

Comparison of WFS, WRS and OGC Catalog Services

Background

The purpose of this note is to address some of the issues that have resulted from the release of the WRS discussion paper. There have been three major areas of discussion:

- Development of a general Stateless Catalog Model within the OGC Catalog Specification ,
 - the compatibility of the WRS interfaces and parameter names with the OGC Catalog General Model,
 - the use of the name WRS rather than Stateless Catalog, and
 - the emphasis of service r search, rather than service or data
- The use of WFS interfaces to implement Stateless Catalog Functionality
- The need for a RegisterService interface and the periodic, transparent harvesting of updates

This note discusses the first and second points with emphasis on the similarity and differences between WFS and WRS interfaces and concepts. The acronym WRS is used in this document as a shorthand for the profile of the OGC Stateless Catalog Service for the OGC Basic Services Model (BSM) Web Services DCP. It will be integrated with OGC Catalog Services.

Analysis

There were several informal meetings in the Liege TC between myself and Serge Margoulies to discuss the concept of using WFS interfaces to implement WRS. Figure 1 below was derived from these meetings to illustrate a Web Registry .Server with the similar interfaces from WFS and the Catalog Services General model listed.. From the results of that meeting the following issues were noted.

- The WFS would need to add a RegisterService interface
- The WFS DescribeFeatureType interface provided a working implementation of the Catalog Services ExplainSchema interface which was missing from the WRS draft
- There was concern about the use of the current WFS interface names due to the overloading of the term Feature
- There was concern that GML2 was overkill for Catalog Data Structures

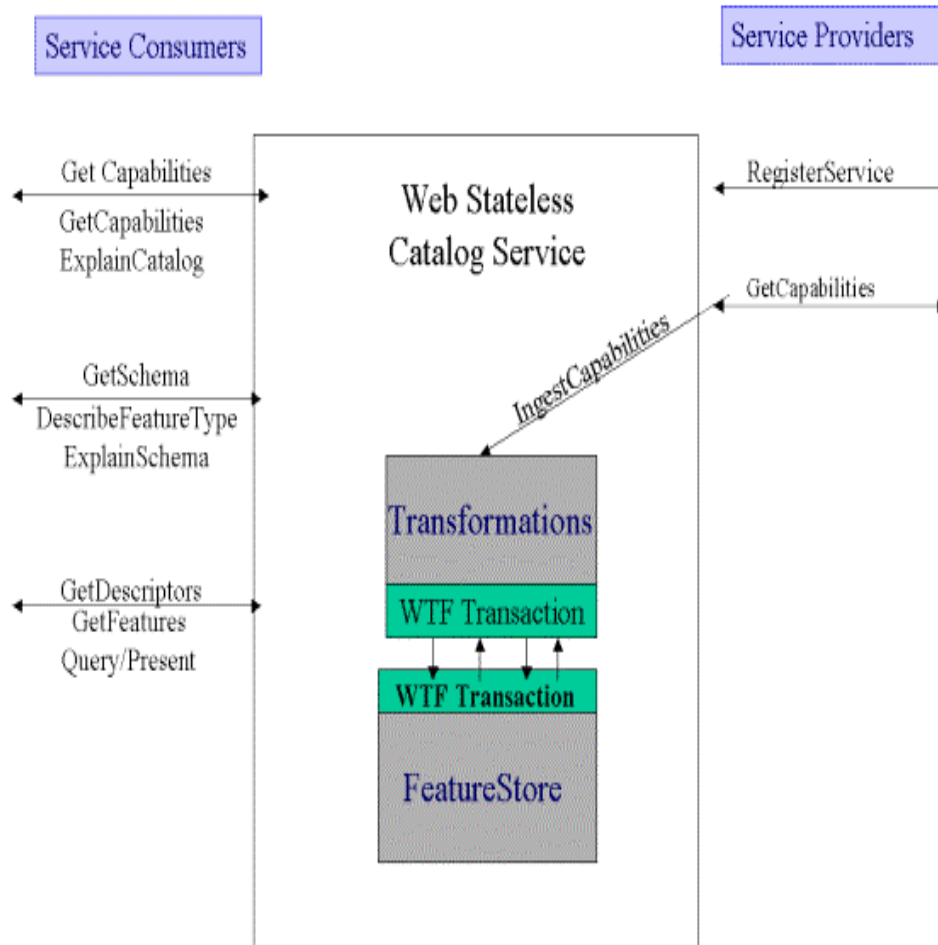


Figure 1 Web Registry Server

Over the past few days I have done more analysis of the WFS and WRS interfaces and parameter names as compared to those General Catalog Services Model and the WWW Profile. Some of this analysis can be found in Tables 1 and 2. The parameter names in the WWW search command were omitted from Table 2 due to time constraints. Though they can be mapped to the General Catalog Service parameters, the names were developed for compatibility with the CIP and GEO profiles of Z39.50 that were the prevalent spatially enabled catalog systems protocols in the FGDC and EO communities.

During the detailed analysis of parameters the following issues arose:

- There is currently a conceptual difference in the object model of WFS and the General Catalog Services Model . The WFS has one type “FeatureType” that provides the property schema for both queries and retrieval of Features. The Catalog model and the WRS specification have two separate entities “AttributeSets” for queries and “ElementSets” for retrieval. There may be an order of magnitude difference between the number of queryable Attributes and the number of Elements in a catalog entry. There also may be a level of abstraction difference between an AttributeSets (eg. Dublin Core) designed for broad interoperability and an ElementSet (e.g. ISO 19115) designed for completeness.
- In a similar vein, the General Catalog Service model allows a consumer to enter either a list of element names to be returned or the name of a common prepackaged subset of elements (i.e. brief, summary, full) that do not depend on knowledge of the underlying data model. Currently the WFS supports the list of element/property names and the WRS supports the prepackaged structure.
- There needs to be a discussion of the meaning of “stateless” among the WRS and WFS teams. The existence of locking transactions and distributed/cascading searches in the WFS implies “stateful objects” in the WFS while the WRS definition involves the assumption no memory of previous state in either the client or server.

Conclusion

There are no insurmountable reasons for not using the WFS interfaces as the basis for an “OGC Web Services” profile of Stateless Catalog Services. However there are several issues on the tuning of the WFS for Discovery ranging from Object Model to Naming that need to be addressed in the near future to enable this activity

Tables

General Model Operation	WWW Profile Service	WRS/Stateless Profile	WFS Spec
CG_InitSessionRequest	initRequest ¹	N/A	N/A
CG_InitSessionResponse	InitResponse ¹	N/A	N/A
CG_TerminateRequest	close ²	N/A	N/A
CG_TerminateResponse	close	N/A	
CG_ExplainServerRequest	searchRequest ^{3, 4}	GetCapabilities	GetCapabilities
CG_ExplainServerResponse	searchResponse	GetCapabilities	GetCapabilities
CG_StatusRequest	triggerResourceControlRequest	N/A	N/A
CG_StatusResponse	resourceControlRequest	N/A	N/A
CG_CancelRequest	triggerResourceControlRequest	N/A	N/A
CG_CancelResponse	none ⁵	N/A	N/A
CG_QueryRequest	searchRequest ^{3,6} and sortRequest	GetDescriptors	GetFeature
CG_QueryResponse	searchResponse and sortResponse	GetDescriptors	GetFeature
CG_PresentRequest	presentRequest	GetDescriptors (N/A)	GetFeature (N/A)
CG_PresentResponse	presentResponse	GetDescriptors (N/A)	GetFeature (N/A)
CG_ExplainCollectionRequest	searchRequest ⁷	GetCapabilities	DescribeFeatureType
CG_ExplainCollectionResponse	searchResponse ⁷	GetCapabilities	DescribeFeatureType
CG_BrokeredAccessRequest	extendedServicesRequest ⁸	?	?
CG_BrokeredAccessResponse	extendedServicesResponse ⁸	?	?
		RegisterService	

Table 1:Interface Comparison

General Model	WRS/Stateless	WFS
sessionID ::= Integer	NA	NA
destinationID ::= CharacterString	NA	NA
requestID ::= CG_RequestID	Replicate query in result set	handle
additionalInfo ::= CharacterString		

queryExpression ::= CG_QueryExpression		Query
theLanguage	queryLanguage	TBA
theNamespace	attributeSetName	featureType
theQuery	queryExpression	Filter
resultType ::= CG_ResultType	?	?
iteratorSize ::= Integer	maxRecs	maxRecs
cursor ::= Integer	startPosition	-----
returnFormat ::= CG_MessageFormat	returnFormat	outputFormat
presentation ::= CG_PresentationDescription - attributes: (type = sequence<RecordType>) – list of attribute name/type pairs - name: (type = CG_PredefinedPresentationType) –), identifying a predefined presentation type.	ElementSetName = PredefinedPresentationType	PropertyName*=attributes
sortField ::= Set<CG_SortField>	sortKey	
queryScope ::= CG_QueryScope	QueryScope	TBD – type of service, local,cascading,distributed
collectionID ::= CG_CollectionName		
catalogType ::= CG_CatalogEntryType	catalogType	featureType

Table 2:Parameter Comparison of “Get Objects Interfaces”