

**TITLE:**

OGC Web Services Stateless Catalog Profile  
(was Web Registry Service)

**VERSION:**

0.06

**DATE:**

29-AUG-2001

**CATEGORY:**

OGC-IP Draft Candidate Specification (01-062)

**EDITORS:**

Lou Reich – [louis.i.reich@gsfc.nasa.gov](mailto:louis.i.reich@gsfc.nasa.gov)  
Computer Sciences Corp.  
7700 Hubble Dr.  
Lanham-Seabrook MD 20706 U.S.A.  
Phone: +1 301 794-2195  
Fax: +1 301 794-8355

Panagiotis (Peter) A. Vretanos – [pvretano@cubewerx.com](mailto:pvretano@cubewerx.com)  
CubeWerx Inc.  
200 Rue Montcalm, Suite R-13  
Hull, Quebec, J8Y 3B5, CANADA  
Phone: +1 416 701-1985  
Fax: +1 416 701-9870

---

Introduction.....	3
Preface .....	3
Status of this Document.....	3
Changes to the OpenGIS® Abstract Specification .....	4
Overview.....	4
Terms and Definitions .....	5
Conventions .....	5
Background.....	6
OGC Web Services Profile .....	8
Architecture .....	8
General Model to WWW Profile Message Mapping.....	9
Example Sequence Diagram .....	10
Interface Definition – HTTP .....	10
Overview of Interfaces.....	10
GetCapabilities Operation.....	12
GetRecord Operation .....	18
DescribeRecordType Operation.....	20
Transaction Operation - Informative.....	26
LockRecord Interface –Informative .....	26
RegisterService Operation - Informative .....	26
APPENDIX A – Normative DTD.....	27
GetCapabilities_Request.dtd.....	27
GetCapabilities_Response.dtd .....	27
DescribeRecordType_Request.dtd.....	30
GetRecord_Request.dtd .....	30
GetRecord_Response.dtd.....	30
FeatureId.dtd.....	31
Expr.dtd .....	31
Filter.dtd.....	32
Transaction_Request.dtd.....	33
Transaction_Response.dtd .....	33
LockRecord_Request.dtd.....	33
LockRecord_Response.dtd .....	34
Native.dtd.....	34

## **Introduction**

---

This is the second draft of this discussion paper. The first version of this document described a Web Registry Server. This document is based on the experience gained in the WMT Stateless Catalog prototype efforts and the discussions in the WMT2 telecons and the Catalog Revision Working Group. This is a draft reflects comments made by Catalog Revision Working Group and the Service Model Working Group. The major comments were:

- Integrate this document with the OGC Catalog Services v 1.0 through the refinement of the general model in that specification and the definition of a Catalog Service profile that is consistent with the BSM
- Reuse Web Feature Service concepts and interface definitions when possible

This document refines the OGC Catalog Services general model interface for each operation the OWS profile implements by eliminating parameters that require or support session management. The document then defines the mapping of general model operation and parameter names to profile parameter names. When a parameter is similar to parameter currently defined in the WFS, the WFS parameter name will be used to assist in the homogenizing of the OWS interfaces.

This draft may change significantly based on reviews and the evolution of the Basic/General Services Model.

### ***EDITORS' NOTE:***

*This specification has not been reviewed by the other team members and represents the editor's view of those activities. This is a draft and may change significantly based on reviews and the evolution of the Basic/General Services Model.*

*There many technical issues that need to be discussed and resolved including:*

- *Do others agree with this view of the evolution of the the Stateless catalogs into a Web Registry Service with service description registration and update as a formal interface and the use of GetCapabilities to substitute for the Explain capabilities in the OGC Catalog Specification*
- *Will the same service registry catalog WMS, WFS and WCS. If so how different are the metadata. For example is a Feature Type or a Feature Collection the correct granularity for a registry descriptor and how does it's metadata vary from that of a mapping layer*
- *If we decide that the Registry descriptors need an incremental update service for scalability, can service providers use the WFS transaction interface to specify the required updates*
- *The specification of the Registry descriptor contents and schema are equivalent/dependant on the same research as the Basic/General Service Model. How should the two efforts interact*
- *Is the full GetObject service interface acceptable for lightweight clients or should we maintain a GetObject profile with defaults similar to those of the Stateless Catalog prototypes*

## **Preface**

---

This document describes the OGC Web Registry Server (WRS) interfaces. The WRS interfaces support “one stop shopping” for the registration, metadata harvesting and descriptor ingest, push and pull update of Object, and discovery of OGC Web Service types and instances using HTTP as the distributed computing platform.

### **Status of this Document**

This specification is a draft document submitted to OGC for review and may be updated, replaced or obsoleted by other documents at any time. Considering the early stages of this specification it is inappropriate to use it as reference material or cite this document as other than “work in progress”.

The purpose of this document is to be the basis of an OGC Discussion paper as described in the OGC Policies and Procedures. This document is dependant on the OGC Basic Service Model for the definition of the general semantics, exception handling and the definition of the GetCapabilities operation.

## **Submitting Companies**

The following companies submitted this implementation specification to the OGC as a Request For Comment:

- Compusult
- CubeWerx Inc.
- SGT

## **Submission Contact Points**

All questions regarding this submission should be directed to the editors or the submitters:

### ***Editors***

Lou Reich - Computer Sciences Corp, [louis.i.reich@gsfc.nasa.gov](mailto:louis.i.reich@gsfc.nasa.gov)

Panagiotis (Peter) A. Vretanos – CubeWerx Inc., [pvretano@cubewerx.com](mailto:pvretano@cubewerx.com)

### ***Contributors***

Larry Bourzran

Allan Doyle

Doug Nebert

Barry O'Rourke

George Percival

Ananth Rao

Archie Warnock

## **Document Conventions**

This document contains sections that are “informative”, meaning they serve as explanation and background. Other sections are “normative”, meaning they contain the formal specification against which conformance testing can be done. The normative sections are labeled as such.

In the sections labeled as normative, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [15].

## **Revision History**

- 0.0.1 WMT2 development version, 2000 October 4
- 0.0.2 WMT2 development version, 2001 January 31
- 0.0.3 WMT2 Discussion Paper, 2001 March 14
- 0.0.4 OGC Discussion paper 2001 May 14
- 0.0.5 OGC Discussion paper 2001 July 10 with section 7 of OGC Catalog Services Spec
- 0.0.6 Validate interface DTDs; add transaction DTDs; format a bit.

## **Changes to the OpenGIS® Abstract Specification**

This document may require changes to the Abstract Specification volume other than the additions noted in this document are required for this specification. This specification should form the basis of a new Web Services Volume of the OGC Bookshelf.

## **Overview**

---

### **Motivation**

Phase 2 of the Web Mapping Testbed (WMT2) resulted in several candidate OGC interface specifications for operations on a Web Map Server (WMS), Web Coverage Server (WCS) or Web Feature Server (WFS). The specs all differ in their purposes and details, but a number of elements are common to most or all of them. There was a desire by service users to have a single entry point (or at most a few) and protocol to search for service types of interest. In order to meet this goal, there needs to be an interface that populates a database of service descriptions and an interface to query the description database to discover service locations. This document specifies a Registry Service framework that operates in a stateless HTTP environment. This allows service type discovery using a single catalog site rather than visiting each server and using GetCapabilities operations.

This specification is mapped to the OGC Catalog Service version 1 and is intended to be consistent with the OGC Catalog Service version 1.1. This is intended to allow Web Registries to interoperate with stateful catalogs implementing the OGC Catalog Service in service discovery.

## Terms and Definitions

---

For the purposes of this document, the following terms and definitions apply.

### **Operation**

Specification of an interaction that can be requested from an object to effect behavior (definition from ISO 19119).

### **Interface**

An implementation of operations including the syntax of the interaction for a given distributed computing technology (definition from ISO 19119).

### **Service**

A collection of operations, accessible through an interface, that allows a user to evoke a behavior of value to the user (definition from ISO 19119).

### **Service Instance**

An actual implementation of a Service. Service Instance is synonymous with Server.

### **Client**

A software component that can invoke an Operation from a Server.

### **Request**

An invocation by Client of an Operation.

### **Response**

The result of an Operation returned from a Server to a Client.

### **Capabilities XML**

Service-level metadata describing the operations and content available at a Service Instance.

### ***EDITOR'S NOTE:***

*At the moment, a number of these terms are still being discussed and clarified in the various OGC SIGs. As the definitions of these terms evolves, the editors will need to update the document accordingly.*

## Conventions

---

### **Normative verbs**

In the sections labeled as normative, the key words "must", "must not", "required", "shall", "shall not", "should", "should not", "recommended", "may", and "optional" in this document are to be interpreted as described in Internet RFC 2119 [15].

### **Abbreviated Terms**

CGI	Common Gateway Interface
DCP	Distributed Computing Platform
DTD	Document Type Definition
EPSG	European Petroleum Survey Group
GIF	Graphics Interchange Format
GIS	Geographic Information Systems
HTTP	Hypertext Transfer Protocol

IETF	Internet Engineering Task Force
JPEG	Joint Photographic Experts Group
MIME	Multipurpose Internet Mail Extensions
OGC	Open GIS Consortium
OWS	OGC Web Service
PNG	Portable Network Graphics
RFC	Request for Comments
SLD	Styled Layer Descriptor
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator
WebCGM	Web Computer Graphics Metafile
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
XML	Extensible Markup Language

## Document Organization

This note is divided into two sections:

- The first section which includes the background of the Stateless Catalog Service and is formatted consistently with other OWS discussion papers
- The second section which is formatted as a new section 7 of the OGC Catalog Services Specification which is titled OGC Web Services Profile

## Background

---

Web Mapping Testbed phase 1 (WMT1) resulted in two specifications: "OpenGIS Web Map Server Interface Implementation Specification" version 1, and "OpenGIS Recommendation - Geography Markup Language" (GML) v.1. These documents were the result of a consensus development process by WMT1 sponsors and participants and were approved by the full OGC membership. GML is an XML-based encoding of geographic features, and is described elsewhere; being a file format, GML does not itself specify any operations. The WMS 1.0 specification was an important first step towards interoperable access to georeferenced information, but it does have a number of acknowledged limitations. The goal of Web Mapping Testbed phase 2, and a simultaneous Geospatial Fusion Services (GFS) testbed, was to advance the state of affairs by enhancing the WMS spec and creating new standards for Web Coverage Server, Web Feature Server, GeoParser, GeoCoder, GeoLinker, and others.

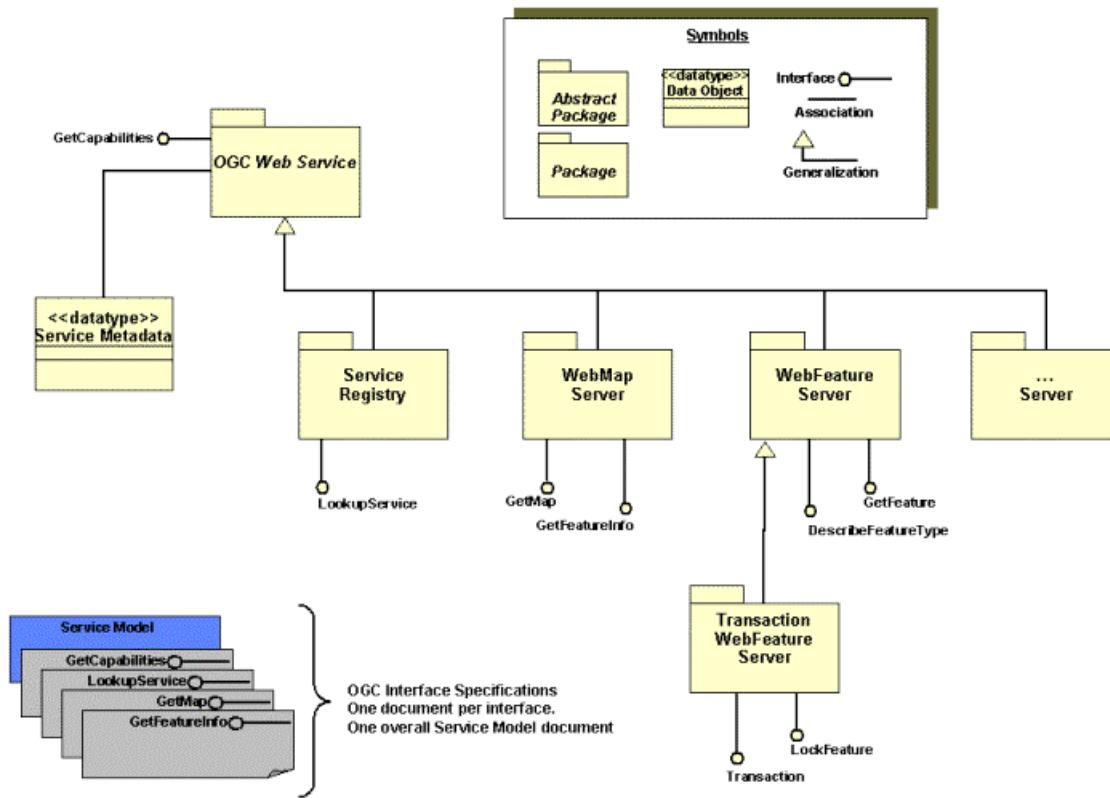
Web Map, Feature, Coverage, and GEOx Servers--collectively called OGC Web Services--are each the subject of detailed specifications. They are described conceptually in the next section as background material.

## Architecture Overview

### *WMT 2 Overview*

Three principal types of georeferenced information access services have been defined by WMT2: Web Map Server (WMS), Web Coverage Server (WCS), and Web Feature Server (WFS). In addition, the Geospatial Fusion Services (GFS) Testbed defined services that return spatially referenced results: GeoParser, GeoCoder, and GeoLinker. Collectively, such services are referred to as OGC Web Services (OWS). Figure 5-1 is an architecture diagram showing conceptually how these components are related, and naming some (not all) of the interfaces between them. The subject of this document and its companion volumes is the interfaces. The internal details of the components are

irrelevant; vendors may build them in any manner that correctly implements the required interfaces.



**Figure 1:WMT2 Architecture emphasizing OWS Catalog Service**

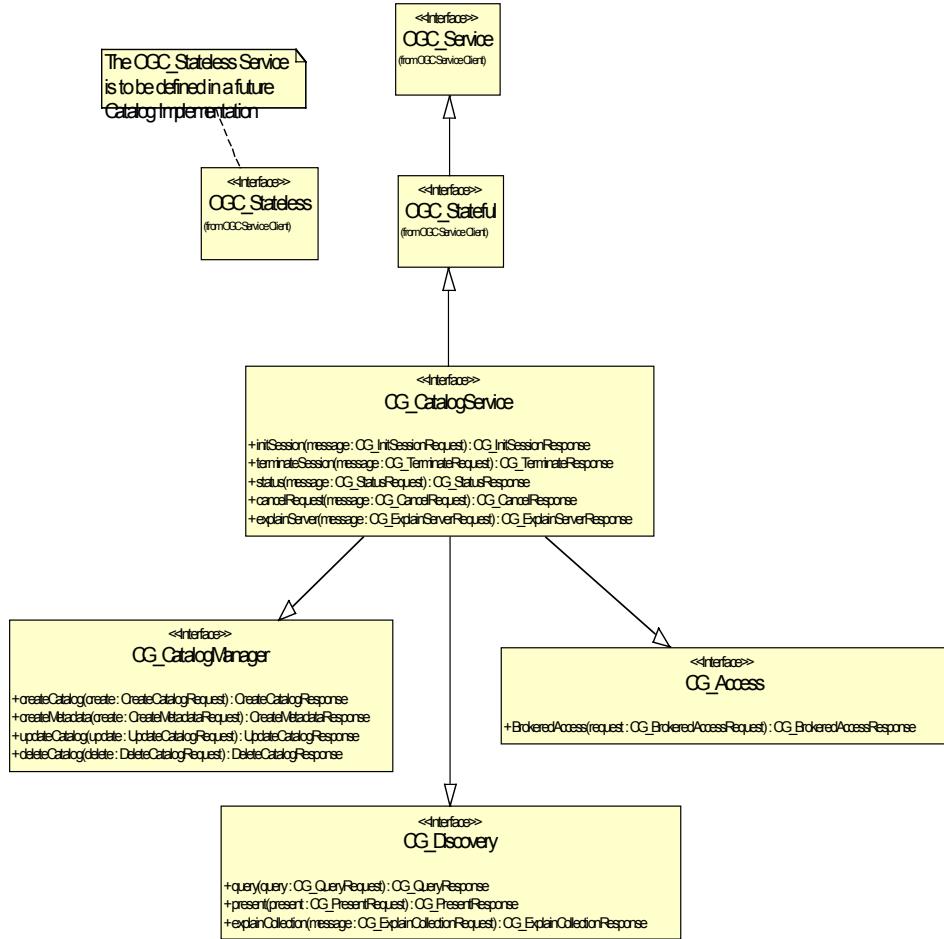
(EDITOR's NOTE: Figure needs to be smoothed).

Each service has several required operations, but it is not required that the same software or internet host computer implement all of them. For example, a WMS may offer GetMap directly but allow its GetCapabilities response to be handled by another machine on its behalf. The service metadata in Capabilities XML includes distinct online resource addresses (e.g., HTTP URLs) for each service offered. There is no requirement in this specification dictating the format or content of those URLs; rather, this document describes only the packaging of the query parameters that must be appended to the service URL.

#### *Overview of OGC Catalog Specification*

Figure 2 shows the general catalog service interfaces. These interfaces allow the discovery, access and management of geospatial data and services. This model is based on the concept of interface operations passing Request – Response Message Pairs between a client and a server. Stated another way, this architecture uses a messaging based structure to describe the access and invocation of Catalog services.

As seen in Figure 2 there are four major interfaces, CG\_CatalogService, CG\_Discovery, CG\_Access and CG\_CatalogManager. These are described in more detail in the following sections of this document. The taxonomy of interfaces that have been placed above the CG\_CatalogService interface (i.e., OGC\_Service and OGC\_Stateful) have been created to put forth the idea of having an overall architectural framework for the different services that will be developed over time to populate the OGC Service Architecture.



This specification defines stateless versions of the CG\_CatalogService, the CG\_Discovery and the CG\_CatalogManager interfaces which are intended to support lightweight clients and servers that support HTTP version 1.0 type functionality and do not necessarily maintain information required to enable persistent sessions.

## OGC Web Services Profile

---

### ***EDITOR's NOTE:***

*Known updates after decisions by WMT3 and CRG before 8/15*

- Add XMLSchema /WSDL interface versions
- Add or point to brief, summary and full DTDs
- Add example of MetadataURL using DEDSL
- The GetCapabilitieUpdate DTDs using Entity notation for extensible choice lists
- The GetCapabilities DTD needs to be improved
- Add more examples

### Architecture

The OWS Profile uses a message-based client server architecture. This profile implements the stateless versions of the CG\_CatalogService, the CG\_Discovery and the CG\_CatalogManager interfaces which are

intended to support lightweight clients and servers that support HTTP version 1.0 type functionality and do not necessarily maintain information required to enable persistent sessions. The profile maps each of the general model operations to a corresponding OGC Web Feature Service.

The OWS Profile specifies the use of the following transport mechanisms:

- With the GET method, a request is encoded into the URI that is sent to the web server.
- A POST request can be arbitrarily long and complex which means that client applications can generate arbitrarily long and complex transaction or query requests.

HTTP defines two methods of exchanging information with a web server; the **GET** method and the **POST** method. The OGC Web Services profile requests, encoded in XML or keyword-value pairs, may be transmitted to a Stateless Catalog Server using either the POST or GET methods. However, the XML encoding in the following sections is most suitable for packaging as an HTTP POST request. A keyword-value pair encoding is more suitable for packaging as an HTTP GET request.

### **GET Method**

An online resource URL intended for HTTP GET requests is in fact only a URL prefix to which additional parameters must be appended in order to construct a valid operation request. A URL prefix is defined as an opaque string including the protocol, hostname, optional port number, path, a question mark '?', and, **optionally**, one or more server-specific parameters delimited by an ampersand '&'. The prefix uniquely identifies the particular service instance. A client appends the necessary request parameters as name/value pairs in the form "name=value". The resulting URL **must** be valid according to the HTTP Common Gateway Interface (CGI) standard [17], which mandates the presence of '?' before the sequence of query parameters and the '&' between each parameter. As with all CGI applications, the query URL is encoded [12] to protect special characters.

The URL prefix **must** end in either a '?' (in the absence of additional server-specific parameters) or a '&'. In practice, however, Clients **should** be prepared to add a necessary trailing '?' or '&' before appending the Operation parameters defined in this specification in order to construct a valid request URL.

Table 1 summarizes the components of an operation request URL.

**Table 1 — A general OGC Web Service Request**

<b>URL Component</b>	<b>Description</b>
<code>http://host[:port]/path?[param[=value]&amp;]</code>	URL prefix of service operation. [ ] denotes an optional part. This portion is entirely at the discretion of the service provider.
<code>name=value&amp;</code>	One or more standard request parameter name/value pairs defined by an OGC Web Service. The actual list of required and optional parameters is mandated for each operation by the appropriate OWS specification.

With the GET method, a request is encoded into the URI that is sent to the web server. The main limitations with this method are:

1. The length of the URL may have a hard limit (typically set by the browser being used) which limits the size of a request.
2. It is difficult (if not impossible) for complicated MIME messages to be sent from the client to the server using the GET method.

### **POST Method**

An Online Resource URL intended for HTTP POST requests is a complete and valid URL to which clients transmit request parameters in the body of the POST request. A WMS **must not** require additional parameters to be appended to the URL in order to construct a valid target for the Operation request. A POST request can be arbitrarily long and complex which means that client applications can generate arbitrarily long and complex transaction or query requests.

### **General Model to WWW Profile Message Mapping**

Table 11 provides a mapping between general model operations, the OWS Profile and the WWW Profile services.

**Table 11 - General Model to WWW Profile/OGC Web Services Message Mapping**

<b>General Model Operation</b>	<b>OGC Web Services Profile</b>
CG_InitSessionRequest	N/A <sup>1</sup>
CG_InitSessionResponse	N/A <sup>9</sup>
CG_TerminateRequest	N/A
CG_TerminateResponse	N/A
CG_ExplainServerRequest	GetCapabilities Request
CG_ExplainServerResponse	GetCapabilities Response
CG_StatusRequest	N/A
CG_StatusResponse	N/A
CG_CancelRequest	N/A
CG_CancelResponse	N/A
CG_QueryRequest	GetRecords Request
CG_QueryResponse	GetRecords Response
CG_PresentRequest	GetRecords Request
CG_PresentResponse	GetRecords Response
CG_ExplainCollectionRequest	DescribeRecordType Request
CG_ExplainCollectionResponse	DescribeRecordType Response
CG_BrokeredAccessRequest	N/A
CG_BrokeredAccessResponse	N/A

### **Example Sequence Diagram**

TBS

### **Interface Definition – HTTP**

#### *Overview of Interfaces*

Figure 6-1 depicts a conceptual architecture to illustrate the relationship of these interfaces to service consumers and producers.

---

<sup>1</sup>

<sup>9</sup>

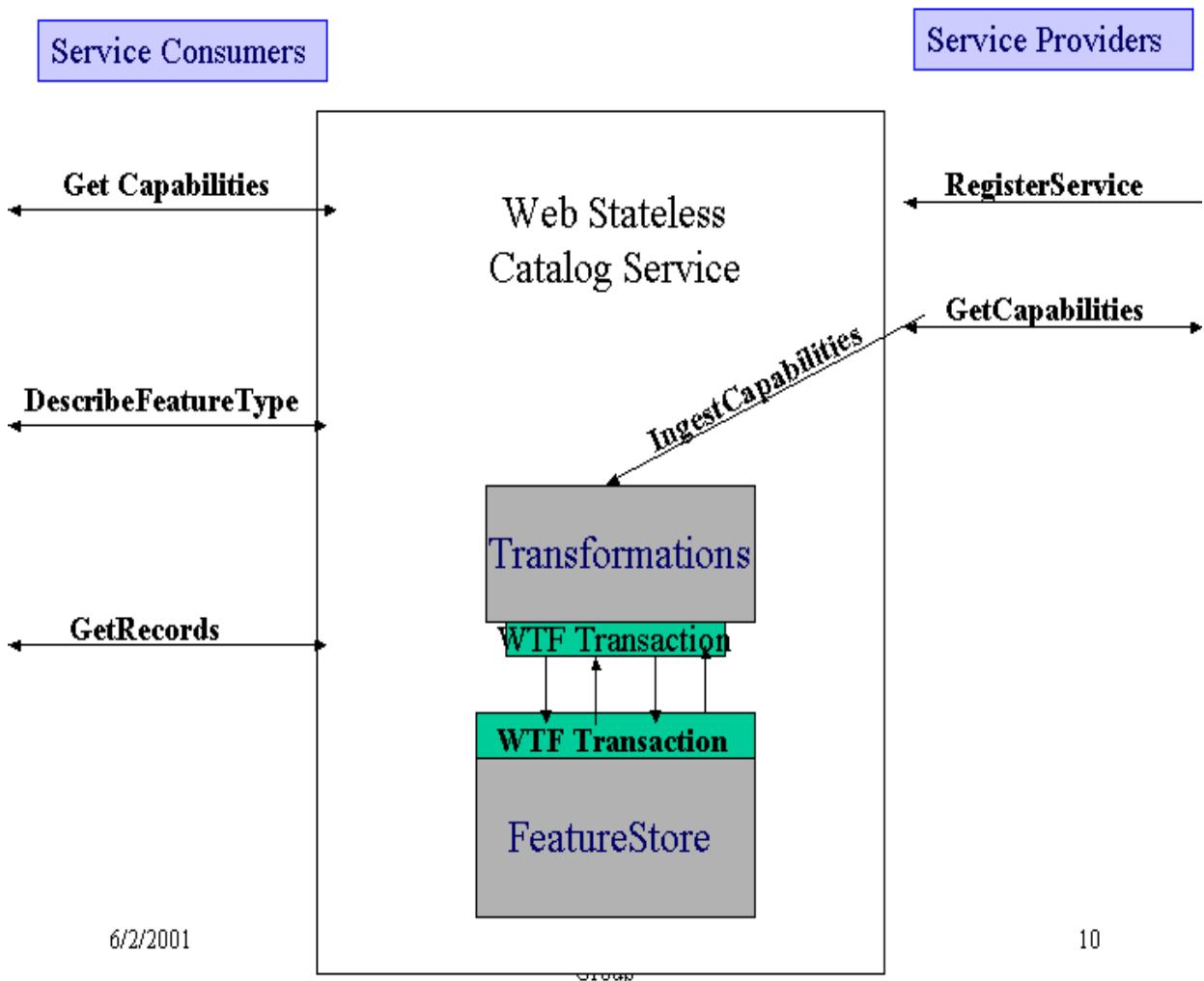


Figure 6-1: Conceptual Architecture

**To support service discovery the Stateless Catalog Service must support the following interfaces:**

**<GetCapabilities>**

A StSC must be able to describe its capabilities. Specifically, it must indicate which query languages, namespaces, return schema and return formats it supports. This service is an implementation of the non-session oriented portions of the CG\_CatalogService Interface in the OGC Catalog Service Specification

**<GetRecord>**

A registry server must be able to respond to client queries in at least one of a number of predefined query languages and return the results of those queries in at least one of a number of predefined formats. This interface is an implementation of the CG\_QueryRequest operation in the CG\_Discovery interface in OGC Catalog Service Specification

**To better support adaptable clients in service discovery the StSC may support the following interfaces:**

**<DescribeRecordType>**

The function of the DescribeRecordType interface is to provide a client the means to request a schema definition of any feature type that a particular StSC can service. This interface is an implementation of the ExplainCollection operation in the CG\_Discovery interface in the OGC Catalog Service Specification.

**EDITORS NOTE:**

*The following interfaces have not been included as normative in this version of the OWS Profile due to the need for discussion, design and prototyping.*

**To support the management of stored description records, a StSC may implement the Transaction and LockFeature interfaces of the OGC WFS.**

These interfaces may be modified in later versions of this document but at this point they are identical to the interfaces specified in the OGC . These interfaces are an implementation of the CG\_CatalogManager Interface

**<Transaction>**

A WMT StSC may implement the **Transaction** interface to describe data transformation operations that are to be applied to catalog records. The web feature server receives a transaction request and either processes it directly or possibly translates it into the language of a target datastore and then has the datastore execute the transaction. When the transaction has been completed, the web feature server will generate an XML response document indicating the termination status of the transaction

**<LockFeature>**

If a WMT StSC implements the Transaction interface it must implement the LockFeature operation. The purpose of the **LockFeature** interface is to expose a *long term object locking* mechanism to ensure consistency. The lock is considered long term because network latency would make feature locks last relatively longer than native commercial database locks.

**To Open Server Registration and Periodic Content Harvesting the OWS Catalog Service may support the following interfaces:**

**<RegisterService>**

OWS Catalog Service may allow service provider's to register their servers online, request periodic update thru the harvesting of capabilities documents from the server or the adhoc update of the description records database by transaction or in bulk

**EDITOR'S NOTE:**

*The <RegisterService> operation may not be required since the mere act of inserting a record into the catalog using the WFS <Transaction><Insert> operation would register the service.*

***GetCapabilities Operation***

**Request DTD**

The <GetCapabilities> element is used to request a capabilities document from a web feature server. It is defined by the following DTD:

```
<! -- **** * -->
<! -- * REQUEST: GETCAPABILITIES * -->
<! -- **** * -->

<!ELEMENT GetCapabilities EMPTY>
<!ATTLIST GetCapabilities version CDATA #Implied
    updateSequence CDATA #Implied>
```

**Keyword=Value Pair Encoding**

The general form of a GetCapabilities request is defined in the Basic Service Elements section. When making this request of a WMS, which may offer other OGC Web Services as well, it is necessary to

indicate that the client seeks information about the WMS in particular. Thus, the SERVICE parameter of the request **must** have the value "WMS" as shown in Table 3 below.

**Table 3 — The parameters of a GetCapabilities request URL**

Request Parameter	Required/ Optional	Description
VERSION=version	O	Request version
SERVICE=WMS	R	Service type
REQUEST=GetCapabilities	R	Request name
UPDATESEQUENCE=number	O	Sequence number for cache control

#### ***VERSION=version***

This parameter, and its use in version negotiation, is specified in the Basic Service Elements section. In WMS version 1.0.0, the name of this parameter was "WMTVER". That name is now deprecated, but for backwards compatibility and version negotiation a post-1.0.0 server **should** accept either form without issuing a Service Exception.

#### ***SERVICE=service\_name***

SERVICE indicates which of the available service types at a particular service instance is being invoked. This parameter allows the same URL prefix to offer Capabilities XML for multiple OGC Web Services. When invoking GetCapabilities on a Stateless Catalog that implements this version of the specification or a later one, the service\_name value "StSC" **must** be used.

#### ***REQUEST=GetCapabilities***

This nature of this parameter is specified in the Basic Service Elements section. To invoke the GetCapabilities operation, the value "GetCapabilities" is used. In WMS version 1.0.0, the value of this parameter was "capabilities". That value is now deprecated, but for backwards compatibility a post-1.0.0 server **must** accept either form without issuing a Service Exception. When a client is initially contacting a WMS whose version it does not know the Client **should** be prepared to recover if REQUEST=GetCapabilities fails and **may** send REQUEST=capabilities.

#### ***UPDATESEQUENCE=number***

UpdateSequence is an optional parameter for maintaining cache consistency. It is envisaged that this number will be a timestamp. The server **may** include an UpdateSequence number in its Capabilities XML. If present, this number **should** be increased when changes are made to the Capabilities (e.g., when new maps are added to the service). The client **may** include this parameter in its GetCapabilities request. The response of the server based on the presence and relative value of UpdateSequence in the client request and the server metadata **must** be according to Table 4:

**Table 4 — Use of UpdateSequence Parameter**

Client Request UpdateSequence Value	Server Metadata UpdateSequence Value	Server Response
None	any	most recent Capabilities XML
Any	none	most recent Capabilities XML
Equal	equal	Exception: code=CurrentUpdateSequence
Lower	higher	most recent Capabilities XML
Higher	lower	Exception: code=InvalidUpdateSequence

## Response DTD

```
<!ELEMENT STSC_Capabilities (Service, Capability)>
<!ATTLIST STSC_Capabilities
      version CDATA #FIXED "0.0.1"
      updateSequence CDATA "0">

<!-- The SCHEMALANGUAGES entity defines the
     schema languages that a feature server is
     capable of using to describe the schema of
     a feature. This entity can be redefined
     by individual servers to include other
     schema languages but XMLSCHEMA must always
     be defined. -->
<!ENTITY % SCHEMALANGUAGES "XMLSCHEMA">
<!ELEMENT XMLSCHEMA EMPTY>

<!-- The RESULTFORMATS entity defines the
     output formats that the web feature server
     can generate. The mandatory format "GML2"
     must always be available. This entity can
     be redefined by individual servers to
     include other formats. -->
<!ENTITY % RESULTFORMATS " XML">
<!ELEMENT XML EMPTY>

<!-- Elements used in multiple places. -->
<!-- The Name is typically for machine-to-machine communication. -->
<!ELEMENT Name (#PCDATA)>

<!-- The Title is for informative display to a human. -->
<!ELEMENT Title (#PCDATA)>

<!-- The abstract is a longer narrative description of an object. -->
<!ELEMENT Abstract (#PCDATA)>

<!-- An OnlineResource is typically an HTTP URL. The URL is placed
     in the xlink:href attribute. The xmlns:xlink attribute is a
     required XML namespace declaration. -->
<!ELEMENT OnlineResource EMPTY>
<!ATTLIST OnlineResource
      xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
      xlink:type CDATA #FIXED "simple"
      xlink:href CDATA #REQUIRED>

<!-- A container for listing an available format's MIME type. -->
<!ELEMENT Format (#PCDATA)>

<!-- General service metadata. -->
<!ELEMENT Service (Name, Title, Abstract?, KeywordList?, OnlineResource,
ContactInformation?, Fees?, AccessConstraints?)>

<!-- List of keywords or keyword phrases to help catalog searching. -->
<!ELEMENT KeywordList (Keyword*)>

<!-- A single keyword or phrase. -->
<!ELEMENT Keyword (#PCDATA)>

<!-- Information about a contact person for the service. -->
<!ELEMENT ContactInformation (ContactPersonPrimary?, ContactPosition?, ContactAddress?,
ContactVoiceTelephone?, ContactFacsimileTelephone?, ContactElectronicEmailAddress?)>

<!--The primary contact person.-->
<!ELEMENT ContactPersonPrimary (ContactPerson, ContactOrganization)>

<!--The person to contact.-->
<!ELEMENT ContactPerson (#PCDATA)>

<!--The organization supplying the service.-->
<!ELEMENT ContactOrganization (#PCDATA)>
```

```

<!--The position title for the contact person.-->
<!ELEMENT ContactPosition (#PCDATA)>

<!--The address for the contact supplying the service.-->
<!ELEMENT ContactAddress (AddressType, Address, City, StateOrProvince, PostCode,
Country)>

<!--The type of address.-->
<!ELEMENT AddressType (#PCDATA)>
<!--The street address.-->
<!ELEMENT Address (#PCDATA)>
<!--The address city.-->
<!ELEMENT City (#PCDATA)>
<!--The state or province.-->
<!ELEMENT StateOrProvince (#PCDATA)>
<!--The zip or postal code.-->
<!ELEMENT PostCode (#PCDATA)>
<!--The address country.-->
<!ELEMENT Country (#PCDATA)>
<!--Contact telephone number.-->
<!ELEMENT ContactVoiceTelephone (#PCDATA)>
<!--The contact fax number.-->
<!ELEMENT ContactFacsimileTelephone (#PCDATA)>
<!--The e-mail address for the contact.-->
<!ELEMENT ContactElectronicMailAddress (#PCDATA)>

<!-- Elements indicating what fees or access constraints are imposed. -->
<!ELEMENT Fees (#PCDATA)>
<!ELEMENT AccessConstraints (#PCDATA)>

<!-- A Capability lists available request
      types, how exceptions may be reported, and
      whether any vendor-specific capabilities
      are defined. It also lists all the
      feature types available from this feature
      server. -->
<!ELEMENT Capability (Request, VendorSpecificCapabilities?, Exceptions, PresentOptions,
RecordTypeList)>

<!ELEMENT Request (GetCapabilities | DescribeRecordType | GetRecord | LockRecord |
Transaction | RegisterService)+>
<!ELEMENT GetCapabilities (DCPType+)>
<!ELEMENT DescribeRecordType (SchemaDescriptionLanguage, DCPTYPE+)>
<!ELEMENT SchemaDescriptionLanguage (%SCHEMALANGUAGES;)+>
<!ELEMENT GetRecord (ResultFormat, DCPTYPE+)>
<!ELEMENT ResultFormat (%RESULTFORMATS;+)>
<!ELEMENT LockRecord (DCPType+)>
<!ELEMENT Transaction (DCPType+)>
<!ELEMENT RegisterService (DCPType+)>

<!-- Available Distributed Computing Platforms
      (DCPs) are listed here. At present, only
      HTTP is defined. -->
<!ELEMENT DCPTYPE (HTTP)>
<!-- Available HTTP request methods. -->
<!ELEMENT HTTP (Get | Post)+>
<!-- HTTP request methods. The onlineResource
      attribute indicates the URL prefix for
      HTTP GET requests (everything before the
      question mark and query string:
      http://hostname[:port]/path/scriptname);
      for HTTP POST requests, onlineResource is
      the complete URL. -->
<!ELEMENT Get EMPTY>
<!ATTLIST Get onlineResource CDATA #REQUIRED >
<!ELEMENT Post EMPTY>
<!ATTLIST Post onlineResource CDATA #REQUIRED >
<!-- The optional VendorSpecificCapabilities
      element lists any capabilities
      unique to a particular server. Because

```

the information is not known a priori, it cannot be constrained by this particular DTD. A vendor-specific DTD fragment must be supplied at the start of the XML capabilities document, after the reference to the general STSC\_Capabilities DTD.

```
-->
<!-- DEFINE THIS ELEMENT AS NEEDED IN YOUR XML
    <!ELEMENT VendorSpecificCapabilities (your stuff here)>
-->
<!ELEMENT VendorSpecificCapabilities EMPTY>
<!-- An Exception element indicates which error-reporting
    formats are supported. -->
<!ELEMENT Exceptions (Format+)>
<!--Presentation Options Supported by Server -->
<!ELEMENT PresentOptions EMPTY>
<!ATTLIST PresentOptions
    StartRec (0 | 1) "0"
    Hits (0 | 1) "1"
    RecsMax CDATA #IMPLIED>
<!-- A list of feature types available from
    this server. The following table
    specified the number and source of the
    various elements that are available for
    describing a feature type.

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

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

    Abstract      0/1   optional; no Default

    Keywords      0/1   optional; no Default

    SRS           0/1   the SRS that should be used
                      when specifying the state of
                      the feature

    LatLonBoundingBox 0/1 exactly one is required;
                      default from parent
    Operations     1+    allowable operations for this record type

    MetadataURL   0+    optional; no default
-->
<!ELEMENT RecordTypeList (RecordType+)>
<!ELEMENT RecordType (Name, Title?, Abstract?, Keywords?, SRS, Operations?, LatLonBoundingBox?, MetadataURL*)>
<!ELEMENT Keywords (#PCDATA)>
<!ELEMENT SRS (#PCDATA)>

<!ELEMENT Operations (Insert | Update | Delete | Present | Query | Lock)+>
<!ELEMENT Insert EMPTY >
<!ELEMENT Update EMPTY>
<!ELEMENT Delete EMPTY >
<!ELEMENT Query EMPTY >
<!ELEMENT Lock EMPTY>
<!ELEMENT Present EMPTY>

<!-- The LatLonBoundingBox attributes indicate
    the edges of the enclosing rectangle in
    latitude/longitude decimal degrees (as in
    SRS EPSG:4326 [WGS1984 lat/lon]).


LatLonBoundingBox MUST be supplied regardless of what SRS the server may support, but it MAY be approximate if EPSG:4326 is not supported. Its purpose


```

```

is to facilitate geographic searches
without requiring coordinate
transformations by the search engine. -->
<!ELEMENT LatLonBoundingBox EMPTY>
<!ATTLIST LatLonBoundingBox
      minx CDATA #REQUIRED
      miny CDATA #REQUIRED
      maxx CDATA #REQUIRED
      maxy CDATA #REQUIRED>
<!-- A CatalogServer MAY use zero or more
MetadataURL elements to offer
detailed, standardized metadata about the
data underneath a particular record type.
The type attribute indicates the standard
to which the metadata complies; the format
attribute indicates how the metadata is
structured. Four types are defined at present:
'TC211' = ISO TC211 19115;
'FGDC' = FGDC CSDGM.
'DEDSL' = CCSDS Data Entity Dictionary Specification Language
'ISO11179' = ISO 11179 part3 Data Element definitions
-->
<!ELEMENT MetadataURL (#PCDATA)>
<!ATTLIST MetadataURL
      type (TC211 | FGDC | DEDSL | ISO11179) #IMPLIED
      format (XML | SGML | TXT) #REQUIRED>

```

### ***Example of returned Capabilities DTD***

```

<?xml version="1.0" encoding="UTF-8"?>
<StSC_Capabilities version="0.4">
  <Service>
    <Name>StatelessCatalogServer</Name>
    <Title>Lou's Stateless Catalog Server</Title>
    <Abstract>
      Web Feature Server maintained by Lou.
    </Abstract>
    <OnlineResource>
      http://tux.lou.com/StSC/cwStSC.cgi?
    </OnlineResource>
  </Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <DCPType>
          <HTTP>
            <Get onlineResource="http://tux.lou.com/StSC/cwStSC.cgi?" />
          </HTTP>
        </DCPType>
        <DCPType>
          <HTTP>
            <Post onlineResource="http://tux.lou.com/StSC/cwStSC.cgi" />
          </HTTP>
        </DCPType>
      </GetCapabilities>
      <DescribeRecordType>
        <SchemaDescriptionLanguage>
          <XMLSCHEMA/>
        </SchemaDescriptionLanguage>
        <DCPType>
          <HTTP>
            <Get onlineResource="http://tux.lou.com/StSC/cwStSC.cgi?" />
          </HTTP>
        </DCPType>
        <DCPType>
          <HTTP>
            <Post onlineResource="http://tux.lou.com/StSC/cwStSC.cgi" />
          </HTTP>
        </DCPType>
      </DescribeRecordType>
    </Request>
  </Capability>
</StSC_Capabilities>

```

```

<GetRecord>
  <ResultFormat>
    <XML/>
  </ResultFormat>
  <DCPType>
    <HTTP>
      <Get onlineResource="http://tux.lou.com/StSC/cwStSC.cgi?"/>
    </HTTP>
  </DCPType>
  <DCPType>
    <HTTP>
      <Post onlineResource="http://tux.lou.com/StSC/cwStSC.cgi"/>
    </HTTP>
  </DCPType>
</GetRecord>
</Request>
<PresentOptions recmax="200"/>
<RecordTypeList>
  <Operations>
    <Query/>
    <Present/>
    <StartRec/>
    <Lock/>
  </Operations>
  <RecordType>
    <Name>Service</Name>
    <Title>Queryable attributes for service records<Title>
    <Operation>Query<Operation>
    <MetadataURL>http://ogc.org/catrec/service.html<MetadataURL>
  </RecordType>
  <RecordType>
    <Name>Product</Name>
    <Title>Queryable attributes for data products records<Title>
    <Operation>Query<Operation>
    <MetadataURL>http://ogc.org/catrec/product.html<MetadataURL>
  </RecordType>
  <RecordType>
    <Name>Collection</Name>
    <Title>Queryable attributes for data collection records<Title>
    <Operation>Query<Operation>
    <MetadataURL>http://ogc.org/catrec/collection.html<MetadataURL>
  </RecordType>
  <RecordType>
    <Name>ISO11919</Name>
    <Title>Metadata schema for service records<Title>
    <Operation>Present<Operation>
    <MetadataURL>http://ogc.org/catrec/11919.html<MetadataURL>
  </RecordType>
</RecordTypeList>
</STSC_Capabilities>

```

### **GetRecord Operation**

#### **Request in DTD**

A query request is described using the <GetRecord> element. The following DTD fragment defines the XML encoding of a GetRecord request:

```

<!ENTITY % FILTERDTD SYSTEM "Filter.dtd">
%FILTERDTD;

<!-- **** REQUEST: GETOBJECT * -->
<!-- * ELEMENT GetRecord (Query+, Filter?)>
<!ATTLIST GetRecord
  handle CDATA #IMPLIED
  maxRecords CDATA #IMPLIED
  startPosition CDATA #IMPLIED
  outputFormat (XML) #REQUIRED
  outputRecType (%METASTANDARDS;) #REQUIRED>

```

```

<!ELEMENT Query ((PropertySet | PropertyName*)?, Filter?)>
<!ATTLIST Query handle CDATA #IMPLIED
          typeName (Service | Product | Collection) #REQUIRED>
<!ELEMENT PropertySet EMPTY>
<!ATTLIST PropertySet
          setName (Brief | Summary | Full | Hits) "Brief">

```

### Keyword = Value Pair Encoding

	Required/ Optional	Description
http://server_address/path/script?	R	URL prefix of server.
<b>VERSION=version</b>	R	Request version.
<b>REQUEST=GetRecord</b>	R	Request name.
<b>queryLanguage</b>	R	Query Language of Query Expression to follow current allowed values are OGC Common, OGC Filter, SFSQL. . Must be in list of queey languages in the response to GetCapabilities from this Service.
<b>typeName</b>	R	Url of record type of attribute names used in Query expression. Current allowed values are One of Collection, Product or Service Must be in list of attribute namespaces list in the response to GetCapabilities
<b>querySpec</b>	R	String expressing constraints used to select Object to return
<b>maxRecs</b>	O	Maximum number of Records to be returned. If not specified returns all matching Object or service maximum defined in GetCapabilities response
<b>outputRecType</b>	O	Schema used for result sets. Must be in list of element schemas list in the response to GetCapabilities. Default listed in GetCapabilite response
<b>SetName</b>	O	Element set name to determine which descriptor fields should be returned. allowed values are Brief, Full, and Summary. Must be in list of quey languages in the response to GetCapabilities from this Service. Default is Brief .
<b>StartPosition</b>	O	The startPosition parameter identifies the first result set entry to be returned specified the default is the first record
<b>SortKey</b>	O	The sortField parameter specifies how the Object data is to be sorted prior to presentation.
<b>ReturnFormat</b>	O	The returnFormat parameter specifies the encoding standard to be used for the result set. Currently the only allowed value is XML

### Example

```

http://www.compusalt.com/stsc.cgi&
VERSION=0.0.4&
REQUEST=GETRECORD&
QUERYLANGUAGE=OGC_COMMON&
TYPENAME=SERVICE&
OUTPUTRECTYPE=ISO19115&
SETNAME=SUMMARY&
QUERYSPEC= SRS='EPSG:4326'
          AND title like '%interstate%'
          AND (Format='gif' OR format='jpeg')
          AND ServiceType='Web Mapping Layer'
AND Intersects(LatLongBoundingBox,Envelope(-160,-50,50,10))

```

### GetRecord Response

The GetRecord response is an XML document that contains information on the success of the query and the description objects at one of three levels of details. The DTD, which appears below, defines those common elements giving background and status of the query. The three DTDs in annex B demonstrate the differing levels of detail in the brief summary and full element sets

Currently the only available element set are based on the FGDC and ISO TC211 metadata standards

```

<!-- **** * ***** * ***** * ***** * ***** * -->
<!-- *   RESPONSE: GETOBJECT               * -->
<!-- **** * ***** * ***** * ***** * -->

<!ENTITY % DistinguishedName "(nameValue,nameNameSpace)">
<!ENTITY % CI_OnLineResource "(linkage,
                               protocol?,
                               applicationProfile?,
                               onlineResourceName?,
                               onlineResourceDescription?,
                               functionCode?)">
<!-- ===== -->
<!-- The top level holder for a response from the -->
<!-- catalog.                                -->
<!-- ===== -->
<!ELEMENT searchResponse (searchParameter*, searchStatus, searchResults)>
<!ATTLIST searchResponse
          DTD_Version CDATA #FIXED "1.1.0"
>
<!ELEMENT searchParameter (#PCDATA)>
<!ATTLIST searchParameter
          name CDATA #REQUIRED>
<!-- ===== -->
<!-- This tag contains a number of attributes that detail -->
<!-- the status of the search                   -->
<!-- ===== -->
<!ELEMENT searchStatus EMPTY>
<!-- ===== -->
<!-- elementSetName - The element set that has been      -->
<!--                           returned (e.g., brief, summary,    -->
<!--                           full)                         -->
<!-- numberOfRecords - The number of hits returned       -->
<!--                           in this result.           -->
<!-- schema        - The type of response returned (e.g.    -->
<!--                           OGCService, FGDC)           -->
<!-- success       - Was the search successful (true,       -->
<!--                           false)                      -->
<!-- timestamp     - The date and time at the completion    -->
<!--                           of the search            -->
<!-- ===== -->
<!ATTLIST searchStatus
          elementSetName CDATA #FIXED "brief"
          success CDATA "true"
          numberOfRecords CDATA #IMPLIED
          schema CDATA #FIXED "ISO19119"
          timestamp CDATA #IMPLIED>
<!-- The holder for the results from the catalog. Although
     defined as EMPTY here, at actual server would insert
     the root element(s) of the metadata schemas in which
     it can present results.
-->
<!ELEMENT searchResults EMPTY>
```

### ***DescribeRecordType Operation***

#### **DTD Request**

A **DescribeRecordType** request is composed of zero or more names of feature types that are to be described. If the content of the request is empty then that shall be interpreted as requesting a description of all Record types that a WFS can service. The XML encoding of a **DescribeRecordType** request is defined by the following DTD:

```

<!-- **** * ***** * ***** * ***** * ***** * -->
<!-- *   REQUEST: DESCRIBE OBJECT             * -->
<!-- **** * ***** * ***** * ***** * -->
<!ENTITY % SCHEMALANGUAGES "(XMLSCHHEMA)">
<!ELEMENT DescribeRecordType (TypeName*)>
<!ATTLIST DescribeRecordType
```

```

    outputFormat (%SCHEMALANGUAGES;) #IMPLIED>
<!ELEMENT TypeName (#PCDATA)>
```

The **outputFormat** attribute is used to indicate the schema description language that should be used to describe a Record schema. The only mandated format is XML-Schema denoted by the **XMLSCHEMA** element; other vendor specific formats specified in the capabilities document are also possible.

### Keyword=Value Pair Encoding

URL Component	Required/ Optional	Description
Http:/server_address/path/script	R	URL prefix of web feature server.
VERSION=0.0.13	R	Request version..
REQUEST=DESCRIBEFEATURETYPE	R	Name of request.
TYPENAME=feature_type_list	R	A comma separated list of feature types to describe.
		.
OUTPUTFORMAT=SCHEMALANGUAGE	O	Name of Schema language from Capabilities document

### DescribeRecordType Response

In response to a **DescribeRecord Type** request, where the output format has been specified as **XMLSCHEMA**, a StSC will dynamically generate a feature schema using XML-Schema

In response to a **DescribeRecord Type** request, where the output format has been specified as **DTD**, an StSC will dynamically generate a feature schema using DTDs

### DescribeRecordType Response in Text

In response to a DescribeRecordType for the *service* type, a text file can be generated instead of an XML file. The following fields can be included as search terms in the query:

```

LatLonBoundingBox:
    Intersects(LatLonBoundingBox, Envelope(westbound, eastbound,
                                              northbound, southbound))

Keyword:
    Keyword like '%contains%'

Title:
    Title like 'begins_with%'

Abstract:
    Abstract like '%ends_with%'

Format:
    Format = 'GIF'

DCPType:
    DCPType = 'Type'

SRS:
    SRS = 'EPSG:4326'

ServiceType:
    ServiceType = 'Web Mapping Service'
```

The 'like' operator is not case sensitive, the '=' operator is. Multiple fields can be searched simultaneously, with terms joined either by AND or OR. For example:

```
ServiceType = 'Web Mapping Layer'
AND Intersects(LatLonBoundingBox, Envelope(-160,-50, 50, 10))
AND Keyword like '%Water%'
AND (SRS = 'EPSG:4236' OR SRS = 'EPSG:6473')
AND (Format = 'GIF' OR Format = 'PNG')
```

### **DescribeRecordType Response as DTD**

```
<!--The top level tag for a result created with the ISO 19119 schema.-->
<!ELEMENT ISO19119 (citation,abstract,purpose?,credit?,statusCode*,pointOfContact*,
                    resourceSpecifiedUsage*,serviceTypeVersion,serviceType,typeProperty*,
                    accessProperties,legalConstraints*,securityConstraints*,quality?,
                    keywords*,operationMetadata*) >

<!--relevanceRank - The relative value of this hit, higher = better
serviceId - The unique identifier for this service. Can be used to later retrieve
additional information.
timestamp - The catalog date for this service record.-->
<!ATTLIST ISO19119
            relevanceRank CDATA    " "
            timestamp   CDATA    #IMPLIED
            serviceId   CDATA    #REQUIRED >

<!ELEMENT citation %CI_Citation; >

<!ELEMENT abstract (#PCDATA) >

<!ELEMENT purpose (#PCDATA) >

<!ELEMENT credit (#PCDATA) >

<!ELEMENT statusCode EMPTY >

<!ATTLIST statusCode
            progressCode (completed | historicalArchive | obsolete | onGoing |
            planned | required | underdevelopment) #REQUIRED >

<!ELEMENT pointOfContact %CI_ResponsibileParty; >

<!ELEMENT resourceSpecifiedUsage (specifiedUsage,usageDateTime?,
                                userDetirminedLimitations?,userContactInfo+) >

<!ELEMENT serviceTypeVersion (#PCDATA) >

<!ELEMENT serviceType %DistinguishedName; >

<!ELEMENT typeProperty (typeName,typeValue) >

<!ELEMENT accessProperties (fees?,plannedAvailableDateTime?,orderingInstructions?,
                           turnaround?) >

<!ELEMENT legalConstraints (useLimitation*,propertyRightsCode*,useConstraintsCode*,
                           otherConstraints*) >

<!ELEMENT securityConstraints (useLimitation*,classificationCode,userNote?,
                               classificationSystem?,handlingDescription?) >

<!ELEMENT quality (TBD_ServiceQuality) >

<!ELEMENT keywords (keyword*,typeCode?,thesaurusName?) >

<!ELEMENT operationMetadata (operationName,operationDescription,parameter*,
                           dependsOn?,DCP+) >

<!ELEMENT specifiedUsage (#PCDATA) >
```

```

<!ELEMENT usageDateTime (#PCDATA) >
<!ELEMENT userDeterminedLimitations (#PCDATA) >
<!ELEMENT userContactInfo (individualName*,organisationName*,positionName*,
                           contactInfo*,roleCode+) >
<!ELEMENT typeName %DistinguishedName; >
<!ELEMENT typeValue %ValueType; >
<!ELEMENT fees (#PCDATA) >
<!ELEMENT plannedAvailableDateTime (#PCDATA) >
<!ELEMENT orderingInstructions (#PCDATA) >
<!ELEMENT turnaround (#PCDATA) >
<!ELEMENT useLimitation (#PCDATA) >
<!ELEMENT propertyRightsCode EMPTY >
<!ATTLIST propertyRightsCode
      Restrict (copyright | patent | patentPending | license |
                 intellectualPropertyRights | trademark) #REQUIRED >
<!ELEMENT useConstraintsCode EMPTY >
<!ATTLIST useConstraintsCode
      Restrict (copyright | patent | patentPending | license |
                 intellectualPropertyRights | trademark) #REQUIRED >
<!ELEMENT otherConstraints (#PCDATA) >
<!ELEMENT classificationCode EMPTY >
<!ATTLIST classificationCode
      Classification (unclassified | codeWord | confidential | secret |
                     restricted | topsecret) #REQUIRED >
<!ELEMENT userNote (#PCDATA) >
<!ELEMENT classificationSystem (#PCDATA) >
<!ELEMENT handlingDescription (#PCDATA) >
<!ELEMENT TBD_ServiceQuality (#PCDATA) >
<!ELEMENT keyword (#PCDATA) >
<!ELEMENT typeCode EMPTY >
<!ATTLIST typeCode
      KeyType (discipline | place | stratum | temporal | theme)
      #REQUIRED >
<!ELEMENT thesaurusName (#PCDATA) >
<!ELEMENT operationName %DistinguishedName; >
<!ELEMENT operationDescription (#PCDATA) >
<!ELEMENT parameter (parameterName,parameterType,parameterDescription?,
                     permittedValues) >
<!ATTLIST parameter
      optional (yes | no) #REQUIRED
      repeatable (true | false) #REQUIRED
      direction (in | out | inout) #REQUIRED >

```

```

<!ELEMENT dependsOn  (operationName*) >
<!ELEMENT DCP    (invocationName,connectPoint,parameter*) >
<!ATTLIST DCP
          type (HTTPGet | HTTPPost)    #REQUIRED  >
<!ELEMENT title   (#PCDATA) >
<!ELEMENT alternateTitle  (#PCDATA) >
<!ELEMENT date    (#PCDATA) >
<!ATTLIST date
          dateType (creation | publication | revision)  #REQUIRED  >
<!ELEMENT edition  (#PCDATA) >
<!ELEMENT editionDate (#PCDATA) >
<!--"editionDate" has a domain of Date which is defined in another standard-->
<!ELEMENT identifier (#PCDATA) >
<!ELEMENT identifierType (#PCDATA) >
<!ELEMENT citedResponsibleParty %CI_ResponsibleParty; >
<!ELEMENT presentationFormCode EMPTY >
<!ATTLIST presentationFormCode
          value (document | hardcopyMap | image | model | profile | rasterMap |
          table | vectorMap | view)  #REQUIRED  >
<!ELEMENT seriesName (#PCDATA) >
<!ELEMENT issueIdentification (#PCDATA) >
<!ELEMENT otherCitationDetails (#PCDATA) >
<!ELEMENT collectionTitle (#PCDATA) >
<!ELEMENT page   (#PCDATA) >
<!ELEMENT ISBN   (#PCDATA) >
<!ELEMENT ISSN   (#PCDATA) >
<!--count of ("individualName" + "organisationName" + "positionName") > 0-->
<!ELEMENT individualName (#PCDATA) >
<!ELEMENT organisationName (#PCDATA) >
<!ELEMENT positionName (#PCDATA) >
<!ELEMENT contactInfo (phone?,address?,onLineResource?,hoursOfService?,
          contactInstructions?) >
<!ELEMENT roleCode EMPTY >
<!ATTLIST roleCode
          value (contentProvider | custodianSteward | owner | user |
          distributor | metadataProvider | originator | pointOfContact |
          principalInvestigator | processor | publisher)  #REQUIRED  >
<!ELEMENT nameValue (#PCDATA) >
<!ELEMENT nameNameSpace (#PCDATA) >
<!ELEMENT parameterName %DistinguishedName; >

```

```

<!ELEMENT parameterType    EMPTY  >

<!ATTLIST parameterType
      type (string | number)  #REQUIRED  >

<!ELEMENT parameterDescription  (#PCDATA)  >

<!ELEMENT permittedValues  (onLineResource?,(%ValueType;)* )  >

<!ELEMENT invocationName  (#PCDATA)  >

<!ELEMENT connectPoint  %CI_OnLineResource;  >

<!ELEMENT phone  (voice*,facsimile*,other*,otherType*)  >

<!ELEMENT address  (deliveryPoint*,city?,administrativeArea?,postalCode?,country?,
                     electronicMailAddress*)  >

<!ELEMENT onLineResource  %CI_OnLineResource;  >

<!ELEMENT hoursOfService  (#PCDATA)  >

<!ELEMENT contactInstructions  (#PCDATA)  >

<!ELEMENT dataType    EMPTY  >

<!ATTLIST dataType
      type (string | number)  #REQUIRED  >

<!ELEMENT instanceValue  %typedDataValue;  >

<!ELEMENT range  (minimumValue,maximumValue)  >

<!ELEMENT enumValues  (%typedDataValue;)*  >

<!--"other" is mandatory if "voice" and "facsimile" not provided
"otherType" is mandatory if "other" is provided-->
<!ELEMENT voice  (#PCDATA)  >

<!ELEMENT facsimile  (#PCDATA)  >

<!ELEMENT other  (#PCDATA)  >

<!ELEMENT otherType  (#PCDATA)  >

<!ELEMENT deliveryPoint  (#PCDATA)  >

<!ELEMENT city  (#PCDATA)  >

<!ELEMENT administrativeArea  (#PCDATA)  >

<!ELEMENT postalCode  (#PCDATA)  >

<!ELEMENT country  (#PCDATA)  >

<!ELEMENT electronicMailAddress  (#PCDATA)  >

<!ELEMENT minimumValue  %typedDataValue;  >

<!ELEMENT maximumValue  %typedDataValue;  >

<!ELEMENT linkage    EMPTY  >

<!ATTLIST linkage
      xmlns:xlink  CDATA      #FIXED "http://www.w3.org/1999/xlink"
      xlink:type   CDATA      #FIXED "simple"
      xlink:href   CDATA      #REQUIRED  >

<!ELEMENT protocol  (#PCDATA)  >

```

```

<!ELEMENT applicationProfile  (#PCDATA) >
<!ELEMENT onlineResourceName  (#PCDATA) >
<!ELEMENT onlineResourceDescription  (#PCDATA) >
<!ELEMENT functionCode  EMPTY  >
<!ATTLIST functionCode
      value (access | additionalInformation | download | order | search)
      #REQUIRED  >
<!ELEMENT valueTitle  (#PCDATA) >
<!ELEMENT valueDescription  (#PCDATA) >
<!ELEMENT valueOnLineResource  %CI_OnLineResource; >
<!ELEMENT value  (#PCDATA) >
<!ATTLIST value
      type (string | number)  #REQUIRED  >

```

#### ***Transaction Operation - Informative***

At the moment, the <Transaction> interface is identical to the one defined for the WFS and the reader is referred to that document for a complete description. Obviously, the “FeatureType” of the WFS would be replaced by the “RecordType” in the context of this document.

#### ***LockRecord Interface -Informative***

The <LockRecord> interface is analogous to the <LockFeature> interface defined for the WFS and the reader is referred to that document for a complete description.

#### ***RegisterService Operation - Informative***

##### **Request DTD**

TBS

##### **Keyword=Value Encoding**

<b><i>URL Component</i></b>	<b>Required/ Optional</b>	<b>Description</b>
<b>http://server_address/path/script?</b>	R	URL prefix of server.
<b>VERSION=version</b>	R	Request version.
<b>REQUEST=RegisterService</b>	R	Request name.
<b>ServiceAddr</b>	R	URL of the Service GetCapability Operation
<b>ServiceOwnerContactInfo</b>	O	Contact point for notification of events
<b>HarvestFrequency</b>	O	Frequency for catalog to check for and harvest new values from the Service capabilities document
<b>User Transactions accepted</b>	O	May a service provider push his update to the WRS in what form. New Capability object or a WFS Transaction object.
<b>Vendor-specific parameters</b>	O	Optional experimental parameters.

## APPENDIX A – Normative DTD

---

### GetCapabilities\_Request.dtd

```
<!-- **** REQUEST: GETCAPABILITIES -->
<!-- *      version CDATA #Implied          * -->
<!-- **** -->

<!ELEMENT GetCapabilities EMPTY>
<!ATTLIST GetCapabilities version CDATA #Implied
               service CDATA #Implied
               section (ALL | SERVICE | DATA) #Implied>
```

### GetCapabilities\_Response.dtd

```
<!ELEMENT STSC_Capabilities (Service, Capability)>
<!ATTLIST STSC_Capabilities
           version CDATA #FIXED "0.0.1"
           updateSequence CDATA "0"
>
<!-- The SCHEMALANGUAGES entity defines the
    schema languages that a feature server is
    capable of using to describe the schema of
    a feature. This entity can be redefined
    by individual servers to include other
    schema languages but XMLSCHEMA must always
    be defined. -->
<!ENTITY % SCHEMALANGUAGES "XMLSCHEMA">
<!ELEMENT XMLSCHEMA EMPTY>
<!-- The RESULTFORMATS entity defines the
    output formats that the web feature server
    can generate. The mandatory format "GML2"
    must always be available. This entity can
    be redefined by individual servers to
    include other formats. -->
<!ENTITY % RESULTFORMATS " XML">
<!ELEMENT XML EMPTY>
<!-- Elements used in multiple places. -->
<!-- The Name is typically for machine-to-machine communication. -->
<!ELEMENT Name (#PCDATA)>
<!-- The Title is for informative display to a human. -->
<!ELEMENT Title (#PCDATA)>
<!-- The abstract is a longer narrative description of an object. -->
<!ELEMENT Abstract (#PCDATA)>
<!-- An OnlineResource is typically an HTTP URL. The URL is placed
    in the xlink:href attribute. The xmlns:xlink attribute is a
    required XML namespace declaration. -->
<!ELEMENT OnlineResource EMPTY>
<!ATTLIST OnlineResource
           xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
           xlink:type CDATA #FIXED "simple"
           xlink:href CDATA #REQUIRED
>
<!-- A container for listing an available format's MIME type. -->
<!ELEMENT Format (#PCDATA)>
<!-- General service metadata. -->
<!ELEMENT Service (Name, Title, Abstract?, KeywordList?, OnlineResource,
ContactInformation?, Fees?, AccessConstraints?)>
<!-- List of keywords or keyword phrases to help catalog searching. -->
<!ELEMENT KeywordList (Keyword*)>
<!-- A single keyword or phrase. -->
<!ELEMENT Keyword (#PCDATA)>
<!-- Information about a contact person for the service. -->
<!ELEMENT ContactInformation (ContactPersonPrimary?, ContactPosition?, ContactAddress?,
ContactVoiceTelephone?, ContactFacsimileTelephone?, ContactElectronicEmailAddress?)>
<!--The primary contact person.-->
<!ELEMENT ContactPersonPrimary (ContactPerson, ContactOrganization)>
<!--The person to contact.-->
```

```

<!ELEMENT ContactPerson (#PCDATA)>
<!--The organization supplying the service.-->
<!ELEMENT ContactOrganization (#PCDATA)>
<!--The position title for the contact person.-->
<!ELEMENT ContactPosition (#PCDATA)>
<!--The address for the contact supplying the service.-->
<!ELEMENT ContactAddress (AddressType, Address, City, StateOrProvince, PostCode,
Country)>
<!--The type of address.-->
<!ELEMENT AddressType (#PCDATA)>
<!--The street address.-->
<!ELEMENT Address (#PCDATA)>
<!--The address city.-->
<!ELEMENT City (#PCDATA)>
<!--The state or province.-->
<!ELEMENT StateOrProvince (#PCDATA)>
<!--The zip or postal code.-->
<!ELEMENT PostCode (#PCDATA)>
<!--The address country.-->
<!ELEMENT Country (#PCDATA)>
<!--Contact telephone number.-->
<!ELEMENT ContactVoiceTelephone (#PCDATA)>
<!--The contact fax number.-->
<!ELEMENT ContactFacsimileTelephone (#PCDATA)>
<!--The e-mail address for the contact.-->
<!ELEMENT ContactElectronicMailAddress (#PCDATA)>
<!-- Elements indicating what fees or access constraints are imposed. -->
<!ELEMENT Fees (#PCDATA)>
<!ELEMENT AccessConstraints (#PCDATA)>
<!-- A Capability lists available request
      types, how exceptions may be reported, and
      whether any vendor-specific capabilities
      are defined. It also lists all the
      feature types available from this feature
      server. -->
<!ELEMENT Capability (Request, VendorSpecificCapabilities?, Exceptions, PresentOptions,
RecordTypeList)>

<!ELEMENT Request (GetCapabilities | DescribeRecordType | GetRecord | LockRecord |
Transaction | RegisterService)+>
<!ELEMENT GetCapabilities (DCPType+)>
<!ELEMENT DescribeRecordType (SchemaDescriptionLanguage, DCPTYPE+)>
<!ELEMENT SchemaDescriptionLanguage (%SCHEMALANGUAGES;)+>
<!ELEMENT GetRecord (ResultFormat, DCPTYPE+)>
<!ELEMENT ResultFormat (%RESULTFORMATS;)+>
<!ELEMENT LockRecord (DCPType+)>
<!ELEMENT Transaction (DCPType+)>
<!ELEMENT RegisterService (DCPType+)>

<!-- Available Distributed Computing Platforms
      (DCPs) are listed here. At present, only
      HTTP is defined. -->
<!ELEMENT DCPTYPE (HTTP)>
<!-- Available HTTP request methods. -->
<!ELEMENT HTTP (Get | Post)+>
<!-- HTTP request methods. The onlineResource
      attribute indicates the URL prefix for
      HTTP GET requests (everything before the
      question mark and query string:
      http://hostname[:port]/path/scriptname);
      for HTTP POST requests, onlineResource is
      the complete URL. -->
<!ELEMENT Get EMPTY>
<!ATTLIST Get onlineResource CDATA #REQUIRED >
<!ELEMENT Post EMPTY>
<!ATTLIST Post onlineResource CDATA #REQUIRED >
<!-- The optional VendorSpecificCapabilities
      element lists any capabilities
      unique to a particular server. Because
      the information is not known a priori, it
      cannot be constrained by this particular

```

```

DTD. A vendor-specific DTD fragment must
be supplied at the start of the XML
capabilities document, after the reference
to the general STSC_Capabilities DTD.

-->
<!-- DEFINE THIS ELEMENT AS NEEDED IN YOUR XML
    <!ELEMENT VendorSpecificCapabilities (your stuff here)>
-->
<!ELEMENT VendorSpecificCapabilities EMPTY>
<!-- An Exception element indicates which error-reporting
    formats are supported. -->
<!ELEMENT Exceptions (Format+)>
<!--Presentation Options Supported by Server -->
<!ELEMENT PresentOptions EMPTY>
<!ATTLIST PresentOptions
    Sort (0 | 1) "0"
    StartRec (0 | 1) "0"
    Hits (0 | 1) "1"
    RecsMax CDATA #IMPLIED
>
<!-- A list of feature types available from
    this server. The following table
    specified the number and source of the
    various elements that are available for
    describing a feature type.

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

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

    Abstract      0/1   optional; no Default

    Keywords      0/1   optional; no Default

    SRS           0/1   the SRS that should be used
                      when specifying the state of
                      the feature

    LatLonBoundingBox 0/1 exactly one is required;
                      default from parent
    Operations     1+    allowable operations for this record type

    MetadataURL   0+    optional; no default
-->
<!ELEMENT RecordTypeList (RecordType+)>
<!ELEMENT RecordType (Name, Title?, Abstract?, Keywords?, SRS, Operations?,  

LatLonBoundingBox?, MetadataURL*)>
<!ELEMENT Keywords (#PCDATA)>
<!ELEMENT SRS (#PCDATA)>

<!ELEMENT Operations (Insert | Update | Delete | Present |
                      Query | Lock)+>
<!ELEMENT Insert EMPTY >
<!ELEMENT Update EMPTY>
<!ELEMENT Delete EMPTY >
<!ELEMENT Query EMPTY >
<!ELEMENT Lock EMPTY>
<!ELEMENT Present EMPTY>

<!-- The LatLonBoundingBox attributes indicate
    the edges of the enclosing rectangle in
    latitude/longitude decimal degrees (as in
    SRS EPSG:4326 [WGS1984 lat/lon]).  

    LatLonBoundingBox MUST be supplied
    regardless of what SRS the server may
    support, but it MAY be approximate if

```

```

EPSG:4326 is not supported. Its purpose
is to facilitate geographic searches
without requiring coordinate
transformations by the search engine. -->
<!ELEMENT LatLonBoundingBox EMPTY>
<!ATTLIST LatLonBoundingBox
    minx CDATA #REQUIRED
    miny CDATA #REQUIRED
    maxx CDATA #REQUIRED
    maxy CDATA #REQUIRED
>
<!-- A CatalogServer MAY use zero or more
MetadataURL elements to offer
detailed, standardized metadata about the
data underneath a particular record type.
The type attribute indicates the standard
to which the metadata complies; the format
attribute indicates how the metadata is
structured. Four types are defined at present:
'TC211' = ISO TC211 19115;
'FGDC' = FGDC CSDGM.
'DEDSL' = CCSDS Data Entity Dictionary Specification Language
'ISO11179' = ISO 11179 part3 Data Element definitions
-->
<!ELEMENT MetadataURL (#PCDATA)>
<!ATTLIST MetadataURL
    type (TC211 | FGDC | DEDSL | ISO11179) #IMPLIED
    format (XML | SGML | TXT) #REQUIRED
>

```

### **DescribeRecordType\_Request.dtd**

```

<!-- **** REQUEST: DESCRIBE RECORD TYPE * -->
<!-- * SCHEMALANGUAGES "(XMLSCHEMA)" -->
<!ELEMENT DescribeRecordType (TypeName*)>
<!ATTLIST DescribeRecordType
    outputFormat (%SCHEMALANGUAGES;) #IMPLIED
>
<!ELEMENT TypeName (#PCDATA)>

```

### **GetRecord\_Request.dtd**

```

<!ENTITY % FILTERDTD SYSTEM "Filter.dtd">
%FILTERDTD;

<!-- * REQUEST: GETRECORD * -->
<!-- * METASTANDARDS "FGDC | ISO19119" -->
<!ELEMENT GetRecord (Query+, Filter?)>
<!ATTLIST GetRecord
    handle CDATA #IMPLIED
    maxRecords CDATA #IMPLIED
    startPosition CDATA #IMPLIED
    outputFormat (XML) #REQUIRED
    outputRecType (%METASTANDARDS;) #REQUIRED
>
<!ELEMENT Query ((PropertySet | PropertyName*?), Filter?)>
<!ATTLIST Query handle CDATA #IMPLIED
    typeName (Service | Product | Collection) #REQUIRED>
<!ELEMENT PropertySet EMPTY>
<!ATTLIST PropertySet
    setName (Brief | Summary | Full | Hits) "Brief"
>
```

### **GetRecord\_Response.dtd**

```
<!-- **** -->
```

```

<!-- *      RESPONSE: GETRECORD                                * -->
<!-- ***** * ***** * ***** * ***** * ***** * ***** * -->

<!ENTITY % DistinguishedName "(nameValue,nameNameSpace)">
<!ENTITY % CI_OnlineResource "(linkage,
                               protocol?,
                               applicationProfile?,
                               onlineResourceName?,
                               onlineResourceDescription?,
                               functionCode?)">
<!-- ===== -->
<!-- The top level holder for a response from the -->
<!-- catalog.                                         -->
<!-- ===== -->
<!ELEMENT searchResponse (searchParameter*, searchStatus, searchResults)>
<!ATTLIST searchResponse
          DTD_Version CDATA #FIXED "1.1.0"
>
<!ELEMENT searchParameter (#PCDATA)>
<!ATTLIST searchParameter
          name CDATA #REQUIRED
>
<!-- ===== -->
<!-- This tag contains a number of attributes that detail -->
<!-- the status of the search                         -->
<!-- ===== -->
<!ELEMENT searchStatus EMPTY>
<!-- ===== -->
<!-- elementSetName - The element set that has been   -->
<!--           returned (e.g., brief, summary,          -->
<!--           full)                                 -->
<!-- numberOfRecords - The number of hits returned    -->
<!--           in this result.                      -->
<!-- schema       - The type of response returned (e.g. -->
<!--           OGCService, FGDC)                   -->
<!-- success      - Was the search successful (true,   -->
<!--           false)                            -->
<!-- timestamp    - The date and time at the completion -->
<!--           of the search.                     -->
<!-- ===== -->
<!ATTLIST searchStatus
          elementSetName CDATA #FIXED "brief"
          success CDATA "true"
          numberOfRecords CDATA #IMPLIED
          schema CDATA #FIXED "ISO19119"
          timestamp CDATA #IMPLIED
>
<!-- The holder for the results from the catalog. Although
     defined as EMPTY here, an actual server would insert
     the root element(s) of the metadata schemas in which
     it can present results.
-->
<!ELEMENT searchResults EMPTY>

```

### **FeatureId.dtd**

```

<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT FeatureId EMPTY>
<!ATTLIST FeatureId fid ID #REQUIRED>

```

### **Expr.dtd**

```

<!ENTITY % expr "(Add|Sub|Mul|Div|PropertyName|Literal|Function)">
<!ELEMENT Expression (%expr;)*>
<!ELEMENT PropertyName (#PCDATA)>
<!ELEMENT Literal ANY>
<!ELEMENT Function (%expr;)*>
<!ATTLIST Function
          name CDATA #REQUIRED
>

```

```

<!ELEMENT Add (%expr;, %expr;)gt;
<!ELEMENT Sub (%expr;, %expr;)gt;
<!ELEMENT Mul (%expr;, %expr;)gt;
<!ELEMENT Div (%expr;, %expr;)gt;

```

### Filter.dtd

```

<!ENTITY % GEOMETRYDTD SYSTEM "gmlgeometry.dtd">
%GEOMETRYDTD;
<!ENTITY % FEATUREIDDTD SYSTEM "FeatureId.dtd">
%FEATUREIDDTD;
<!ENTITY % EXPRDTD SYSTEM "Expr.dtd">
%EXPRDTD;
<!ENTITY % spatial_ops "( Equals
                           Disjoint
                           Intersects
                           Touches
                           Crosses
                           Within
                           Contains
                           Overlaps
                           Beyond
                           BBOX )">
<!ENTITY % comparison_ops "(PropertyIsEqualTo
                           PropertyIsLessThan
                           PropertyIsGreaterThan
                           PropertyIsLessThanOrEqualTo
                           PropertyIsGreaterThanOrEqualTo
                           PropertyIsLike
                           PropertyIsNull
                           PropertyIsBetween)">
<!ENTITY % logical_ops "(And|Or|Not)">

<!ELEMENT QueryConstraint (Filter|CqlText|SfSql|SqlMM)>
<!ELEMENT CqlText (#PCDATA)>
<!ELEMENT SfSql (#PCDATA)>
<!ELEMENT SqlMM (#PCDATA)>

<!ELEMENT Filter (%spatial_ops; | %comparison_ops; | %logical_ops; | FeatureId+)>
<!ELEMENT Equals (PropertyName, %GeometryClasses;)>
<!ELEMENT Disjoint (PropertyName, %GeometryClasses;)>
<!ELEMENT Intersects (PropertyName, %GeometryClasses;)>
<!ELEMENT Touches (PropertyName, %GeometryClasses;)>
<!ELEMENT Crosses (PropertyName, %GeometryClasses;)>
<!ELEMENT Within (PropertyName, %GeometryClasses;, Distance)>
<!ELEMENT Contains (PropertyName, %GeometryClasses;)>
<!ELEMENT Overlaps (PropertyName, %GeometryClasses;)>
<!ELEMENT Beyond (PropertyName, %GeometryClasses;, Distance)>
<!ELEMENT BBOX (PropertyName, %GeometryClasses;)>
<!ELEMENT Distance (#PCDATA)>
<!ELEMENT PropertyIsEqualTo (%expr;, %expr;)gt;
<!ELEMENT PropertyIsLessThan (%expr;, %expr;)gt;
<!ELEMENT PropertyIsGreaterThan (%expr;, %expr;)gt;
<!ELEMENT PropertyIsLessThanOrEqualTo (%expr;, %expr;)gt;
<!ELEMENT PropertyIsGreaterThanOrEqualTo (%expr;, %expr;)gt;
<!ELEMENT PropertyIsLike (PropertyName, Literal)>
<!ATTLIST PropertyIsLike
      wildCard CDATA #REQUIRED
      escape CDATA #REQUIRED
>
<!ELEMENT PropertyIsNull (PropertyName | Literal)>
<!ELEMENT PropertyIsBetween (PropertyName, LowerBoundary, UpperBoundary)>
<!ELEMENT LowerBoundary (%expr;)gt;
<!ELEMENT UpperBoundary (%expr;)gt;
<!ELEMENT And ((%comparison_ops; | %spatial_ops; | %logical_ops;), (%comparison_ops; | %spatial_ops; | %logical_ops;)+)>
<!ELEMENT Or ((%comparison_ops; | %spatial_ops; | %logical_ops;), (%comparison_ops; | %spatial_ops; | %logical_ops;)+)>
<!ELEMENT Not ((%comparison_ops; | %spatial_ops; | %logical_ops;), (%comparison_ops; | %spatial_ops; | %logical_ops;)+)>

```

### **Sort.dtd**

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % EXPRDTD SYSTEM "Expr.dtd">
%EXPRDTD;
<!ELEMENT SortBy (PropertyName)* >
<!ATTLIST SortBy sortOrder (DESC | ASC) "ASC">
```

### **Transaction\_Request.dtd**

```
<!ENTITY % FILTERSTUFF SYSTEM "Filter.dtd">
%FILTERSTUFF;
<!ENTITY % NATIVEDTD SYSTEM "Native.dtd">
%NATIVEDTD;

<!-- For a particular WFS, this entity should be
     redefined to describe the list of possible features
     that can be acted upon. For example, the entity
     might look like: &lt;!ENTITY % FEATURETYPES "F1|F2..."&gt;
--&gt;
&lt;!ENTITY % FEATURETYPES "DummyFeature"&gt;
&lt;!ELEMENT DummyFeature EMPTY&gt;

&lt;!-- *****
 * Transaction REQUEST
 **** --&gt;
&lt;!ELEMENT Transaction (LockId?, (Insert | Update | Delete | Native)*)&gt;
&lt;!ELEMENT LockId (#PCDATA)&gt;
&lt;!ATTLIST Transaction handle CDATA #IMPLIED
           releaseAction (ALL | SOME) #IMPLIED&gt;

&lt;!ELEMENT Insert (%FEATURETYPES;)+&gt;
&lt;!ATTLIST Insert handle CDATA #IMPLIED&gt;

&lt;!ELEMENT Update (%FEATURETYPES;, Filter)&gt;
&lt;!ATTLIST Update handle CDATA #IMPLIED&gt;

&lt;!ELEMENT Delete (Filter)&gt;
&lt;!ATTLIST Delete handle CDATA #IMPLIED
           typeName CDATA #IMPLIED&gt;</pre>

```

### **Transaction\_Response.dtd**

```
<!ENTITY % FEATUREIDDTD SYSTEM "FeatureId.dtd" >
%FEATUREIDDTD;

<!-- *****
 * Transaction RESPONSE
 **** -->
<!ELEMENT WFS_TransactionResponse (InsertResult*, TransactionResult)>
<!ATTLIST WFS_TransactionResponse
           version CDATA #REQUIRED
>
<!ELEMENT InsertResult (FeatureId+)>
<!ATTLIST InsertResult
           handle CDATA #IMPLIED
>
<!ELEMENT TransactionResult (Status, Locator?, Message?)>
<!ATTLIST TransactionResult
           handle CDATA #IMPLIED
>
<!ELEMENT Status (SUCCESS | FAILED | PARTIAL)>
<!ELEMENT Locator (#PCDATA)>
<!ELEMENT Message (#PCDATA)>
<!ELEMENT SUCCESS EMPTY>
<!ELEMENT FAILED EMPTY>
<!ELEMENT PARTIAL EMPTY>
```

### **LockRecord\_Request.dtd**

```

<!ENTITY % FILTERDTD SYSTEM "Filter.dtd">
%FILTERDTD;
<!-- **** LockRecord REQUEST **** -->
<!ELEMENT LockRecord (Expiry?, Lock+)>
<!ELEMENT Expiry (#PCDATA)>
<!ELEMENT Lock (Filter?)>
<!ATTLIST Lock
      handle CDATA #IMPLIED
      typeName CDATA #IMPLIED
      releaseAction (ALL | SOME) #IMPLIED
>

```

### **LockRecord\_Response.dtd**

```

<!ENTITY % FEATUREIDDTD SYSTEM "FeatureId.dtd" >
%FEATUREIDDTD;
<!-- **** LockRecordRESPONSE **** -->
<!ELEMENT WFS_LockRecordResponse (LockId?, RecordsLocked?, RecordsNotLocked?)>
<!ELEMENT LockId (#PCDATA)>
<!ELEMENT RecordsLocked (FeatureId)+>
<!ELEMENT RecordsNotLocked (FeatureId)+>

```

### **Native.dtd**

```

<!-- **** NATIVE REQUEST **** -->
<!ELEMENT Native (#PCDATA)>
<!ATTLIST Native vendorId      CDATA #REQUIRED
           safeToIgnore (False|True) #REQUIRED>

```