitt* verweist, die aber beide identische Daten enthalten. Theoretisch könnte einer der beiden *Teilabschnitte* zusätzlich auch noch an einem *Netzbereich* beteiligt sein.

6. **Identifier**: Es ist ein einheitliches Konzept für die Bezeichnung von Features (Objektversionen) zu entwicklen. Es sollen serverübergreifende Referenzen möglich sein (siehe Punkt 2).

7. **GML-Geometrierepräsentation:** Das GML-Profil des OKSTRA® ermöglicht tlw. verschiedene Varianten zur Angabe von Geometrie. Hier ist ein möglichst einfaches Profil vorzusehen.

8. **Koordinatenreferenzsysteme:** Der OKSTRA® ermöglicht die Angabe von Koordinaten in verschiedenen Koordinatenreferenzsystemen (vgl. Dokument T0006, Abschnitt 4.8.2). Es ist zu festzulegen, welche Koordinatenreferenzsysteme durch die WFS-Datenbereitstellung unterstützt werden sollen. Hierbei ist insbesondere eine Abwägung vorzunehmen zwischen

[7] Der grundsätzliche Unterschied besteht darin, dass bei einer der Implementierung grundsätzlich alle Bereichsobjekte aus *Teilabschnitten* bestehen, während bei einer anderen bei den Bereichsobjekten *Strassenbaudienststelle* und *Verwaltungsbezirk* die *Netzbereiche* der untersten Ebenen aus *Teilabschnitten* und die Netzbereiche der höheren Ebenen aus den *Netzbereichen* der jeweils darunterliegenden Ebene zusammengesetzt werden.

den bei OGC-Diensten gebräuchlichen EPSG-Bezeichnungen und den Bezeichnern der AdV[8], welche in T0006, 4.8.2 vorgeschlagen wurden.

9. **Fachobjektarten:** Es ist nicht auszuschließen, dass im OKSTRA® – ähnlich wie bei den Netzverortungen (siehe Punkte 3 – 5) auch bei einzelnen Fachobjektarten alternative Repräsentierungen existieren. Auch hier sollten erforderlichenfalls Normalisierungen vorgenommen werden.

Anforderung
In allen genannten Bereichen sind einschränkende Festlegungen zu treffen.

7.7 Protokolle der Datenbereitstellungsschnittstelle

OGC-Dienste wie auch die WFS-Spezifikation benutzen im Allgemeinen http/POST und http/GET. Nur der POST-Schnittstelle ist es möglich, den gesamten Funktionsumfang abzubilden – sie muss daher angeboten werden.

GET ist oft sehr praktisch bei der Erstellung von Client-Software, ist aber nicht unbedingt erforderlich. Es ist aber zu erwarten, dass es zumindest für lesende Zugriffe zunehmend wichtiger werden wird.

Falls Authentifizierung und Autorisierung durch zusätzliche Komponenten unterstützt werden soll, ist zu empfehlen, die in der Spezifikation vorgeschlagene SOAP-Schnittstelle verpflichtend zu machen, da sie es gestattet in den Header-Feldern die erfordlichen Informationen zu übertragen, ohne die eigentlichen Verwendungsparameter zu stören.

Anforderung
Die WFS http/GET, http/POST- und SOAP-Protokolle sind verpflichtend.

7.8 Zusätzliche Anforderungen

Es hat sich erwiesen, dass die Implementierung von Geodateninfrastrukturen oft an der fehlenden Performanz der beteiligten Dienste scheitern.

Unter 500 Features pro Sekunde bei Queries und 100 Features pro Sekunde bei Insert/ Update/ Delete, sind solche Dienste nicht sinnvoll einsetzbar.

Gleichfalls hat sich herausgestellt, dass Dienste oft starke Einschränkungen bezüglich der abgebbaren Datenmengen und der Größe der Eingabedokumente bei schreibenden Operationen aufweisen, welche ihre Nutzung sehr erschweren oder unmöglich machen.

Authentifizierung und Autorisierung sind nicht Bestandteil des WFS-Konzepts. Die Standardisierung dieser (sehr berechtigten) Anforderungen wird zur Zeit aktiv betrieben, aber natürlich nicht im Rahmen des WFS-Standards selbst.

Es ist zu entscheiden, ob solche zusätzlichen Anforderungen mit in das Profil aufgenommen werden sollen.

[8] Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland

Empfehlung
Die Problematik wird beschrieben, aber nicht in den verpflichtenden, normativen Teil des Profils aufgenommen.

8 OKSTRA®-WFS lesend (normativ)

Die *Datenbereitstellungs-Schnittstelle für OKSTRA®-Daten auf der Basis des OGC Web Feature Service*, kurz *OKSTRA®-WFS*, wird in zwei aufeinander aufbauenden Spezifikationen als „Profil" des OGC WFS definiert.

In diesem Kapitel erfolgt die Definition des *OKSTRA®-WFS lesend,* welcher nur Funktionalität für den lesenden Zugriff auf die Daten bereitstellt.

Ein ***OKSTRA®-WFS lesend*** ist ein **OGC WFS basic Version 1.1.0**, der die gesamte obligatorische Funktionalität des OGC WFS basic aufweist, wie sie in der normativen Referenz 4.2, ergänzt durch Referenz 4.1, beschrieben ist, **und** der zusätzlich **alle** Anforderungen, aus dem vorliegenden Kapitel erfüllt.

Unter der obligatorische Funktionalität des OGC WFS basic wird hier die Funktionalität verstanden,

- welche im Standard durch entsprechende Verbformen wie „shall" oder „must", etc. ausdrücklich als solche bezeichnet wird **oder**
- für die keine Möglichkeit vorliegt, deren Fehlen im Capabilities-Response-Dokument zu beschreiben.

Ausgenommen von dieser Regel sind alle Angaben, die im vorliegenden Dokument ausdrücklich anders geregelt sind.

Verbindliche Regelungen werden durch die Verbformen „muss" oder „soll" ausgedrückt. Konjunktivische Formen wie „sollte" dagegen drücken Empfehlungen aus.

Folgende Konventionen werden im Folgenden einheitlich verwendet:

Verwendung von Namespaces:

Abkürzung	Namespace	Bedeutung
xlink	http://www.w3.org/1999/xlink	Xlink-Schema
ows	http://www.opengis.net/ows	OWS Common Schema
ogc	http://www.opengis.net/ogc	Filter Encoding
wfs	http://www.opengis.net/wfs	Web Feature Service
okwfs	http://www.okstra.de/namespaces/okwfs	OKSTRA-WFS
okstra	http://schema.okstra.de/1012/okstra	OKSTRA-Schema Version 1.012

8.1 Profil des Web Feature Service

Dieser Teil der OKSTRA-WFS-Spezifikation für den *OKSTRA-WFS lesend* behandelt die zusätzlichen Definitionen, welche die OGC-WFS-Spezifikation betreffen.

Als erstes werden die drei bereitzustellenden Operationen des WFS (GetCapabilities, DescribeFeatureType und GetFeature) im Einzelnen festgelegt. Darauf folgend werden Aussagen zu Koordinatensystemen, Identifiern, Ausnahmebehandlungen und Protokollen gemacht.

8.1.1 Operation: GetCapabilities

Die Operation GetCapabilities muss ein WFS_Capabilities-Dokument zurückbringen, das die Fähigkeiten des OKSTRA-WFS korrekt beschreibt.

Es sind alle Abschnitte außer *wfs:ServesGMLObjectTypeList* anzugeben. Im Einzelnen:

8.1.1.1 ows:ServiceIdentification

ows:ServiceType und ows:ServiceTypeVersion sind entsprechend als WFS und 1.1.0 auszuweisen.

Es wird empfohlen, bei ows:Title, ows:Abstract und ows:Keywords Angaben zu machen, welche den jeweiligen OKSTRA-WFS in Metadata Verzeichnissen gut unterscheidbar beschreiben.

8.1.1.2 ows:ServiceProvider

Keine Anforderungen oder Empfehlungen.

8.1.1.3 ows:OperationsMetadata

Es sind wenigstens die Operationen GetCapabilities, DescribeFeatureType und GetFeature zu beschreiben.

Alle Operationen müssen unter ows:DCP/ows:http ein URL für ows:Get und ows:Post vorsehen.

Bei DescribeFeatureType und GetFeature ist das *outputFormat* „text/xml; subtype=gml/3.1.1" anzugeben. Wenn durch GetFeature neben dem von der WFS-Spezifikation geforderten wfs:FeatureCollection-Dokument zusätzlich auch ein okstra:OKSTRAObjektmenge-Dokument entsprechend dem OKSTRA-XML-Standard erzeugt werden kann, so ist dies durch das *outputFormat* „text/xml; subtype=okstra/1.012" anzuzeigen.

Bei GetFeature soll der Parameter *resultType* die Werte „results" und „hits" aufweisen.

Der globale Parameter mit Namen „srsName" ist anzugeben. Er soll alle in 8.1.4 geforderten und vom OKSTRA-WFS unterstützten Systeme mit ihren EPSG-Codes[9] aufzählen. Es ist dabei die in 8.1.4 beschriebene urn-Notation zu verwenden.

Der globale Constraint mit Namen „SupportsSOAP" ist anzugeben und soll den Wert „true" aufweisen.

Der globale Constraint mit Namen „DefaultMaxFeatures" soll nicht angegeben werden.

Der globale Constraint mit Namen „LocalTraverseXlinkScope" ist anzugeben und soll die Werte „0" und „*" aufweisen.

Falls der OKSTRA-WFS die Auflösung von „Remote Xlinks" nicht unterstützt (das ist nicht gefordert), ist der globale Constraint mit Namen „RemoteTraverseXlinkScope" mit den Werten „0" und „0" anzugeben. Falls „Remote Xlinks" unterstützt werden, ist „RemoteTraverseXlinkScope" entsprechend den tatsächlich vorhandenen Fähigkeiten anzugeben.

Zusätzlich kann es erforderlich sein, dass als letztes Element von ows:OperationsMetadata das Element ows:ExtendedCapabilities anzugeben ist. Es ist erforderlich, um folgende OKSTRA-WFS-spezifische Sachverhalte auszudrücken:

- Unterstützt die Implementierung *ImplicitGeometryFunctions*, wie sie in 8.2.1.4 beschrieben werden und gegebenenfalls welche?

[9] Für den Fall, dass okstra:OKSTRAObjektmenge-Dokumente erzeugt werden können, auch die in T0006 verlangten Koordinatenreferenzsystem im adv-Namespace.

- Ist im Falle eines *OKSTRA-WFS transaktional*, die *optimistische* Lockstrategie implementiert (Wert *optimistic*) oder die im OGC WFS-Standard vorgesehene *pessimische* (Wert *pessimistic*)? Die pessimistische Variante ist der Defaultwert.
- Werden im Falle eines *OKSTRA-WFS transaktional* nicht nur die geforderten lokalen Rückwärtsverweise unterstützt (Wert *backwards-only*) sondern auch Vorwärtsverweise (Wert *both-directions*)? „Nur Rückwärtsverweise" bilden den Defaultwert.
- Die Netzbereiche welcher Bereichsobjekte sind aus Teilabschnitten bzw. aus weiteren Netzbereichen zusammengesetzt? Es müssen alle (nicht abstrakten) Bereichsobjekte angegeben werden, welche aus Netzbereichen zusammengesetzt sind, d.h. Teilabschnitt ist der Defaultwert.

Die zu verwendende Syntax ist wie folgt:

```
<ows:OperationsMetadata>
…
    <ows:ExtendedCapabilities>
        <okwfs:ImplicitGeometryFunction>okwfs:punktAusPunktobjekt</okwfs:ImplicitGeometryFunction>
        <okwfs:ImplicitGeometryFunction> okwfs:linieAusStreckenobjekt</okwfs:ImplicitGeometryFunction>
        …
        <okwfs:LockingStrategy>optimistic</okwfs:LockingStrategy>
        <okwfs:LocalReferenceAllowedDirection>both-directions</okwfs:LocalReferenceAllowedDirection>
        <okwfs:NetzbereichsAufbau
            bereichsobjekt='okstra:Gemeindebezirk'>okstra:Netzbereich</okwfs:NetzbereichsAufbau>
        <okwfs:NetzbereichsAufbau
            bereichsobjekt='okstra:Ortsteil'>okstra:Teilabschnitt</okwfs:NetzbereichsAufbau>
        …
    </ows:ExtendedCapabilities>
</ows:OperationsMetadata>
```

ows:ExtendedCapabilities kann weggelassen werden, wenn in allen Fällen die Defaultwerte zutreffen.

8.1.1.4 wfs:FeatureTypeList

Es sind alle vom OKSTRA-WFS bereitgestellten FeatureTypes zu beschreiben. Die anzugebenden FeatureTypes stellen eine Teilmenge der in der OKSTRA-XML-Spezifikation festgelegten OKSTRA-FeatureTypes dar.

Der Name des FeatureTypes ist bei wfs:Name anzugeben, z.B.:

```
<wfs:Name>okstra:Abschnitt</wfs:Name>
```

Es wird empfohlen, bei wfs:Title, wfs:Abstract und ows:Keywords Angaben zu machen, welche den jeweiligen FeatureType in Metadata Verzeichnissen gut unterscheidbar beschreiben.

Die Elemente wfs:DefaultSRS und wfs:OtherSRS sind anzugeben. Sie sollen alle in 8.1.4 geforderten und vom OKSTRA-WFS unterstützten Systeme mit ihren EPSG-Codes[10] aufzählen. Es ist dabei die in 8.1.4 beschriebene urn-Notation zu verwenden.

Wenn das wfs:Operations Element angegeben wird, so muss wenigstens

```
<wfs:Operation>Query</ wfs: Operation>
```

enthalten sein.

wfs:OutputFormats muss „text/xml; subtype=gml/3.1.1" enthalten.

[10] Für den Fall, dass okstra:OKSTRAObjektmenge-Dokumente erzeugt werden können, auch die in T0006 verlangten Koordinatenreferenzsystem im adv-Namespace.

Das Element wfs:WGS84BoundingBox ist anzugeben, wenn es sich um einen geometrietragenden FeatureType handelt oder um einen solchen mit impliziter Geometrie. Es soll das einschließende Rechteck der Geometrien in Grad angegeben werden. Reihenfolge der Koordinaten: Geographische Länge – Geographische Breite.

8.1.1.5 wfs:SupportsGMLObjectTypeList

Es ist auszudrücken, welche GML-ObjectTypes durch den WFS bereitgestellt werden können. Im Minimalfall sind das für den OKSTRA-WFS:

- gml:AbstractFeatureType,
- gml:PointType (falls der Datenbestand punktförmige Geometrien enthält),
- gml:LineStringType (falls der Datenbestand linienförmige Geometrien enthält),
- gml:PolygonType (falls der Datenbestand flächenförmige Geometrien enthält).

8.1.1.6 ogc:Filter_Capabilities

Folgende GeometryOperands sind mindestens anzugeben:

gml:Envelope, gml:Point, gml:LineString, gml:Polygon

Folgende SpatialOperators sind mindestens anzugeben:

BBOX, Disjoint, Intersects, Within, Contains, Overlaps, Dwithin

LogicalOperators und alle im Filter Encoding (Referenz 4.1) spezifizierten ComparisonOperators sind anzugeben.

Von den ArithmeticOperators ist wenigstens SimpleArithmetic anzugeben.

Von den Id_Capabilities ist wenigstens ogc:EID anzugeben.

8.1.2 Operation: DescribeFeatureType

Die Operation DescribeFeatureType erzeugt ein Schemaprofil des OKSTRA-XML-Schemas für einen oder mehrere FeatureTypes. Die Operation erzeugt ein XML-Schema-Dokument, oder im Fehlerfall ein Exception-Dokument.

Es ist zu beachten, dass die Bildung solcher Schemaprofile im selben Namespace (okstra-Namespace), wie durch DescribeFeatureType gefordert, im Grunde nicht durch den XML Schema Standard abgedeckt ist, da ein Client beim Eintreffen einens Dokuments zu einem Namespace grundsätzlich annehmen darf, dass dieses das korrekte und vollständige Schema zum Namespace beinhaltet. Er darf den Inhalt „cachen" und in Zukunft als das gesamte Schema zum Namespace ansehen. Wenn OKSTRA-WFS-Clients so vorgehen, kann dies zu Fehlersituationen führen.

Im Folgenden wird im Einzelnen das Vorhandensein und die Wirkung einzelner Bestandteile der DescribeFeatureType-Operation beschrieben:

8.1.2.1 wfs:DescribeFeatureType/@outputFormat

Ein OKSTRA-WFS soll „text/xml; subtype=gml/3.1.1" unterstützen.

Das Response-Dokument soll ein XML-Schema-Dokument sein, das nur die Definitionen der durch TypeName geforderten FeatureTypes enthält und ansonsten möglichst nur die dafür benötigten zusätzlichen Schemafragmente.

8.1.2.2 wfs:DescribeFeatureType/wfs:TypeName

Es sollen alle in den Capabilities angebotenen FeatureTypes unterstützt werden.

8.1.3 Operation: GetFeature

Die Operation GetFeature erfragt OKSTRA-Objekte (Features), welche sich entsprechend den Angaben im GetFeature-Request qualifizieren. Die Operation bringt ein gml:FeatureCollection-Dokument zurück, oder im Fehlerfall ein Exception-Dokument.

Das wfs:FeatureCollection-Doument enthält wenigstens

1. das gml:boundedBy-Property und
2. null oder mehr gml:featureMember-Properties mit OKSTRA-Objektversionen entsprechend der Definition von OKSTRA-XML.

Bezüglich der Rückgabe anderer Dokumentenformate siehe 8.1.3.2.

Im Folgenden wird im Einzelnen das Vorhandensein und die Wirkung einzelner Bestandteile der GetFeature-Operation beschrieben:

8.1.3.1 wfs:GetFeature/@resultType

Die beiden Werte „results" und „hits" müssen unterstützt werden.

Bei „hits" sollen nur die primär durch Query-Konstrukte gefundenen Features gezählt werden. Zusätzliche Features, welche möglicherweise durch *traverseXlinkDepth* oder *XlinkPropertyName* hinzutreten, sollen nicht gezählt werden.

8.1.3.2 wfs:GetFeature/@outputFormat

Der OKSTRA-WFS muss den in der OGC-WFS-Spezifikation geforderten Wert „text/xml; subtype=gml/3.1.1" verstehen. Dies ist auch der Standardwert beim Weglassen des Parameters.

Falls ein OKSTRA-WFS neben der vom OGC-WFS-Standard geforderten Rückgabe eines wfs:FeatureCollection-Dokuments auch ein okstra:OKSTRAObjektmenge-Dokument entsprechend der OKSTRA XML abliefern kann, so ist dies mittels des Attributs

outputFormat="text/xml; subtype=okstra/1.012"

anzufordern. Die Möglichkeit, dies zu tun, ist in Capabilities-Response-Dokument anzuzeigen.

8.1.3.3 wfs:GetFeature/@maxFeatures

Die Implementierung des Attributs ist verpflichtend. Ohne Angabe des Attributs, sind alle qualifizierenden Features abzuliefern.

8.1.3.4 wfs:GetFeature/@traverseXlinkDepth

Die Implementierung des Attributs ist verpflichtend.

Wenn durch wfs:XlinkPropertyName (siehe 8.1.3.12) im Einzelfall nicht anders bestimmt wird, werden rekursiv xlink:href-Angaben aller Properties aller durch Query-Konstrukte sich qualifizierender Features bis zur angegebenen Tiefe verfolgt. Die dabei gefundenen Features werden (soweit sie nicht bereits darin vorkommen) ans Ende der erzeugten wfs:FeatureCollection gestellt.

traverseXlinkDepth="0" ist der Standardwert und bedeutet keine Auflösung, d.h. Angabe des xlink:href-Attributs im Property.

Es ist nur erforderlich, Verweise innerhalb des Datenbestands aufzulösen, sofern dies im Capabilities-Response-Dokument so angegeben wurde, siehe 8.1.1.3.

Achtung: Das OKSTRA-WFS-Profil weicht von der OGC-WFS-Spezifikation ab, weil die in letzterer geforderte Strategie der Einbettung der gefundenen Features zu fehlerhaften Resultatdokumenten führt.

8.1.3.5 wfs:GetFeature/@traverseXlinkExpiry

Der Wert des Attributs kann ignoriert werden, soweit keine datenhaltungsübergreifende Verfolgung von xlink:href implementiert ist.

8.1.3.6 wfs:GetFeature/wfs:Query/@typeName

Das Attribut gibt den FeatureType an, der gesucht werden soll, z.B. okstra:Abschnitt.

Die im OGC-WFS-Dokument angedeutete Syntax für *Joins* und *Alias*-Definitionen, ist im Standard nicht hinreichend für eine Implementierung spezifiziert. Es ist nicht erforderlich, diese Funktionalität zu implementieren.

8.1.3.7 wfs:GetFeature/wfs:Query

Jedes Query-Element führt entsprechend seiner Parametrisierung durch die im Folgenden beschriebenen Elemente und Attribute zu einer Suchoperation auf dem Datenbestand.

Die gefundenen Features werden in der Reihenfolge der Query-Elemente in der wfs:FeatureCollection abgelegt. Treten infolge sich überschneidender Queries Feature-Exemplare mehrfach auf, so wird nur das erste Feature explizit im gml:featureMember eingetragen. Bei allen weiteren wird das xlink:href-Konstrukt eingesetzt, um auf das erste Auftreten zu verweisen.

8.1.3.8 wfs:GetFeature/wfs:Query/@handle

Die Angabe des handle-Attributs dient zur verbesserten Zuodnung von Exceptions zu den einzelnen Query-Angaben bei GetFeature. Die Implementierung ist verpflichtend.

8.1.3.9 wfs:GetFeature/wfs:Query/@featureVersion

Das Attribut wird beim OKSTRA-WFS im Query zur Auswahl nach Stichtagen in einem historisierten Bestand eingesetzt.

Folgende Werte sind zu unterstützen:

ALL — Es erfolgt keine Auswahl nach Stichtagen. Eine historienbezogene Selektion kann über die OKSTRA-Properties „gueltig_von" und „gueltig_bis" erfolgen.

CURRENT — Nur der letzte Stand in der Datenhaltung wird sichtbar. Dies betrifft auch relationale Verknüpfungen, d.h. nur xlink:href-Verknüpfungen zu Features des letzten Stands werden nachgewiesen.

CURRENT ist der Standardwert beim Weglassen von featureVersion.

JJJJ-MM-TT — Z.B.: 2008-08-15. Der Stand zum angegebenen Stichtag wird sichtbar. Dies betrifft auch relationale Verknüpfungen, d.h. nur xlink:href-Verknüpfungen zu Features mit Gültigkeit zum angegebenen Stichtag werden nachgewiesen.

Hinweis: Der Einsatz von featureVersion im beschriebenen Sinne ist syntaktisch mit dem WFS-Schema verträglich, stimmt aber nicht mit der schriftlichen Spezifikation überein.

8.1.3.10 wfs:GetFeature/wfs:Query/@srsName

Die Angabe eines der unterstützten (siehe 8.1.4) Koordinatensysteme bewirkt die Ausgabe der Geometrien in diesem angegebenen Koordinatensystem.

Die in der WFS-Spezifikation geforderte Angabe in der Form EPSG:<code> ist nicht gültig, weil sie nicht mit dem Datentyp anyURI übereinstimmt – sie also nicht zu unterstützen. Die anderen Formen der srsName-Angabe sind zu unterstützen.

Zusätzlich zu unterstützen (und bevorzugt zu verwenden) ist die urn-Schreibweise

urn:ogc:def:crs:EPSG::<code>,

welche im Referenzdokument 4.3 beschrieben wird.

8.1.3.11 wfs:GetFeature/wfs:Query/wfs:PropertyName

Die im OGC WFS vorgesehene Möglichkeit zur Einschränkung der Struktur der einzelnene Features auf explizit angegebene PropertyNames ist zu unterstützen. Qua Schema obligatorische Properties sind immer auszugeben.

8.1.3.12 wfs:GetFeature/wfs:Query/wfs:XlinkPropertyName

Alternativ zu PropertyName ist die Verwendung von XlinkPropertyName vorzusehen.

Durch dieses Element werden wie bei PropertyName einzelne Properties selektiert. Zusätzlich ist es aber durch das Attribut „traverseXlinkDepth" möglich, xlink:href-Werte im angesprochenen Property bis zur gewünschten Tiefe aufzulösen. Die Angabe hat Vorrang vor einer etwaigen globalen Angabe bei GetFeature, siehe 8.1.3.4.

PropertyName und XlinkPropertyName können gemischt auftreten. XlinkPropertyName wirkt wie PropertyName bei Properties ohne xlink:href-Wert.

Wie bei der globalen Definition in GetFeature müssen nur datenhaltungslokale Referenzen aufgelöst werden.

8.1.3.13 wfs:GetFeature/wfs:Query/ogc:Filter

Das Filterelement, soll entsprechend den Angaben in 8.1.6 interpretiert werden.

8.1.3.14 wfs:GetFeature/wfs:Query/ogc:SortBy

Ein OKSTRA-WFS braucht SortBy nicht zu unterstützen[11].

8.1.4 Koordinatenreferenzsysteme

Für OKSTRA-WFS sind folgende Koordinatenreferenzsysteme verbindlich:

Name	Datum	Ellipsoid	Projektion	Koordinatenreihenfolge	EPSG Code
ETRS89	ETRS89	GRS80	Keine (geographisch)	North-East	EPSG:4258
WGS84	WGS84	WGS84	Keine (geographisch)	North-East	EPSG:4326
DHDN-GKz z=2,3,4,5*)	Potsdam	Bessel	Transverse Mercator	North-East	EPSG:n, wobei n=31464+z
ETRS-UTM32N	ETRS89	GRS80	Transverse Mercator	East-North	EPSG:25832

*) Nur die Gauß-Krügerstreifen, in denen die Daten liegen, sind verpflichtend.

Die genannten Koordinatensysteme können in allen Operationen eines OKSTRA-WFS verwendet werden, z.B. in Filter-Angaben oder als gewünschtes System beim Erfragen von Features.

Zur Spezifikation von Koordinatensystemen soll an allen Stellen die urn-Schreibweise

[11] Leider ist das Fehlen dieser Eigenschaft in den Capabilities nicht nachweisbar. Da in der nächsten WFS-Version (2.0) Sorting explizit nicht verpflichtend ist, erklären wir dies hier bereits im Vorgriff.

urn:ogc:def:crs:EPSG::<code>,

welche im Referenzdokument 4.3 beschrieben wird, eingesetzt werden.

Beispiel:

```
<gml:Point srsName= "urn:ogc:def:crs:EPSG::31466">
…
</gml:Point>
```

Für den Fall, dass ein OKSTRA-WFS als Ausgabeformat auch ein okstra:OKSTRAObjektmenge-Dokument liefern kann (siehe 8.1.1.3 und 8.1.3.2), müssen auch die in T0006 verlangten Koordinatenreferenzsysteme im adv-Namespace unterstützt werden.

8.1.5 Identifier, gml:id und xlink:href Verweise

Objekt-Identifier für einen OKSTRA-WFS werden verwendet zur Bezeichnung von Features (OKSTRA-Objektversionen). Sie treten auf im gml:id-Attribut und in xlink:href-Referenzen.

Objekt-Identifier müssen eindeutig und stabil sein.

Aus syntaktischen Gründen müssen die Objekt-Identifier mit einem eingeschränkten Zeichensatz auskommen und zwar den Zeichen, die für den ID Attributtyp von XML 1.0 definiert sind. Das Alphabet umfasst etwa die ASCII-Buchstaben, Ziffern, Bindestrich „-" und Unterstrich „_". Zusätzlich ist der Punkt gestattet, aber diesen benutzt das OKSTRA-WFS-Profil für einen Sonderzweck. Objekt-Identifer müssen mit einem alphabetischen Zeichen anfangen.

Insgesamt bestehen Objekt-Identifikatoren eines OKSTRA-WFS aus drei Teilen wie folgt:

namensraum-präfix.*feature-type-name*.*lokal-eindeutiger-identifier*

lokal-eindeutiger-identifier steht für einen stabilen und innerhalb des FeatureTypes eindeutigen Identifier für Features (also bei Historisierung für Objektversionen), welchen die jeweilige Datenhaltung vergibt.

feature-type-name ist der Klartext-Name des FeatureTypes, der bezeichnet wird.

namensraum-präfix bezeichnet den OKSTRA-WFS-Service, welcher das Feature beherbergt. Die Namen (und ihre Umsetzung in das URL des zugehörigen OKSTRA-WFS) werden zentral in einem Register durch die OKSTRA-Pflegestelle verwaltet, welches durch einen Web-Service zugänglich ist. Ein neuer Präfix kann auf der OKSTRA-Webseite beantragt werden[12].

Für ausschließlich lokal verwendete Daten kann der Präfix beliebig gewählt werden, wobei ein „x-" voranzustellen ist, um die lokale Natur des Präfix zu dokumentieren.

Die drei Teile werden durch Punkte getrennt.

Das gml:id-Attribut wird also wie folgt geschrieben:

gml:id="*namensraum-präfix*.*feature-type-name*.*lokal-eindeutiger-identifier*"

[12] Dieser Dienst steht zum Zeitpunkt der Schriftlegung des OKSTRA-WFS-Profils noch nicht zur Verfügung. Das Profil soll um entsprechende Details fortgeschrieben werden, sobald der beschriebene Registry-Dienst eingerichtet ist. Ein OKSTRA-WFS ohne externe Referenzen benötigt den Dienst nicht.

Dieselbe Schreibweise[13] wird auch bei Objekt-Referenzen eingesetzt. Die verbindliche Syntax ist hier:

xlink:href="urn:x-okstra:object:*namensraum-präfix*.*feature-type-name*.*lokal-eindeutiger-identifier*"

Über den beschriebenen Registry-Dienst kann die Client-Software das URL des OKSTRA-WFS-Service erfragen, welcher das referierte Objekt liefern kann. So können serverübergreifende Referenzen dokumentiert werden.

8.1.6 Ausnahmebehandlung

Beim Auftreten von Fehlersituationen erzeugt der OKSTRA-WFS ein ExceptionReport-Element entsprechend dem OGC-WFS-Standard anstelle der normalen Rückgabe. Bei Verwendung des SOAP-Protokolls sind die hierzu gemachten Aussagen im OGC-WFS-Standard zu beachten.

Codes und Fehlertexte sind, obwohl dies wünschenswert wäre, z.Z. im WFS-Standard nicht festgelegt. Es folgt deshalb hier für den OKSTRA-WFS eine Festlegung der möglichen Werte für das Attribut

exceptionCode="..."

am Element <Exception>.

Der zugehörige <ExceptionText> ist wird hier nicht wörtlich festgelegt. Er soll jedoch in deutscher Sprache angegeben werden und so aussagekräftig formuliert werden, dass die Fehlersitutation damit klar beschrieben ist.

exceptionCode="..."	**Bedeutung**
RequestValidationError	Der POST- oder SOAP-Request ist kein gültiges XML oder validiert nicht gegen das Request-Schema.
NoOperation	Der REQUEST-Parameter bei GET-Request fehlt.
UnknownOperation	Der REQUEST-Parameter beim GET-Request spezifiziert eine ungültige Operation.
OperationNotSupported	Die verlangte Operation entspricht dem WFS-Standard, wird aber nicht unterstützt.
VersionNotSupported	Die verlangte WFS-Version wird nicht unterstützt.
OutputFormatNotSupported	Das verlangte Ausgabeformat wird nicht unterstützt.
FeatureTypeNotSupported	Der verlangte FeatureType wird nicht unterstützt.
SRSNotSupported	Das verlangte Koordinatenreferenzsystem wird nicht unterstützt.
ParameterInconsistency	Es wurden inkonsistente oder widersprüchliche Parameter angegeben, z.B. bei GET FILTER= zusammen mit BBOX=
InvalidPropertyName	Der angegebene PropertyName entspricht nicht dem OKSTRA-Schema oder ist syntaktisch ungültig.
InvalidLiteralValue	Der Wert einer Konstante in einem Filterausdruck ist unzu-

[13] Es wird empfohlen, den Namensraum „okstra" bei IANA zu beantragen. Sobald der Namesraum bewilligt wird, kann „x-okstra" durch „okstra" ersetzt werden.

	lässig.
InvalidGeometryLiteralValue	Der Wert einer Geometriekonstanten ist unzulässig.
FilterOperatorNotAvailable	Der verlangte Operator in einem Filterausdruck wird nicht unterstützt.
InvalidParameterValue	Ein Parameterwert des Requests ist ungültig. (Dies ist eine Sammelfehlermeldung für sonstige Request-Fehler.
DatabaseAccessError	Ein Zugriff auf die Datenbank schug fehl.
SystemError	Konfigurationsinkonsistenz oder sonstiger unerwarteter Fehler.
ImplementationSpecificError	Fehler, der sich in einer Implementierung ergibt und der sich nicht sinnvoll auf die oben angegebenen Codes abbilden lässt.

Es ist zu erwarten, dass sich aus der Implementierung und in der Praxis zusätzliche Fehlersitutation ergeben werden, welche durch die oben angegebene Liste nicht adäquat abgedeckt sind. Diese sind zunächst auf „ImplementationSpecificError" abzubilden. In einer zukünftigen Überarbeitung der OKSTRA-WFS-Spezifikation sind diese Fehlersituationen dann in die obige Liste einzuarbeiten.

8.1.7 Protokolle

Der *OKSTRA®-WFS lesend* muss für alle unterstützten Operationen die Protokolle http/POST, http/GET und SOAP/POST in vollem Umfang unterstützen.

8.2 Profil des Filter Encoding

Das Filter Encoding für den OKSTRA-WFS wird nahezu vollständig als Implementierung vorausgesetzt. Es werden aufgrund der starken relationalen Vernetzung des OKSTRA sogar weitreichende Erweiterungen, besonders beim Zugriff auf Properties, gefordert.

Im Einzelnen:

8.2.1 Property-Zugriffe

8.2.1.1 Erweiterte XPath-Semantik

In Erweiterung der Semantik der geforderten XPath-Notation im Element ogc:PropertyName in Referenz 4.1, soll diese beim OKSTRA-WFS auch Anwendung finden, um auf Eigenschaften zugreifen zu können, die referenziell (über xlink:href) angebunden sind.

Es ist dabei syntaktisch unerheblich, ob ein objekthafter Property-Wert eingebettet vorliegt oder über xlink:href angebunden ist.

Ein solcher Zugriff repräsentiert in seiner Gesamtheit die Menge der so vom Objekt (Feature) aus erreichten Werte.

8.2.1.2 Positionsprädikate

Entsprechend den Anforderungen in Referenz 4.1 kann an jedem Schritt ein Positionsprädikat der Form [<zahl>] stehen, die bestimmt, welches der im XPath-Schritt multipel auftretenden Elemente gemeint ist. <zahl> ist dabei die Position ab 1 zählend.

In den Fällen, wo im OKSTRA nur eine mengenmäßige Relation besteht, ist die Reihenfolge, die der Auswahl unterliegt, implementierungsabhängig. Obwohl implemetierungsabhängig, muss die Zuordnung von Zahlen zu Elementinhalten aber eindeutig und stabil sein. Die Reihenfolge braucht allerdings nur stabil zu sein, solange keine Änderungen an der Datenbank vorgenommen werden.

Sieht der OKSTRA eine geordnete Relation vor, so gilt die Reihenfolge dieser Ordnung.

8.2.1.3 Allgemeine Prädikate

XPath-Ausdrücke in Property-Zugriffen sollen Prädikate im Sinne der XPath-Syntax unterstützen. Die Unterstützung ist erforderlich in Pfad-Schritten, welche den Objekten (Features) entsprechen. Die Ausdruckssyntax soll umfassen:

- XPath-Property-Zugriffe (rekursiv definiert)
- Konstanten (Typen: numerisch, Text, ISO8601-Date/DateTime-Formate in Text)
- Vergleiche
- and, or, not(...)
- Klammern

8.2.1.4 Geometrische Funktionen

Es wird empfohlen, zusätzlich geometriewertige Funktionen, genannt *ImplictGeometryFunctions*, bereitzustellen, welche die aus dem Netzbezug implizit vorhandene Geometrie bereitstellen.

Alle Funktionen berechnen aus dem gegebenen „nodeset", welches je nach Funktion bestimmte Features repräsentieren muss, und einem Stichtag eine gml:MultiGeometry bestimmter Dimensionalität.

Signatur	Resultattyp	Einschränkung für nodeset
okwfs:punktAusPunktobjekt(nodeset,stichtag)	MultiPoint	Punktobjekt
okwfs:linieAusStreckenobjekt(nodeset,stichtag)	MultiCurve	Streckenobjekt
okwfs:strassenachse(nodeset,stichtag)	MultiCurve	Strasse

Das nodeset kann fehlen. In diesem Falle wird der Kontext herangezogen, welcher dann den Einschränkungen genügen muss.

Das Argument „stichtag" steht für ein Datum, gegeben als Text in ISO8601-Format. Es wählt von den möglichen Geometrien in einer Relation zwischen historisierten Objekten die zum Stichtag gültigen aus. Das Argument kann fehlen, wenn ein Stichtag auf andere Weise bekannt ist, z.B. über das featureVersion-Attribut.

Falls diese Funktionen zur Verfügung stehen, müssen sie der hier vorgegebenen Benennung folgen und in den ows:ExtendedCapabilities (siehe 8.1.1.3) erklärt sein.

8.2.2 Geometrische Prädikate

Folgende in 4.1 definierte Prädikate sind obligatorisch für einen OKSTRA-WFS:

- BBOX
- Disjoint
- Intersects

- Within
- Contains
- Overlaps
- DWithin

Für den Fall, dass der Property-Zugriff auf die Geometrie über XPath-Notation eine Menge von Geometrien beschreibt, so werden diese im Prädikat implizit zu einer Collection zusammengefasst.

8.2.3 Skalare Vergleiche

Alle in 4.1 definierten skalaren Vergleiche sind obligatorisch für einen OKSTRA-WFS.

Für den Fall, dass der Property-Zugriff auf den Wert eines Operanden über XPath-Notation eine Menge von Werten beschreibt, so wird der Vergleich im Sinne einer Existenzquantifizierung („es gibt ein...") durchgeführt.

Der Datentyp des Vergleichs ergibt sich aus dem Typ des teilnehmenden Property-Zugriffs gemäß dem OKSTRA-Schema. Vergleiche werden durchgeführt für numerische Größen, Texte und DateTime-Angaben.

8.2.4 Logik

ogc:And, ogc:Or und ogc:Not sind zu unterstützen.

8.2.5 Ausdrücke, Literale, Funktionen

Die Grundrechenarten ogc:Add, ogc:Sub, ogc:Mul und ogc:Div sind zu unterstützen. Sie sind nur auf numerische Größen anwendbar.

Literale können Texte, numerische Größen, Date/DateTime-Größen (in ISO8601-Syntax) und Geometrien (in GML-Syntax) enthalten. Letztere sind nur bei der Anwendung in ogc:Functions erforderlich, welche beim OKSTRA-WFS keine Rolle spielen.

ogc:Function wird im OKSTRA-WFS nicht eingesetzt, darf aber als Erweiterung vorhanden sein, falls in den Filter_Capabilities erklärt.

8.2.6 Identifier-Filter

Bezüglich der besonderen Syntax für Filter nach Objektidentifier ist die Syntaxvariante mit ogc:GmlObjectId verpflichtend.

8.3 Einschränkungen im OKSTRA®-Schema

Dieser Abschnitt beschreibt Normalisierungen im OKSTRA-Schema, welche

- bei der Formulierung von Filterausdrücken – insbesondere beim Zugriff auf Eigenschaften über Xpath – vorausgesetzt werden dürfen und
- welche bei der Abgabe von OKSTRA-Daten durch den OKSTRA-WFS eingesetzt werden.

8.3.1 Schlüsseltabellen

Ein OKSTRA-WFS hat die Freiheit, die Inhalte von Schlüsseltabellen entweder in die Properties einzubetten oder als getrennte Objekte im GetFeature-Response aufzuführen und diese aus den Properties per xlink:href zu referenzieren.

Hinweis zur Referenzierung:

Bei der Rückgabe einer okstra:OKSTRAObjektmenge ist durch das Schema geregelt, wo Schlüsseltabellenwerte eingesetzt werden können, nämlich als okstra:okstraKeyValue-Properties folgend auf die okstra:okstraObjekt-Properties. Verwendungen dieser Schlüsseltabellenobjekte erfolgen durch xlink:href über dokumentlokale Identifikatoren.

Bei der dem WFS-Standard entsprechenden Rückgabe einer wfs:FeatureCollection muss etwas anders vorgegangen werden. Da Schlüsseltabellen laut OKSTRA XML keine Features sind, können sie nicht einfach als gml:featureMembers an das Ende der wfs:FeatureCollection gestellt werden. Schlüsseltabellenobjekte müssen daher zumindest an einer Stelle in das verwendende Property eingebettet werden. Dabei kann ein lokaler Identifier in gml:id vergeben werden, der dann an anderer Stelle von einem schlüsseltabellenwertigen Property per xlink:href referenziert werden kann.

8.3.2 Abstrakte Verweise

Abstrakte Verweise werden durch den OKSTRA-WFS nicht unterstützt. Sie sind zu ersetzen durch serverübergreifende xlink:href-Verweise. Hierzu siehe die Definitionen zu Identifikatoren in 8.1.4.

8.3.3 Bereichsobjekte

Da kein normalisiertes OKSTRA-Profil gefunden werden kann, das allen Implementierungen gerecht wird, soll durch den OKSTRA-WFS das OKSTRA-konform interpretierte, tatsächlich in den Straßendatenbanken vorhandene Modell angeboten werden. Das verwendete Modell ist in den Capabilties anzuzeigen, siehe 8.1.1.3.

8.3.4 Streckenobjekte

Streckeneigenschaften werden immer auf einer Strecke verortet, die aus 1 bis beliebig vielen Teilabschnitten besteht. Ein Teilabschnitt kann nur zu einem einzigen Zweck verwendet werden.

8.4 Allgemeine zusätzliche Forderungen

8.4.1 Performanz

Es wird empfohlen, dass Queries an einen OKSTRA-WFS mit mindestens 500 Features pro Sekunde im Durschnitt beantwortet werden.

8.4.2 Größenbeschränkungen

Es wird empfohlen, die Implementierung eines OKSTRA-WFS so auszulegen, dass keine Beschränkungen bezüglich der Größe von Request- oder Response-Dokumenten bestehen, z.B. der Größe der erfragbaren FeatureCollections.

8.4.3 Authentifizierung und Autorisierung

Es ist nicht Aufgabe des OKSTRA-WFS, eine Zugriffskontrolle durchzuführen.

9 OKSTRA®-WFS transaktional (normativ)

In diesem Kapitel erfolgt die Definition des *OKSTRA®-WFS transaktional,* welcher die Funktionalität des *OKSTRA®-WFS lesend* auf schreibende Zugriffe erweitert.

Ein ***OKSTRA®-WFS transaktional*** ist ein **OGC WFS transactional Version 1.1.0**, der die gesamte obligatorische Funktionalität des *OKSTRA®-WFS lesend* aufweist, wie sie in Kapitel 8 dargelegt wurde, **und** der zusätzlich **alle** Anforderungen, aus dem vorliegenden Kapitel erfüllt.

Die verwendeten Notationen entsprechen denen im Kapitel 8.

9.1 Profil des Web Feature Service

Dieser Teil der OKSTRA-WFS-Spezifikation für den *OKSTRA-WFS transaktional* behandelt die zusätzlichen, über die im Abschnitt 8.1 hinaus vorzunehmenden Definitionen, welche die OGC-WFS-Spezifikation betreffen.

Als erstes werden die Operationen des WFS behandelt (GetCapabilities, DescribeFeatureType, GetFeature, GetFeatureWithLock, LockFeature, Transaction) im Einzelnen festgelegt.

Die Aussagen zu Koordinatensystemen, Identifier, Ausnahmebehandlung und Protokollen sind wie bei *OKSTRA-WFS lesend.*

9.1.1 Operation: GetCapabilities

Im Folgenden werden die Unterschiede bezüglich des WFS_Capabilities-Dokuments dargelegt.

9.1.1.1 ows:ServiceIdentification

Identisch.

9.1.1.2 ows:ServiceProvider

Identisch.

9.1.1.3 ows:OperationsMetadata

Es sind wenigstens die Operationen GetCapabilities, DescribeFeatureType, GetFeature, GetFeatureWithLock, LockFeature und Transaction zu beschreiben.

Alle Operationen müssen unter ows:DCP/ows:http ein URL für ows:Get und ows:Post vorsehen.

Bei GetFeatureWithLock ist das *outputFormat* „text/xml; subtype=gml/3.1.1" anzugeben. *resultType* soll wie bei GetFeature sein.

LockFeature soll als *lockAction* „ALL" und „SOME" angeben.

Bei der Operation Transaction ist wenigstens *inputFormat* „text/xml; subtype=gml/3.1.1" anzugeben. Beim Parameter *idgen* ist wenigstens „GenerateNew" erforderlich. Bei *releaseAction* soll „ALL" und „SOME" angeben sein.

Zusätzlich kann es erforderlich sein, dass als letztes Element von ows:OperationsMetadata das Element ows:ExtendedCapabilities angegeben wird. Siehe hierzu 8.1.1.3.

9.1.1.4 wfs:FeatureTypeList

Der Abschnitt ist wie bei *OKSTRA-WFS lesend* zu erzeugen.

Zusätzlich gilt:

Wenn das wfs:Operations Element angegeben wird, so soll für die FeatureTypes, welche zum Schreiben freigegeben sind, wenigstens

```
<wfs:Operation>Query</wfs: Operation>
<wfs:Operation>Insert</ wfs: Operation>
<wfs:Operation>Update</ wfs: Operation>
<wfs:Operation>Delete</ wfs: Operation>
<wfs:Operation>Lock</ wfs: Operation>
```

enthalten sein.

9.1.1.5 wfs:SupportsGMLObjectTypeList

Identisch.

9.1.1.6 ogc:Filter_Capabilities

Identisch.

9.1.2 Operation: DescribeFeatureType

Identisch.

9.1.3 Operation: GetFeature

Identisch.

9.1.4 Operation: GetFeatureWithLock

Die Operation GetFeatureWithLock ist weitgehend identisch zu GetFeature.

Es wird versucht, ein Lock auf alle selektierten Feature-Instanzen zu nehmen, einschließlich derer welche durch Xlink-Verfolgung automatisch hinzugetreten sind. Falls dies gelingt, wird das vergebene Lock im Attribut *lockId* der wfs:FeatureCollection nachgewiesen. Falls das Lock nicht genommen werden kann, wird eine Ausnahme erzeugt.

Zum Nachweis von Locks siehe 9.1.4.2.

9.1.4.1 wfs:GetFeatureWithLock/@outputFormat

Aufgrund des zwingenden Nachweises des Locks über das lockId-Attribut, ist nur das outputFormat „text/xml; subtype=gml/3.1.1" für GetFeatureWithLock geeignet. Andere Formate sollten deshalb nicht angeboten werden.

9.1.4.2 wfs:GetFeatureWithLock/@expiry

Das Attribut gibt an, nach wievielen Minuten das Lock (verteten durch das LockId) automatisch aufgegeben wird, falls es nicht vorher benutzt wurde. Der Defaultwert ist „5".

9.1.5 Operation: LockFeature

Die Operation LockFeature versucht Locks für spätere schreibende Zugriffe auf selektierte Feature-Instanzen zu nehmen. Die Selektion erfolgt über einen Filter. Die Operation kann so parametriesiert werden, dass entweder

- alle dem Filter genügenden Features gelockt werden, wobei die Operation scheitert, wenn einige der qualifizierenden Features bereits andere Locks besitzen, oder
- nur die Features ein Lock erhalten, welche noch kein anderweitiges Lock aufweisen. Diese Variante scheitert nur, wenn alle qualifizierenden Features bereits vorher gelockt sind.

Bei Erfolg wird das Lock durch eine LockId im Response der Operation nachgewiesen. Bei Nichterfolg wird eine Ausnahme erzeugt.

Die Locks werden nur für eine gewisse Zeit gewährt, die im Request-Paket durch den Parameter *expiry* angegeben werden kann. Danach verfallen die nachgewiesenen LockIds – ihre Verwendung in schreibenden Operationen führt dann zu einer Ausnahmebehandlung.

Die Locks in der Spezifikation des OGC-WFS sind von ihrer Funktionalität her als klassische, exklusive („pessimistische") Locks aus der Datenbanktechnologie gemeint. Während der Lebensdauer der Locks steht das Feature für schreibende Operationen ausschließlich dem Client zur Verfügung, der Kenntnis der „LockId" hat.

Für den OKSTRA-WFS sollen neben diesen „pessimistischen" Locks auch „optimistische" Locks zugelassen werden. In diesem Falle wird durch LockFeature der jüngste (also von der Zeitachse her größte) Zeitstempel aller sich qualifizierenden Features ermittelt und im LockId nachgewiesen. Bei nachfolgenden schreibenden Operationen wird geprüft, ob der Zeitstempel des Features kleiner oder gleich dem LockId ist. Ist er es nicht, wird die schreibende Operation (die gesamte Transaction-Operation) zurückgewiesen.

Die „optimistische" Locking-Strategie muss zwingend im Capabilities-Response-Dokument (in ows:OperationsMetadata/ows:ExtendedCapabilities) des OKSTRA-WFS bekannt gemacht werden, siehe hierzu 9.1.1.3.

Nicht normativer Hinweis: Im Falle von „optimistischer" Locking-Strategie muss der Client darauf vorbereitet sein, bei Scheitern des Schreibens, den gesamten Zyklus des Holens der Daten und der Fortführung zu wiederholen[14].

9.1.5.1 wfs:LockFeature/@expiry

Das Attribut gibt an, nach wievielen Minuten das Lock (verteten durch das LockId) automatisch aufgegeben wird, falls es nicht vorher benutzt wurde. Der Defaultwert ist „5".

9.1.5.2 wfs:LockFeature/@lockAction

Der OKSTRA-WFS unterstützt die Werte ALL und SOME.

Im Falle ALL folgt eine Ausnahme, falls es nicht gelingt, alle selektierten Features mit einem Lock zu versehen. Bei SOME folgt eine Ausnahme nur, wenn kein Feature gelocked werden konnte.

Die Ausnahme bei SOME erfolgt nur, wenn das Nichtzustandekommen eines Locks auf vorliegende fremde Locks zurückgeht. Für den Fall, dass der Filter die leere Menge selektiert, soll ein *WFS_LockFeatureResponse* erzeugt werden, der weder *FeaturesLocked* noch *FeaturesNotLocked* enthält.

Im Falle einer „optimistischen" Lockingstrategie ist es immer möglich, Locks für alle selektierten Features anzufordern. Die *lockAction* "SOME" / "ALL" hat in diesem Zusammenhang keine Bedeutung, da es niemals zu einer Ausnahme auf Grund von vorherigen Sperren kommen kann.

9.1.5.3 wfs:LockFeature/wfs:Lock

Jedes Lock-Element führt entsprechend seiner Parametrisierung durch die im Folgenden beschriebenen Elemente und Attribute zu einer Suchoperation auf dem Datenbestand.

Die gefundenen Features werden je nach lockAction (ALL|SOME) und Locking-Strategie („pessimistisch", „optimistische") mit Locks versehen. Werden infolge sich überschneidender Filter Feature-Exemplare mehrfach gelocked, so wird dies behandelt als sei das Lock nur einmal vergeben.

[14] Das muss er im Grunde auch bei „pessimistischen" Locks. Diese können z.B. durch „time-out" infolge des Überschreitens des expiry-Zeitraums verloren gehen.

9.1.5.4 wfs:LockFeature/wfs:Lock/@handle

Die Angabe des handle-Attributs dient zur verbesserten Zuodnung von Exceptions zu den einzelnen Lock-Angaben bei LockFeature. Die Implementierung ist verpflichtend.

9.1.5.5 wfs:LockFeature/wfs:Lock/@typeName

Das Attribut gibt den FeatureType an, der gesucht werden soll.

9.1.5.6 wfs:LockFeature/wfs:Query/@featureVersion – nicht vorhanden!

Dieses Attribut **steht** bei der Operation LockFeature **nicht zur Verfügung**.

Eine Auswahl nach Stichtagen kann daher bei der Locking-Operation nur explizit über die OKSTRA-Properties „gueltig_von" und „gueltig_bis" erfolgen.

9.1.5.7 wfs:LockFeature/wfs:Lock/ogc:Filter

Das Filterelement, soll entsprechend den Angaben in 8.1.6 interpretiert werden.

9.1.5.8 wfs:WFS_LockFeatureResponse

Im Response-Dokument wird im Einzelnen (durch ogc:GmlObjectId-Elemente) nachgewiesen, welche Features ein Lock erhalten haben und welche nicht.

Die Liste wfs:FeaturesNotLocked ist bei lockAction="ALL" naturgmäß immer leer und wird daher weggelassen.

Wird infolge des Filters kein Feature mit einem Lock versehen, so wird sowohl wfs:FeaturesLocked als auch wfs:FeaturesNotLocked weggelassen.

9.1.6 Operation: Transaction

Die Operation Transaction führt schreibende Operationen auf den FeaturesTypes aus, für die dies gestattet ist. Welche dies im Einzelnen sind, ist im Capabilities-Response-Dokument beschrieben, siehe 9.1.1.4.

Ein OKSTRA-WFS gestattet es, vor verändernden Operationen Locks auf den betroffenen Features zu nehmen. Neben der im OGC-WFS vorgesehenen „pessimistischen" Locking-Strategie ist auch eine „optimistische" Strategie vorgesehen. Siehe hierzu die Ausführungen in 9.1.5.

Grundsätzlich werden die Einzeloperationen einer Transaction-Operation (also Insert, Update und Delete) in der angegebenen Reihenfolge ausgeführt. Die gesamte Operation erfolgt (unabhängig von der Existenz möglicher Locks) in jeder Sicht von außerhalb atomar und isoliert, d.h.:

- Parallele lesende und schreibende Operationen sehen, solange eine Transaction durchgeführt wird, grundsätzlich den Zustand vor der Transaction oder den Zustand nach der Transaction (keine „dirty reads", keine „lost updates"),
- Die Einzeloperationen gelingen entweder ganz oder gar nicht. D.h. wenn die Transaction fehlschlägt, ist der Zustand vor der Transaction-Operation wieder hergestellt.

Wenn ein Lock zur Verfügung steht und angegeben wird, so wird für die Features, für die das Lock gilt, sichergestellt, dass sie nicht durch andere Transactions verändert wurden. Bei „pessimistischer" Locking-Strategie wird dies erzwungen, indem das Feature für andere Transactions gesperrt wird, bei „optimistischer" Strategie, wird die eigene Transaction zurückgewiesen, falls eine Veränderung an den betroffenen Features festgestellt wird.

Veränderte Feature-Instanzen werden grundsätzlich von ihrem Lock befreit. Für die restlichen Features, die durch dasselbe Lock gesperrt sind, gibt es eine Option, das Lock zu behalten oder aufzulösen.

9.1.6.1 wfs:Transaction/@releaseAction

Ein OKSTRA-WFS unterstützt beide Werte, ALL und SOME, in der vom OGC-WFS-Standard beschriebenen Art und Weise.

9.1.6.2 wfs:Transaction/wfs:LockId

Falls in einer Transaction LockId nicht angegeben ist, so können mit dieser Transaction alle Features verändert werden, die kein fremdes Lock tragen. Der Versuch einer Veränderung an einem solchen Feature führt zu Abbruch und Rollback der Transaction. Im Falle einer „optimistischen" Lock-Strategie können Features mit Locks verändert werden – der Abbruch erfolgt dann später.

Wenn ein Lock zur Verfügung steht und angegeben wird, so wird für die Features, für die das Lock gilt, sichergestellt, dass sie nicht durch andere Transactions verändert wurden. Bei „pessimistischer" Locking-Strategie wird dies erzwungen, indem das Feature für andere Transactions gesperrt wird, bei „optimistischer" Strategie, wird die eigene Transaction zurückgewiesen, falls eine Veränderung an den betroffenen Features festgestellt wird.

Auch wenn ein Lock angegeben ist, darf der OSKTRA-WFS Features, welche kein fremdes Lock tragen, verändern.

Für die „optimistische" Strategie ist die LockId immer der Zeitstempel, der bei Aufrufs der LockFeature- bzw. GetFeatureWithLock-Operation ermittelt wird. Wenn Features verändert werden (Delete oder Update) wird überprüft, ob die betroffenen Features zu einem Zeitpunkt nach diesem übergebenen Zeitstempel verändert wurden. Falls dies der Fall ist, wird die Transaktion abgebrochen. Wenn nicht, hat kein anderer Prozess diese Daten verändert und die Transaktion kann ausgeführt werden.

9.1.6.3 wfs:Transaction/wfs:Insert

Insert erzeugt neue Feature-Instanzen, aus

- einzelnen Features, gegeben in OKSTRA-XML, oder
- aus einer gml:FeatureCollection, solche Features enthaltend.

Insert kann auch eine okstra:OKSTRAObjektMenge akzeptieren, wenn dieses im inputFormat durch „text/xml; subtype=okstra/1.012" deklariert ist und der OKSTRA-WFS dies in den Capabilities (9.1.1.3) angeboten hat.

Um relationale Beziehungen zwischen Features herstellen zu können, die in einer Transaction zusammen erzeugt werden, muss der OKSTRA-WFS lokale Veweise verarbeiten können. Dabei wird das gml:id-Attribut eines einzufügenden Features mit einem lokal eindeutigen Identifier versehen, das dann aus später auftretenden xlink:href-Properties heraus referiert werden kann. Der WFS setzt dann automatisch dafür die im Datenbestand erzeugten Identifikatoren ein. Ein OKSTRA-WFS muss derartige rückwärtsgerichtete lokale Verweise innerhalb eines Transaction-Dokuments verarbeiten können. Es wird empfohlen, auch Verweise in Vorwärtsrichtung zu unterstützen.

Beispiel:

```
<wfs:Transaction>
  <wfs:Insert>
    <okstra:Netzknoten gml:id="marke1">
      ...
    </okstra:Netzknoten>
    <okstra:Nullpunkt>
      ...
      <okstra:in_Netznoten xlink:href="#marke1"/>
    </okstra:Nullpunkt>
  </wfs:Insert>
</wfs:Transaction>
```

9.1.6.4 wfs:Transaction/wfs:Insert/@idgen

Für den OKSTRA-WFS ist der Wert „GenerateNew" verbindlich.

Andere Werte können zugelassen werden, wenn sie in den Capabilites (9.1.1.3) erklärt wurden.

9.1.6.5 wfs:Transaction/wfs:Insert/@handle

Die Angabe des handle-Attributs dient zur verbesserten Zuodnung von Exceptions zu den einzelnen Bestandteilen der Transaction. Die Implementierung ist verpflichtend.

9.1.6.6 wfs:Transaction/wfs:Insert/@inputFormat

Das Akzeptieren des Formats „text/xml; subtype=gml/3.1.1" ist verpflichtend.

Insert kann auch eine okstra:OKSTRAObjektMenge akzeptieren, wenn dieses im inputFormat durch „text/xml; subtype=okstra/1.012" verlangt wird und der WFS dies in den Capabilities (9.1.1.3) angeboten hat.

9.1.6.7 wfs:Transaction/wfs:Insert/@srsName

Die Implementierung dieses Attributs ist verpflichtend. Es definiert das Default-Koordinatenreferenzsystem für die im Insert-Inhalt angegebenen Geometrien. Dieses kann durch explizite Angaben bei den Geometrien der Features oder FeatureCollections überschrieben werden.

9.1.6.8 wfs:Transaction/wfs:Update

Die Update-Elemente verändern die Properties von selektierten Features.

Die Selektion erfolgt über einen Filterausdruck. Dieser darf einen Datenhaltungszustand voraussetzen, wie der durch die Interpretation der Transaction bis vor das Update-Element erreicht wurde.

9.1.6.9 wfs:Transaction/wfs:Update/@handle

Die Angabe des handle-Attributs dient zur verbesserten Zuodnung von Exceptions zu den einzelnen Bestandteilen der Transaction. Die Implementierung ist verpflichtend.

9.1.6.10 wfs:Transaction/wfs:Update/@typeName

Das Attribut gibt den FeatureType an, dessen Feature-Instanzen verändert werden sollen, z.B. okstra:Abschnitt.

9.1.6.11 wfs:Transaction/wfs:Update/@inputFormat

Das Akzeptieren des Formats „text/xml; subtype=gml/3.1.1" ist verpflichtend.

Ein anderes Format ist im Fall Update nicht sinnvoll und soll deshalb nicht angeboten werden.

9.1.6.12 wfs:Transaction/wfs:Update/@srsName

Die Implementierung dieses Attributs ist nicht verpflichtend.

Das Koordinatensystem geht bereits aus den Geometrien in Property/Value hervor.

9.1.6.13 wfs:Transaction/wfs:Update/wfs:Property

In den Property-Angaben wird jeweils der Name eines Property seinem neuen Wert gegenübergestellt:

```
<wfs:Property>
    <wfs:Name>propertyName</wfs:Name>
    <wfs:Value>neuer Wert</wfs:Value>
</wfs:Property>
```

Properties werden gelöscht, indem das Value-Element ausgelassen wird.

Das Setzen von multiplen Properties soll für den OKSTRA-WFS durch mehrere aufeinander folgende Property-Elemente mit demselben Namen erfolgen.

Das Setzen relationaler Bezüge erfolgt durch

<wfs:Value xlink:href="urn:x-okstra:object:*id*"/>

wobei *id* wie in 8.1.5 beschrieben ist.

Der Wert von xlink:href darf auf den lokalen Bezeichner im gml:id eines vorangehenden durch Insert erzeugten Feature verweisen. Damit können relationale Bezüge zu Features hergestellt werden, welche in derselben Transaction erzeugt wurden.

Es wird empfohlen, auch Vorwärtsverweise auf von Insert erzeugte Features zuzulassen, die erst später im Transaction-Dokument aufgeführt sind.

9.1.6.14 wfs:Transaction/wfs:Update/ogc:Filter

Das Filterelement, soll entsprechend den Angaben in 8.1.6 interpretiert werden.

Fehlt es, so werden alle Instanzen des FeatureTypes verändert.

9.1.6.15 wfs:Transaction/wfs:Delete

Die Delete-Elemente löschen Features. Ihre relationalen Bezüge werden automatisch gepflegt. Etwaige resultierende Verstöße gegen Kardinalitätsbedingungen werden erst nach Abarbeiten der gesamten Transaction geprüft und führen im Fehlerfall zum Scheitern der Transaktion.

Die Selektion erfolgt über einen Filterausdruck. Dieser darf einen Datenhaltungszustand voraussetzen, wie der durch die Interpretation der Transaction bis vor das Delete-Element erreicht wurde.

9.1.6.16 wfs:Transaction/wfs:Delete/@handle

Die Angabe des handle-Attributs dient zur verbesserten Zuodnung von Exceptions zu den einzelnen Bestandteilen der Transaction. Die Implementierung ist verpflichtend.

9.1.6.17 wfs:Transaction/wfs:Delete/@typeName

Das Attribut gibt den FeatureType an, dessen Feature-Instanzen gelöscht werden sollen, z.B. okstra:Abschnitt.

9.1.6.18 wfs:Transaction/wfs:Delete/ogc:Filter

Das Filterelement, soll entsprechend den Angaben in 8.1.6 interpretiert werden.

Fehlt es, so werden alle Instanzen des FeatureTypes gelöscht.

9.1.6.19 wfs:Transaction/wfs:Native

Native muss erkannt werden, muss aber nicht beachtet werden. Die gesamte Funktionalität des OKSTRA-WFS muss ohne Verwendung des Native-Elements erreichbar sein.

9.1.6.20 wfs:TransactionResponse

Das TransactionResponse-Dokument soll wie im OGC-WFS-Standard gefordert implementiert werden.

Die Angabe des TransactionResult-Elements ist nicht erforderlich, da ein OKSTRA-WFS immer atomare Transactions unterstützt.

9.1.7 Ausnahmebehandlung

Es gelten für den OKSTRA-WFS transaktional die Festlegungen aus 8.1.6. Zusätzlich werden die folgenden exceptionCodes festgelegt:

exceptionCode="..."	Bedeutung
InvalidLockId	Das in einer Transaction verwendete LockId ist nicht bekannt oder „expired".
FeatureCannotBeLocked	Auf wenigstens einem der verlangten Features konnte kein Lock genommen werden, weil es bereits anderweitig „gelockt" ist.
LockedFeatureCannotBeAltered	Die Transaction mit dem Versuch, ein Feature zu ändern oder zu löschen, welches durch eine fremdes Lock gesperrt ist, wird abgewiesen.
FeatureAlteredAfterLocking	Nur bei „optimistischem Locking": Die Tansaction mit dem Versuch, ein Feature zu ändern oder zu löschen, welches nach der Locknahme verändert wurde, wird abgewiesen.
DataConsistencyFault	Der Versuch eines Verstoßes gegen das Datenschema durch eine Transaction wird abgewiesen.

9.2 Profil des Filter Encoding

Identisch zu 8.2.

9.3 Einschränkungen im OKSTRA®-Schema

Identisch zu 8.3.

9.4 Allgemeine zusätzliche Forderungen

9.4.1 Performanz

Es wird empfohlen, dass ein OKSTRA-WFS wenigstens 50 Features pro Sekunde einfügen kann. Ein ähnlicher Durchsatz ist für das Update und Delete einzeln adressierter Features zu erreichen.

9.4.2 Größenbeschränkungen

Keine zusätzlichen Anforderungen zu 8.4.2.

9.4.3 Authentifizierung und Autorisierung

Keine zusätzlichen Anforderungen zu 8.4.3.