
Open GIS Consortium, Inc.

**OpenGIS® Conformance Test Guidelines for OpenGIS
Catalog Services Specification for CORBA**

00-027r1

Release Date: June 9, 2000

Geodan Holding bv, the Netherlands

The companies listed above have granted the Open GIS Consortium, Inc. (OGC) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version.

Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright, in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

NOTICE

The information contained in this document is subject to change without notice.

The material in this document details an Open GIS Consortium specification in accordance with the license and notices set forth on this page. This document does not represent a commitment to implement any portion of this specification in any company's products.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, THE OPEN GIS CONSORTIUM AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. The Open GIS Consortium and the companies list above shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

The copyright holders list above acknowledge that the Open GIS Consortium (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks, or other special designations to indicate compliance with these materials.

This document contains information, which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems—without permission of the copyright owner.

RESTRICTED RIGHTS LEGEND. Use, duplication, or disclosure by government is subject to restrictions as set forth in subdivision (c)(1)(ii) of the Right in Technical Data and Computer Software Clause at DFARS 252.227.7013

OpenGIS® is a trademark or registered trademark of Open GIS Consortium, Inc. in the United States and in other countries

0 PREFACE.....	IV
0.1 SUBMITTING COMPANIES.....	IV
0.2 SUBMISSION CONTACT POINTS	IV
0.3 EDITORS.....	IV
0.4 DOCUMENT CONVENTIONS	IV
1 OVERVIEW	5
2 BACKGROUND.....	6
2.1 CATALOG SERVICES IMPLEMENTATION SPECIFICATION	6
2.2 CG_CATALOGSERVICE INTERFACE.....	6
2.3 CG_DISCOVERY INTERFACE.....	7
3 ENVIRONMENT AND SETUP OF THE CATALOG CERTIFICATION PROGRAM	8
3.1 SETTING UP THE ENVIRONMENT	8
3.2 SETTING UP THE SERVER	8

3.3	RUNNING THE CLIENT.....	8
4	DESCRIPTION OF THE TEST PROCEDURES	10
4.1	INTRODUCTION.....	10
4.2	OPERATIONS OF THE CG_CATALOGSERVICE INTERFACE.....	10
4.2.1	<i>Testing the InitSession operation.....</i>	10
4.2.2	<i>Testing the TerminateSession operation</i>	10
4.2.3	<i>Testing the ExplainServer operation</i>	11
4.3	OPERATIONS OF THE CG_DISCOVERY INTERFACE.....	11
4.3.1	<i>Query.....</i>	11
4.3.2	<i>Present.....</i>	15
4.3.3	<i>ExplainCollection</i>	16
	APPENDIX A: METADATA DATABASE IN XML FORMAT	17
	APPENDIX B: QUERY FILE USED TO TEST ASPECTS OF THE OGC QUERY LANGUAGE	19
	APPENDIX C: SOURCE CODE	22

0 Preface

0.1 Submitting Companies

The following company submitted this document:

- Geodan Holding bv, the Netherlands;

0.2 Submission Contact Points

All questions about the submission should be directed to:

Barend Gehrels
Geodan SDT bv, the Netherlands
tel: +31 73 692 5151
fax: +31 73 692 5150
E-mail: barend@geodan.nl
<http://www.geodan.nl>

0.3 Editors

The following editors contributed to this document

Laura Diaz, on behalf of Geodan Holding bv
University of Valencia, Spain
E-mail: laudiaz2@alumni.uv.es
<http://www.uv.es>

Bart van der Eijnden
Geodan SDT bv, the Netherlands
tel: +31 73 692 5151
fax: +31 73 692 5150
E-mail: bart@geodan.nl
<http://www.geodan.nl>

Barend Gehrels
Geodan SDT bv, the Netherlands
tel: +31 73 692 5151
fax: +31 73 692 5150
E-mail: barend@geodan.nl
<http://www.geodan.nl>

0.4 Document Conventions

The Courier New font has been used to indicate code segments.

1 Overview

This document describes the guidelines that the Open GIS Consortium will use for testing software implementations with respect to conformance to its specification entitled Revision 1.0.

The Open GIS Consortium, Inc. (OGC) maintains a brand (in the form of a certification mark) that cannot be used in connection with a software product by any organization unless it has submitted a software product to OGC's conformance test, successfully completed this test, and received a certificate stating such success. Organizations that have earned the certification mark may use it in ways defined within this document. This set of rules ensures that users who buy products that are branded can be sure that the products carrying the certification mark have been submitted to a test. The primary purpose of the conformance test is to protect the value of the OpenGIS® brand as an element of OGC's program to promote interoperability between diverse geoprocessing systems.

This document contains the following:

- An outline on the background of Catalog Services;
- A general description of how to configure the environment for the Catalog Certification Program;
- A description of the testing procedures of the Catalog Certification Program;
- Allowable adaptations: the testsuite may be enhanced and extended to include more options or more settings or more combinations of settings, or to do more extensive testing.

2 Background

2.1 Catalog Services Implementation Specification

According to the OpenGIS Catalog Services Implementation Specification, OGC Catalog servers have three kinds of services:

- **Discovery Service;**
- **Access Service;**
- **Management Service.**

Discovery Services are those services which allow a client to locate metadata that describe data. Access Services provide the client with methods to request services on the data. Access Services have been divided into two types: Direct Access and Brokered Access. Management Services define methods for a client to change the metadata held by a catalog.

The Discovery Service has to be provided by all Application Servers claiming compliance with the OGC Catalog Interface. The Access and Management Services are optional for an OGC compliant catalog.

The Catalog Services General Model provides the standards of compliance for specific implementation communities, in a profile-based way. There are different profiles for each Distributed Computing Platform, like CORBA, OLEDB and WWW. The test program is specific for CORBA in combination with a coarse-grained interface. This means that the client and server have very little knowledge of each other. Therefore, the interface in this environment must be flexible, well-defined and simple.

As specified in the Coarse-Grained General Interface Model there are four major interfaces, CG_CatalogService, CG_Discovery, CG_Access and CG_Manager. These interfaces allow the discovery, access and management of geospatial data and services. The above-mentioned model has been based on the concept of interface operations passing Request-Response Message Pairs between a client and a server. Stated another way, the Coarse-Grained architecture uses a messaging-based structure to describe the access and invocation of Catalog services.

The Catalog Certification Program is centered on the CG_Discovery interface, it will test all the functionality supported by the discovery service as well as all the functionality of the CG_CatalogService interface. The Catalog Certification Program focuses on these two interfaces because as said before the Access and Management Service are optional.

2.2 CG_CatalogService Interface

Server level interfaces provide access to the services that support the establishment and management of a user session through operations. The operations included in this interface that have to be supported by all servers are listed in figure 1. The *InitSession* operation generates a unique identifier used to track the context of a session. To terminate a session, the *TerminateSession* operation can be used. The *Status* operation checks the status of a currently pending request. To terminate a request, the *Cancel* operation should be used. The *ExplainServer* operation lists all conventions and services available during the current session.

In the current version of the Catalog Certification Program only synchronous requests are considered. This rules out the possibility to test for the CancelRequest and Status operation.

2.3 CG_Discovery Interface

The CG_Discovery Interface provides users with a way to discover what data, services and other resources are available to them. This interface does not provide access to the resources themselves. The operations provided by this interface can be deducted from figure 1. The *Query* operation searches for data on or services from a given catalog server and may return records from the result set. The *Present* operation retrieves records from a result set created by issuing a query. To get an explanation of the data model of the catalog the *ExplainCollection* operation can be used.

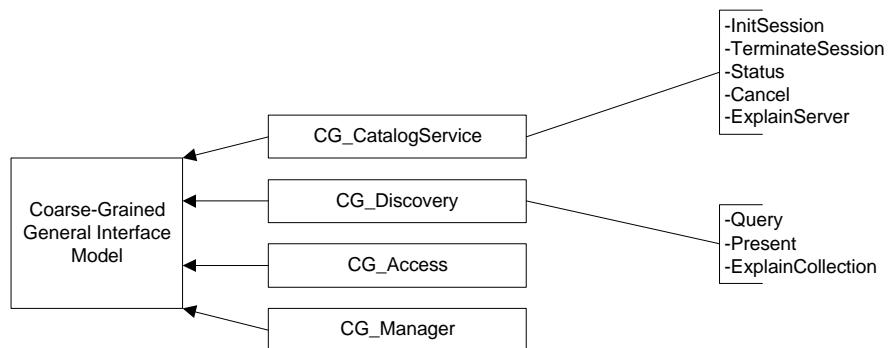


Figure 1 The Coarse-Grained (CG) General Interface Model.

3 Environment and setup of the Catalog Certification program

3.1 Setting up the environment

The Catalog Certification Program has been developed with JDK 1.2.2. The code is in a single file called *catalogCertification.java*. The module OGC_CatalogService has been defined in one *idl* file (*ogc_cat.idl*), the *idl* file contains all definitions, i.e. the message definitions and the required interfaces.

To translate the *idl* file to Java classes, the file *ij.bat* has been written. This batch file uses *idl2java* of VisiBroker for Java 3.4. To compile all Java classes of the module OGC_CatalogService and the *catalogCertification.java* file, the *cj.bat* file has been provided. These two compilation procedures are only necessary if the version of ORB for Java differs from that in VisiBroker for Java 3.4.

Allowable adaptations

Future versions of the testsuite may be adapted to support other Java versions. Furthermore the testsuite may be adapted to support another ORB Vendors. This can imply adaptations to the Java source code and changes to the compilation process.

3.2 Setting up the server

For performing the conformance test, the server should be started and it should write an IOR file. Alternately the IOR file could be created manually by pasting the IOR from somewhere. The IOR file is used by the Catalog Certification Program to obtain a reference to the server. For a distributed query two servers should be started.

Allowable adaptations

The testsuite might be adapted to support also other mechanisms for connection from the testclient to the server to be tested. Possibilities are:

- use of a Naming Service
- use of a Trading Service
- use of Java's JNDI (Java Naming & Directory Interface)
- other or future mechanisms

To support more extensive tests on distributed catalogs, future versions of the testsuite may be adapted with respect to distributed queries.

3.3 Running the client

The Catalog Certification Program functions like a client. To start the Catalog Certification Program, run the *catalogCertification* file.

Allowable adaptations

4 Changes in the testsuite might result in other or additional methods to start the Catalog Certification Program. Furthermore the client might be converted into an applet,

bean **or** **other** **Java** **appearance.**

Description of the test procedures

4.1 Introduction

This section details the tests that have to be executed to test all the operations mentioned in section 2. In order to test the operations, a default query is sent to the catalog server being tested. The database is a known XML (eXtendable Markup Language) file. The retrieved results will be compared with the expected results. A series of queries, applying multiple aspects of the Query Language, will be tested.

To test for a distributed search, two servers are used using both the sample XML file.

An example of the database with metadata is given in appendix A. If a server is not capable of using a database in XML format, the XML file should be ported to a supported format.

Allowable adaptations

- In the future more extensive testing might be added to the testsuite. All fixed values in this chapter are subject to change in future versions or might be made flexible.
- Testing the distributed search might be enhanced in the future.
- The sample metadata database might be extended, adapted, or there might be one or more alternate metadata databases to support other metadata schemas, for example Service Metadata.

4.2 Operations of the CG_CatalogService interface

4.2.1 Testing the InitSession operation

In order to test whether or not the InitSession operation is supported by the server, the Catalog Certification Program follows the following steps:

1. It tries to read the default IOR (Interoperable Object Reference) file, which contains the address of the catalog server;
2. It initializes the ORB (Object Request Broker);
3. It creates a reference to the catalog server;
4. If step 3 has been successful, it creates an InitSession request and invokes this method on the server;
5. It retrieves the identifier, which will be used in the current session.

If any of these steps fail, the InitSession operation is not supported by the server being tested.

Allowable adaptations

If there are alternate methods for the connection of the testclient to the server, as described in the previous chapter, these steps are adapted to reflect that change.

4.2.2 Testing the TerminateSession operation

The Catalog Certification Program tests whether or not the identifier of the session and the reference to the catalog server exist. If these two constraints are met, the TerminateSession method is invoked and a TerminateSession response is sent by the server.

If the status of the TerminateSession response is successful and no exception has been raised, the TerminateSession operation is considered supported. In order to be compliant with the specification, the server should send a successful status when the session has been terminated in a correct way.

4.2.3 Testing the ExplainServer operation

For invoking this method on the catalog server, the *capabilities* field in the request is filled with all the ctAllSupportedRequest. After invoking this method, the server returns a response, which should contain a list of all the capabilities supported by the server. There is no *status* in the ExplainServer response. The Catalog Certification Program works as follows: if the catalog server sends an empty list or if an exception is raised during the process of testing, the ExplainServer operation will be considered not supported. For a full compliance the server should return a list of all capabilities that are supported, the correct capabilities are specified in the OpenGIS Catalog Services Implementation Specification as CG_CapabilityType. At least the OGC_Common QueryLanguage should be supported, as this is required by the Implementation Specification.

Allowable adaptations

Future versions of the testsuite might test more capabilities and might use the retrieved capabilities for further detailed testing. Furthermore more extensive testing of the explainServer request itself could be implemented.

4.3 Operations of the CG_Discovery interface

4.3.1 Query

A query is built by combining a set of parameters in a request. For some parameters there is no viable way to test if their different values are working or they are part of an asynchronous operation. To overcome this problem, their value is fixed in the request (iteratorsize=100, cursor=0, asynchronous=false, maxlevel=2, presentation=full, entrytype=collection). Note that these parameters might be set to other values or made variable as described in the section possible extensions at the beginning of this chapter. The same applies for the settings in the rest of this section.

There is a general approach for testing the operations, although some of them differ slightly from this general approach. The general approach can be described as:

- if a general exception occurs during the *query* call, the capability which is being tested at that moment will be considered *not tested*;
- if a successful status of the response has been retrieved, the capability will be considered *supported*;
- if an unsuccessful status has been retrieved, it depends on the capability if it is considered not supported or not tested.

If, during the initial invocation of the query operation, a general exception is raised or an invalid status is returned, the query operation will be considered as not supported. This test checks the results retrieved in NV format (name-value pairs). If the NV format is not supported by the server, the query method will only be considered supported if no exception has been raised during a call with another format and if the retrieved status is successful. The parameters tested and their possible values are shown in figure 2:

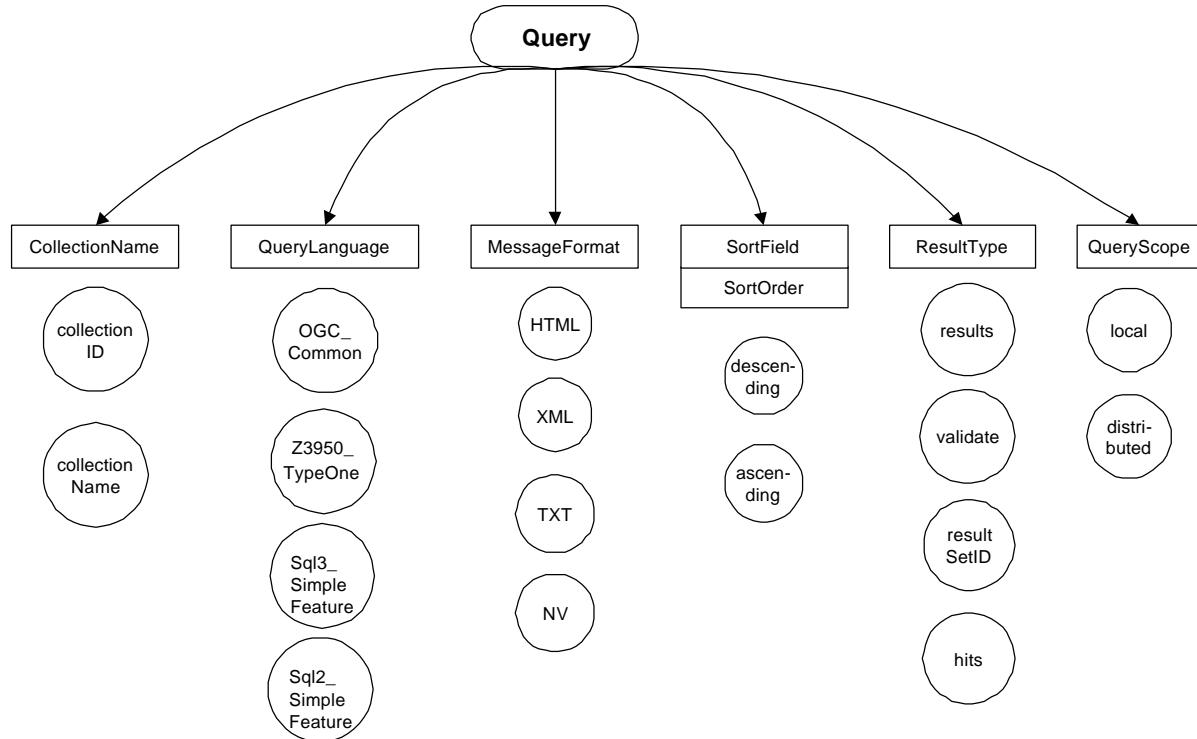


Figure 2 Outline of the parameters (boxes) involved in the Query operation and their possible values (circles).

4.3.1.1 CollectionName

The CollectionName parameter can have two possible values:

- CollectionID: this parameter value combination is considered successful if, after invoking the query method, the status returned is success and there are no problems with reading the string that this field contains;
- CollectionName: the same procedure is followed as with CollectionID.

When the query method is invoked there are several parameters in the request and there is no way to know which parameter is failing if an unsuccessful status is retrieved. If a general exception occurs during the remote call this capability will not be tested, as we do not know for sure which parameter is causing the exception.

4.3.1.2 QueryLanguage

While testing this parameter, the ReturnFormat is set to NV, the QueryScope is local and the ResultType equals results. This means this part of the test depends strongly on the success of the NV format. For the QueryLanguage parameter two values are valid:

- OGC_Common: if an exception is returned, this capability will be considered as not tested. If the response status is unsuccessful, the capability is considered not supported. If none of the above failures occurs, and if the retrieved number of hits is correct, the capability will be considered supported;
- Z3950_TypeOne: the same procedure applies to this language as to the OGC_Common language.

4.3.1.3 *MessageFormat*

In order to test the different values of this parameter, the QueryScope is set to local, the ResultType is results and the QueryLanguage OGC_Common. The series of queries is invoked in combination with the following values of MessageFormat:

- XML: if an exception is raised during invocation of the query method or if the status is unsuccessful, this capability (MessageFormat=XML) will not be tested. Otherwise, the program will test if the return format is the correct one. First of all the program will compare if the *returnFormat* field in the request matches the *retrieveData.encoding* field in the response. After this, the retrieved string will be read and searched for "<?xml". If both conditions are met, the capability will be considered supported;
- HTML: the same applies to this format, only in this case the string is searched for "<html>" and "</html>";
- TXT: the same applies here, but in this case, the string cannot be searched for a specific substring, so it can only be tried to read the string;
- NV: the same applies, it will be tried to read the retrieved NV values, if the number of NV pairs matches the correct number of hits, this capability is considered supported.

Allowable adaptations

Note that the NV format is used to test many aspects of the server now. The testsuite might be enhanced in the future by testing also the XML format more extensive. Note that the structure of the XML format must then be fixed and known by the testsuite, as the current Catalog Specification does not prescribe the XML format. The same applies for the HTML format and the TXT format.

4.3.1.4 *Number of hits retrieved*

To test this parameter, the QueryScope is local, the ResultType equals results, the MessageFormat is NV and the QueryLanguage is set to OGC_Common. If the NV format is not supported by the server, it's not possible to test if the field containing the number of hits in the response is working. The only thing that is tested for is whether or not this field contains the number of hits found in the search which should equal the number of NV pairs (it doesn't have to be the correct number of hits which is to be expected, but it can be). If an exception occurs or if the status is unsuccessful, this capability will not be tested.

4.3.1.5 *SortOrder*

Again, the QueryScope is set to local, the ResultType is results, the MessageFormat equals NV and the language is OGC_Common. The SortOrder has two possible values: ascending or descending. The sort order should always be used in conjunction with a SortField, like in the following example:

```
sortField[0] = OGC_CatalogService.CG_SortField ("title",
OGC_CatalogService.CG_SortOrder.ascending)
```

In this example the Catalog Certification program will test whether or not the results are ordered by title in an ascending way. The possibility of testing this capability depends on the support of the NV format. If an exception occurs, this capability will not be tested. If the status is successful and the NV format is supported, the sorting order will be checked using an array.

4.3.1.6 *ResultType*

There is no easy way to test this parameter, it is difficult to know if a query call is failing due to the query method itself or due to another capability. There are four different values possible for the parameter ResultType, they are:

- Results: if an exception is raised or a status is unsuccessful, the results capability will be considered not supported. In order to be considered supported, the test should work for one or more of the formats. When the Catalog Certification Program invokes the query method for testing the NV format, it will test the results too. If a general exception occurs, the status obtained will be the status obtained with another format. If the status is unsuccessful or if the retrieved number of hits doesn't match the correct number of hits, this capability will be considered unsupported;
- Validate: if an exception is raised or if an unsuccessful status is retrieved, this capability will be considered not supported. Otherwise, the program will test if the *retrieveData* field in the response is empty (this should be the case because only a validation should be performed). Testing if the field is empty is done using the txt format and looking for an empty string "";
- Hits: testing this capability is almost equivalent to the Validate test. In addition, the number of hits is checked;
- ResultSetID: this capability is tested equivalent to the Validate test with one addition: the program also checks whether or not the collectionID field is filled.

4.3.1.7 *QueryScope*

The MessageFormat is fixed as NV, the ResultType as results. The two options for QueryScope are:

- Local: if an exception occurs during the invocation of the query method, this capability will not be tested. If the status is successful and the correct number of hits is retrieved, this capability will be considered as supported;
- Distributed: because of the use of 2 servers during the test of the distributed query, the correct number of hits should be 2 times the correct number of hits using a local scope.

4.3.1.8 *GeographicBoundingBox*

The QueryScope is local, the MessageFormat equals NV and the ResultType results. The queryExpression used is given by:

```
queryExpression=OGC_CatalogService.CG_QueryExpression("intersects(GeographicBoundingBox, envelope(40.71,-24.17,71.26,27.63))", "",  
OGC_CatalogService.CG_QueryLanguage.UGC_Common)
```

The correct number of hits for this query, with respect to the standard metadata database, is 2. If an exception occurs, this capability will be considered not tested. If the status of the response is invalid or if the returned number of NV pairs doesn't match the expected number of hits, this capability is considered not supported.

4.3.1.9 *Invalid query request*

This capability will show if the server is able to detect an invalid query and act in the right way. The MessageFormat is again of the NV type. The query which is invoked is: "abstract lkdd 'river%'". The server has to return an exception or an invalid status in order to consider this capability as supported.

4.3.2 Present

In order to test for this operation of the CG_Discovery interface, some parameters have to be fixed in the request, these are: Iteratorsize(100), Cursor(0), Presentation(full) and ResultSetID ResultSetID sent by the query). Other parameters and their possible values are shown in figure 3.

The general approach which applies to the testing of the capabilities of present is equal to the general approach of the query method (see section 4.3.1). First of all the Catalog Certification Program will set the *Present* capability to not supported but in case one of the calls was successful the capability will be considered supported. If a successful status is retrieved, the last check will be using the NV format, i.e., the program will check if the functionality of this method is working testing the results retrieved with NV format. If the NV format is working the present capability will be shown as supported if the number of hits retrieved is the number of hits expected by the test.

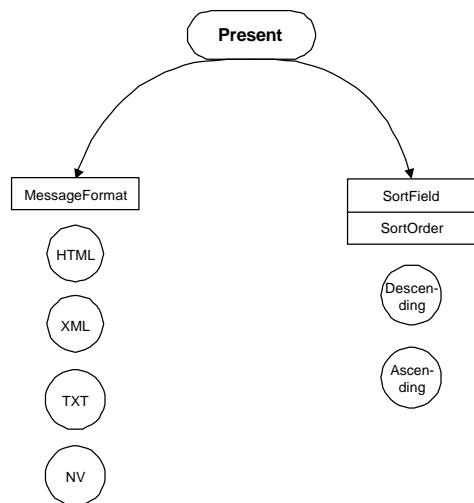


Figure 3 Parameters (in boxes) and their possible values (in circles) of the present operation of the CG_Discovery interface.

4.3.2.1 MessageFormat

The possible values for this parameter are equivalent to those of the query method:

- XML: the same applies here as for the MessageFormat of the query method (section 4.3.1.3);
- HTML: idem;
- TXT: idem;
- NV: idem.

4.3.2.2 Number of hits retrieved

The field containing the number of hits has to have a value equivalent to the number of NV pairs. After invocation of the present method, no exception should be raised and a successful status should be returned. In this case, this capability will be considered supported.

4.3.2.3 SortOrder

There are two options for this parameter: ascending and descending. If the NV format in present is working, this parameter can be tested. The same applies here as to SortOrder of the query method (section 4.3.1.5).

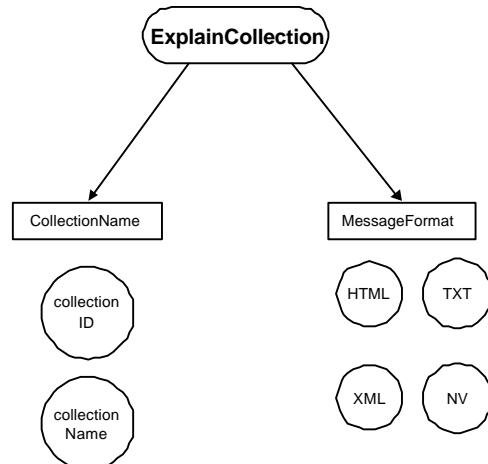


Figure 4 The parameters (in boxes) and their possible values (in circles) of the ExplainCollection method.

4.3.3 ExplainCollection

The ExplainCollection is the last operation of the CG_Discovery interface to be detailed. In order to test this operation, the following parameters are fixed:

- `collectionID.collectionID(" ")`
- `AttributeCategory = OGC_CatalogService.CG_AttributeCategory.both`

There is no viable way to test all the parameters of this method. Both the CG_ExplainCollectionRequest and the CG_ExplainCollectionResponse contain parameters which cannot be tested. The parameters that are tested are shown in figure 4. If a general exception is raised during the remote call the different parameters will be considered not tested. The overall ExplainCollection capability will first be set to not supported, if one of the submethods is supported, the overall capability is also set to supported. Because of the combination of parameters it's difficult to render a probable cause of a failure.

4.3.3.1 CollectionName

As with the query method, the CollectionName can have a value of:

- CollectionID: if it's possible to read the string that this field contains and if the status is successful, this capability will be considered supported;
- CollectionName: the same applies here as to CollectionID.

4.3.3.2 Format

As with Query and Present, the format can be one of the following options:

- XML: in this case there is no *retrievedData* field in the response sent by the server, so the only way to test for this capability is to check whether or not the status is successful;
- HTML: the same applies here as to XML;
- TXT: idem;
- NV: idem.

4.3.3.3 Parameters not tested

The attributeCategory could be tested in the same way the format is, i.e., testing if the status is successful but there is no way to know if the *queriable* or *presentable* attributes are working in a correct way. The dataModel attribute is not tested for because it's impossible to test the information each database contains.

Appendix A: Metadata Database in XML Format

The XML file required for a good operation of the Catalog Certification Program is listed below. For servers that do not support the XML format this file should be ported to a format supported by the server.

Allowable adaptations

This file might be changed or extended to allow more extensive testing in any query language. Furthermore there could be more than one metadata databases in the future. This could be necessary to support for example service metadata.

Contents of the file

```
<?xml version="1.0"?>
<Database>
  <Metadata>
    <Title>Provinces of the Netherlands</Title>
    <Abstract>This dataset contains the 12 provinces which make up the Netherlands</Abstract>
    <GeographicBoundingBox>
      <westBoundLongitude>1.34</westBoundLongitude>
      <eastBoundLongitude>5.00</eastBoundLongitude>
      <northBoundLatitude>70.50</northBoundLatitude>
      <southBoundLatitude>20.70</southBoundLatitude>
    </GeographicBoundingBox>
  </Metadata>
  <Metadata>
    <Title>Communities of the Netherlands</Title>
    <Abstract>This metadata describes the geographic file containing the municipalities of the netherlands</Abstract>
    <GeographicBoundingBox>
      <westBoundLongitude>3.34</westBoundLongitude>
      <eastBoundLongitude>7.22</eastBoundLongitude>
      <northBoundLatitude>53.54</northBoundLatitude>
      <southBoundLatitude>50.74</southBoundLatitude>
    </GeographicBoundingBox>
  </Metadata>
  <Metadata>
    <Title>Railroads of the Netherlands</Title>
    <Abstract>This dataset contains the railroads of the Netherlands</Abstract>
    <GeographicBoundingBox>
      <westBoundLongitude>2.50</westBoundLongitude>
      <eastBoundLongitude>10.00</eastBoundLongitude>
      <northBoundLatitude>25.75</northBoundLatitude>
      <southBoundLatitude>50.75</southBoundLatitude>
    </GeographicBoundingBox>
  </Metadata>
  <Metadata>
    <Title>Railroad stations of the Netherlands</Title>
    <Abstract>This dataset contains the railroad stations of the Netherlands</Abstract>
    <GeographicBoundingBox>
      <westBoundLongitude>1.90</westBoundLongitude>
      <eastBoundLongitude>3.60</eastBoundLongitude>
    </GeographicBoundingBox>
  </Metadata>
</Database>
```

```
<northBoundLatitude>73.55</northBoundLatitude>
<southBoundLatitude>70.75</southBoundLatitude>
</GeographicBoundingBox>
</Metadata>
<Metadata>
<Title>Nuts areas of Europe</Title>
<Abstract>This dataset contains the NUTS-regions of Europe</Abstract>
<GeographicBoundingBox>
<westBoundLongitude>-25.15</westBoundLongitude>
<eastBoundLongitude>25.70</eastBoundLongitude>
<northBoundLatitude>50.25</northBoundLatitude>
<southBoundLatitude>10.65</southBoundLatitude>
</GeographicBoundingBox>
</Metadata>
<Metadata>
<Title>Countries of Europe</Title>
<Abstract>This dataset contains the countries of Europe</Abstract>
<GeographicBoundingBox>
<westBoundLongitude>-24.17</westBoundLongitude>
<eastBoundLongitude>40.71</eastBoundLongitude>
<northBoundLatitude>71.26</northBoundLatitude>
<southBoundLatitude>27.63</southBoundLatitude>
</GeographicBoundingBox>
</Metadata>
<Metadata>
<Title>Rivers of Europe</Title>
<Abstract>This dataset contains the Rivers of Europe</Abstract>
<GeographicBoundingBox>
<westBoundLongitude>-24.17</westBoundLongitude>
<eastBoundLongitude>40.71</eastBoundLongitude>
<northBoundLatitude>71.26</northBoundLatitude>
<southBoundLatitude>27.63</southBoundLatitude>
</GeographicBoundingBox>
</Metadata>
<Metadata>
<Title>Mountains of Italy</Title>
<Abstract>This dataset contains the mountains of Italy</Abstract>
<GeographicBoundingBox>
<westBoundLongitude>2.15</westBoundLongitude>
<eastBoundLongitude>8.70</eastBoundLongitude>
<northBoundLatitude>20.25</northBoundLatitude>
<southBoundLatitude>1.05</southBoundLatitude>
</GeographicBoundingBox>
</Metadata>
</Database>
```

Appendix B: Query file used to test aspects of the OGC Query Language

The file query.txt is used to store queries that are used in the testsuite. Each line in the file contains three items: the number of hits the query should return with the known metadatabase, the query language to use and the query itself. The items are separated by tabs

Allowable adaptations

This file might be changed or extended to allow more extensive testing in any query language. Furthermore, if there are more metadata databases in the future, there need to be also more query files.

Contents of the file

```
1    OGC_Common      abstract like '%rivers%' and title like '%Europe%'  
2    OGC_Common      title like 'c%'  
1    OGC_Common      abstract not like '%dataset%'  
5    OGC_Common      abstract like '%Netherlands%' OR abstract like '%Italy%'  
1    OGC_Common      equals(geographicBoundingBox, envelope(1.9, 3.6, 73.55, 70.75))  
3    OGC_Common      overlaps(geographicBoundingBox, envelope(2.9, 3.6, 73.55, 70.75))  
1    OGC_Common      touches(geographicBoundingBox, envelope(-25.15, 25.7, 0.65, 10.65))  
1    OGC_Common      overlaps(geographicBoundingBox, envelope(2.9, 3.6, 73.55, 70.75)) and  
                    title like 'rivers%'  
2    OGC_Common      equals(geographicBoundingBox, envelope(1.9, 3.6, 73.55, 70.75)) or  
                    abstract not like '%dataset%'  
3    Z3950_TypeOne   Europe[1,4]
```

Appendix C: Batch files

These batchfiles compile the IDL file from the Catalog Implementation Specification to Java, compile the Java source code to Java Classes, and startup the testclient.

Allowable adaptations

These files may be changed to reflect other compilers or ORB Vendors or to reflect future changes in the testsuite.

Contents of the file catalogCertification.bat

```
@echo off

rem -----
rem Program      : catalogCertification.bat
rem Copyright    : OpenGIS Consortium
rem -----
rem Purpose       : Starts the testsuite
rem Notes        : orb jars must be in the classpath
rem -----
rem Authors       : Barend, Geodan SDT b.v. Amsterdam
rem Date         : May 11, 2000
rem -----


REM JAVA 1.1 adapt the JDK directory to the correct place
REM JAVA 1.2 set some properties to choose Visibroker's or
java -classpath .;vbjorb.jar;vbjapp.jar;ogccat.jar -
Dorg.omg.CORBA.ORBClass=com.visigenic.vbroker.orb.ORB -
Dorg.omg.CORBA.ORBSingletonClass=com.visigenic.vbroker.orb.ORB
catalogCertification %1
```

Contents of the file ij.bat

```
@echo off
rem -----
rem Program      : ij.bat
rem Copyright    : OpenGIS Consortium
rem -----
rem Purpose       : Compiles the IDL file ogc_cat.idl to java code
rem -----
rem Authors       : Barend, Geodan SDT b.v. Amsterdam
rem Date         : May 11, 2000
rem -----


rem Remove all generated java files
del OGC_Basic\*.java
del OGC_CatalogService\*.java
del Ogis\*.java

rem copy ..\..\idl\ogc_cat.idl
REM NOTE removed -back_compatible for Java 2 with visibroker
idl2java -no_skel -no_tie -no_examples -no_comments -portable -strict
ogc_cat.idl
rem del ogc_cat.idl
```

Contents of the file cj.bat

```
@echo off
```

```
rem -----
rem Program      : cj.bat
rem Copyright    : OpenGIS Consortium
rem -----
rem Purpose       : Compiles the testsuite (full with option ALL or brief)
rem -----
rem Authors       : Barend, Geodan SDT b.v. Amsterdam
rem Date         : May 11, 2000
rem -----  
  
if A%1 == AALL goto all  
goto rest  
  
:all  
javac -classpath .;vbjorb.jar OGIS\*.java  
javac -classpath .;vbjorb.jar OGC_Basic\*.java  
javac -classpath .;vbjorb.jar OGC_CatalogService\*.java  
del ogccat.jar  
jar -cvf ogccat.jar OGIS OGC_Basic OGC_CatalogService  
  
:rest  
javac -classpath .;vbjorb.jar *.java
```

Appendix D: Source code

This code below contains the source code of the testsuite. It is also available in the OGC Archives as catalogCertification.java.

Allowable adaptations

The file might be adapted to reflect all possible adaptations described in the document above. Furthermore the file might be enhanced by restructuring functions and classes and by adding some comments, or making other enhancements.

Contents of the file

```
// -----
// Program: Java Client for Catalog Services Certification JDK 1.2.2
// -----
// Authors: Laura Diaz, University of Valencia, Spain
//          Bart v.d. Eijnden, Geodan SDT b.v.
//          Barend Gehrels (BSG), Geodan SDT b.v.
// Copyright: OpenGIS Consortium, inc., all rights reserved worldwide
// Purpose: Testsuite for conformance of servers claiming compliancy
//           to the OpenGIS Catalog Implementation Specification
// Document: See also OGC Document 00-027rl for description
// Date: may 11, 2000
// -----
// Revisions: may 27, 2000: (Bart) added query set from file
//             june 9, 2000: (BSG) added query language in file
//             (BSG) added comments
// -----
import java.awt.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;
import java.lang.*;
import java.io.*;
import org.omg.CORBA.ORB;

// -----
// Class: catalogCertification
// -----
// Purpose: Main class, containing certificaton suite
// -----
public class catalogCertification extends JFrame
{
    public Vector QueryVector = new Vector();
    public Vector QueryLanguageVector = new Vector();
    public Vector NumberOfHits = new Vector();

    public class c_capabilities
    {
        private String[] name;
        private int[] available;
        //available = 0 -> not supported
        //available = 1 -> supported
        //available = 2 -> not tested
        public c_capabilities()
        {
            try
            {
                InputStream in = new BufferedInputStream(new
FileInputStream("query.txt"));

```

```

        BufferedReader dis = new BufferedReader(new
InputStreamReader(in));
        String line = null;
        while ( (line = dis.readLine()) != null)
        {
            StringTokenizer split = new StringTokenizer(line,
" ");
            NumberOfHits.addElement(split.nextToken());
            QueryLanguageVector.addElement(split.nextToken());
            QueryVector.addElement(split.nextToken());
        }
        name = new String[(38 + QueryVector.size())];
        available = new int[(38 + QueryVector.size())];
    }
    catch(Exception e)
    {
        ExceptionReport("getquery", e);
    }
    for (int i=0; i < (38+QueryVector.size());i++)
        available[i] = 2;
    writeTableCapabilities();
}

private void writeTableCapabilities()
{
    name[0] = "Init Session";
    name[1] = "Explain Collection";
    name[2] = "Explain Collection collectionID = collectionID";
    name[3] = "Explain Collection collectionID = collectionName";

    name[4] = "Explain Collection Format = XML";
    name[5] = "Explain Collection Format = HTML";
    name[6] = "Explain Collection Format = TXT";
    name[7] = "Explain Collection Format = NV";
    name[8] = "Explain Server";
    name[9] = "Query";
    name[10] = "Query collectionID = collectionID";
    name[11] = "Query collectionID = collectionName";
    name[12] = "Query Expression Language = OGC_Common";
    name[13] = "Query Expression Language = Z3950_TypeOne";
    name[14] = "Query Message Format = XML";
    name[15] = "Query Message Format = HTML";
    name[16] = "Query Message Format = TXT";
    name[17] = "Query Message Format = NV";
    name[18] = "Query number of Hits retrieved";
    name[19] = "Query sortField sortOrder = ascending";
    name[20] = "Query sortField sortOrder = descending";
    name[21] = "Query result Type = results";
    name[22] = "Query result Type = validate";
    name[23] = "Query result Type = hits";
    name[24] = "Query result Type = resultSetID";
    name[25] = "Query Scope = local";
    name[26] = "Query Scope = distributed";
    name[27] = "Query GeographicBoundingBox";
    name[28] = "Invalid query request";
    name[29] = "Present";
    name[30] = "Present Message Format = XML";
    name[31] = "Present Message Format = HTML";
    name[32] = "Present Message Format = TXT";
    name[33] = "Present Message Format = NV";
    name[34] = "Present number of Hits retrieved";
    name[35] = "Present sortField sortOrder = ascending";
    name[36] = "Present sortField sortOrder = descending";
    name[37] = "Terminate Session";
    for (int i=0; i<(QueryVector.size()); ++i)
    {
        name[38+i] = "Query " + QueryVector.elementAt(i);
    }
}

// CORBA support
private org.omg.CORBA.ORB m_Orb;

```

```
private OGC_CatalogService.CG_CatalogServices m_Catalog = null;

// Program logic
private int m_SessionID = 0;
private int m_Counter = 0;
private Vector m_OrderVector = new Vector(); // Global variable for storing
ordered attribute
private c_capabilities tableCapabilities = new c_capabilities();
static private int correctHits = 3; // number of Hits I know is correct
static private int correctHitsCoord = 7; //number of Hits I know is correct in
boundingbox query()

//----- DECLARE_CONTROLS -----
private JPanel queryPanel = new JPanel();
private JPanel capabilitiesPanel = new JPanel();
private JScrollPane capabilityScroll = new JScrollPane();
private JScrollPane queryScroll = new JScrollPane();

private GridBagLayout theGridBagCap = new GridBagLayout();
private GridBagConstraints theConstraintsCap = new GridBagConstraints();

private ButtonGroup languageButtons = new ButtonGroup();
private JRadioButton language1 = new JRadioButton();
private JRadioButton language2 = new JRadioButton();
private JRadioButton language3 = new JRadioButton();

private JScrollPane resultsAreaPane = new JScrollPane();
private JScrollPane diagnosticAreaPane = new JScrollPane();
private JTextField queryText = new JTextField();
static JButton submitButton = new JButton();
private JTextArea resultsArea = new JTextArea(5,41);
private JTextArea diagnosticArea = new JTextArea(1,41);
private JTextField statusText = new JTextField();

private JLabel languageLabel = new JLabel();
private JLabel queryLabel = new JLabel();
private JLabel resultsLabel = new JLabel();
private JLabel diagnosticLabel = new JLabel();
private JLabel capabilitiesLabel = new JLabel();
private JLabel currentStatusLabel = new JLabel();

//-----

public void drawQueryPart()
{
    queryScroll.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);

    queryScroll.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
    getContentPane().add(queryScroll);
    queryScroll.setViewport().add(queryPanel);
    GridBagLayout theGridBag = new GridBagLayout();
    queryPanel.setLayout(theGridBag);
    GridBagConstraints theConstraints = new GridBagConstraints();

    queryPanel.setBackground(new java.awt.Color(204,204,204));
    theConstraints.fill = GridBagConstraints.BOTH;
    theConstraints.gridwidth = GridBagConstraints.REMAINDER;
    theConstraints.anchor = GridBagConstraints.WEST;
    theConstraints.weightx = 1.0;
    languageLabel.setText("Language");
    theGridBag.setConstraints(languageLabel,theConstraints);
    queryPanel.add(languageLabel);
    language1.setText("OGC_Common");
    language1.setSelected(true);
    theConstraints.ipady = 5;
    theGridBag.setConstraints(language1,theConstraints);
    queryPanel.add(language1);
    language2.setText("Z3950 Type One");
    theGridBag.setConstraints(language2,theConstraints);
}
```

```
queryPanel.add(language2);
language3.setText("SQL 3 Simple Features");
theGridBag.setConstraints(language3,theConstraints);
queryPanel.add(language3);
languageButtons.add(language1);
languageButtons.add(language2);
languageButtons.add(language3);
queryLabel.setText("Query");
queryLabel.setAutoscrolls(true);
theGridBag.setConstraints(queryLabel,theConstraints);
queryPanel.add(queryLabel);
theConstraints.gridx = GridBagConstraints.RELATIVE;
queryText.setText("title like '%Europe%'");
//queryText.setText("title like '%Asia%'");
theGridBag.setConstraints(queryText,theConstraints);
queryPanel.add(queryText);
submitButton.setText("Submit");
submitButton.setActionCommand("Submit");
submitButton.setAutoscrolls(true);
submitButton.addActionListener(new submitQuery()));

theConstraints.gridx = GridBagConstraints.REMAINDER;
theConstraints.weightx=0.0;
theGridBag.setConstraints(submitButton,theConstraints);
queryPanel.add(submitButton);
resultsLabel.setText("Results");
resultsLabel.setAutoscrolls(true);
theConstraints.ipady = 10;
theGridBag.setConstraints(resultsLabel,theConstraints);
queryPanel.add(resultsLabel);

resultsAreaPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);

resultsAreaPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
theConstraints.ipady = 250;
theConstraints.fill = GridBagConstraints.NONE;
theConstraints.anchor = GridBagConstraints.CENTER;
theGridBag.setConstraints(resultsAreaPane,theConstraints);
queryPanel.add(resultsAreaPane);
resultsAreaPane.setViewport().add(resultsArea);
resultsArea.setEditable(false);
diagnosticLabel.setText("Diagnostic");
theConstraints.anchor = GridBagConstraints.WEST;
theConstraints.ipady = 10;
theConstraints.fill = GridBagConstraints.BOTH;
theGridBag.setConstraints(diagnosticLabel,theConstraints);
queryPanel.add(diagnosticLabel);

diagnosticAreaPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);

diagnosticAreaPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
theConstraints.ipady = 65;
theConstraints.fill = GridBagConstraints.NONE;
theConstraints.anchor = GridBagConstraints.CENTER;
theGridBag.setConstraints(diagnosticAreaPane,theConstraints);
queryPanel.add(diagnosticAreaPane);
diagnosticAreaPane.setViewport().add(diagnosticArea);
diagnosticArea.setEditable(false);

submitButton.setEnabled(false);

theConstraints.anchor = GridBagConstraints.NORTHWEST;
theConstraints.gridx = GridBagConstraints.REMAINDER;
theConstraints.ipady = 10;
currentStatusLabel.setText("Current Status");
theGridBag.setConstraints(currentStatusLabel,theConstraints);
queryPanel.add(currentStatusLabel);
theConstraints.ipady = 5;
theConstraints.ipadx = 200;
theGridBag.setConstraints(statusText,theConstraints);
```

```
        queryPanel.add(statusText);
        statusText.setEditable(false);
    }

    public void drawCapabilitiesPart()
    {

        capabilitieScroll.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);

        capabilitieScroll.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        getContentPane().add(capabilitieScroll);
        capabilitieScroll.setViewport().add(capabilitiesPanel);
        capabilitiesPanel.setLayout(theGridBagCap);

        capabilitiesPanel.setBackground(new java.awt.Color(204,204,204));
        theConstraintsCap.fill = GridBagConstraints.HORIZONTAL;
        theConstraintsCap.gridwidth = GridBagConstraints.REMAINDER;
        theConstraintsCap.weightx = 1.0;
        theConstraintsCap.ipady = 15;

        JLabel theLab = new JLabel("Capabilities");
        theGridBagCap.setConstraints(theLab,theConstraintsCap);
        capabilitiesPanel.add(theLab);
    }

    public void addCapability(String s, int supported)
    {
        Image yes = (Toolkit.getDefaultToolkit()).getImage("images/checkit.gif");
        Image no = (Toolkit.getDefaultToolkit()).getImage("images/no.gif");
        Image cuestion =
        (Toolkit.getDefaultToolkit()).getImage("images/cuestion.gif");
        ImageIcon cuestionIcon = new ImageIcon(cuestion);
        ImageIcon yesIcon = new ImageIcon(yes);
        ImageIcon noIcon = new ImageIcon(no);

        theConstraintsCap.ipady = 5;
        theConstraintsCap.gridwidth = GridBagConstraints.REMAINDER;

        if (supported == 1)
        {
            JLabel theLab = new JLabel(s,yesIcon,JLabel.LEFT);
            theLab.setVerticalTextPosition(JLabel.CENTER);
            theLab.setHorizontalTextPosition(JLabel.LEFT);
            theLab.setForeground(java.awt.Color.black);
            theGridBagCap.setConstraints(theLab,theConstraintsCap);
            capabilitiesPanel.add(theLab);
            capabilitiesPanel.updateUI();
        }
        else
        {
            if (supported == 0)
            {
                JLabel theLab = new JLabel(s,noIcon,JLabel.LEFT);
                theLab.setVerticalTextPosition(JLabel.CENTER);
                theLab.setHorizontalTextPosition(JLabel.LEFT);
                theLab.setForeground(java.awt.Color.black);
                theGridBagCap.setConstraints(theLab,theConstraintsCap);
                capabilitiesPanel.add(theLab);
                capabilitiesPanel.updateUI();
            }
            else
            {
                JLabel theLab = new JLabel(s,cuestionIcon,JLabel.LEFT);
                theLab.setVerticalTextPosition(JLabel.CENTER);
                theLab.setHorizontalTextPosition(JLabel.LEFT);
                theLab.setForeground(java.awt.Color.black);
                theGridBagCap.setConstraints(theLab,theConstraintsCap);
                capabilitiesPanel.add(theLab);
                capabilitiesPanel.updateUI();
            }
        }
    }
}
```

```
public catalogCertification() // Class Client Constructor
{
    setTitle("OpenGis Catalog Certification");
    setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    getContentPane().setLayout(new GridLayout(1,1,0,0));
    setSize(970,695);
    setVisible(false);

    drawQueryPart();
    drawCapabilitiesPart();

    // --- REGISTER_LISTENERS ---
    SymWindow aSymWindow = new SymWindow();
    this.addWindowListener(aSymWindow);
}

// -----
// Method: catalogCertification, constructor
// -----
// Purpose: Creates a new instance of Client with the given title.
// -----
public catalogCertification(String sTitle)
{
    this();
    setTitle(sTitle);
}

// -----
// Method: setStatusCapability
// -----
// Purpose: fill the tablecapabilities with the correct status in each position
// -----
private void setStatusCapability(int pos, int status)
{
    tableCapabilities.available[pos] = status;
}

// -----
// Method: getStatusCapability
// -----
// Purpose: returns the state of the capability in "pos" position
// -----
private int getStatusCapability(int pos)
{
    return(tableCapabilities.available[pos]);
}

// -----
// Method: baseGet
// -----
// Purpose: fills out the base of each message
// -----
private OGC_CatalogService.CG_Message BaseGet()
{
    return new OGC_CatalogService.CG_Message(m_SessionID,"",
                                             new OGC_CatalogService.CG_RequestID(m_SessionID, m_Counter++),"");
}

// -----
// Method: ExceptionReport
// -----
// Purpose: Show the errors when a exception occurs
// -----
private void ExceptionReport(String s, Exception e)
{
    diagnosticArea.append("Error in " + s + ": " + e.getMessage() + "\n");
    e.printStackTrace();
}
```

```

// -----
// Method:
// -----
// Purpose:
// -----
public boolean initSession(String[] args)
{
    String theior = "";

    setStatus("Initializing Session ");
    try
    {
        // Open InputStream, then open DataInputStream
        InputStream in = new BufferedInputStream(new
FileInputStream("ogcatsrv.ior"));
        BufferedReader dis = new BufferedReader(new InputStreamReader(in));
        // Read IOR from first line (there is only one line)
        theior = dis.readLine();
    }
    catch(Exception e)
    {
        ExceptionReport("getIOR", e);
        // Exception. There is no IOR, we can return because we can not get
objects
        return false;
    }
    // Initialize the ORB and catalog object from ior
    try
    {
        m_Orb = org.omg.CORBA.ORB.init(args, null);

        //narrow : used for typecast, it lets you locate server-side
objects references of a more derived type
        m_Catalog =
OGC_CatalogService.CG_CatalogServicesHelper.narrow(m_Orb.string_to_object(theior));

    }
    catch (Exception e)
    {
        // if narrow fails a CORBA::BAD_PARAM is thrown
        ExceptionReport("connect", e);
        return false;
    }
    if (m_Catalog != null)
    {
        OGC_CatalogService.CG_InitSessionRequest r = new
OGC_CatalogService.CG_InitSessionRequest();
        r.base = BaseGet();
        OGC_CatalogService.CG_InitSessionResponse resp;
        try
        {
            resp = m_Catalog.initSession(r);
            m_SessionID = resp.base.sessionID;
        }
        catch (Exception e)
        {
            ExceptionReport("initSession", e);
            return false;
        }
        return true;
    }
    else
        return false; // m_Catalog == null
}

// -----
// Method:      terminateSession
// -----
// Purpose:    tests the terminateSession request
// -----
public boolean terminateSession()
{
    // Session must be terminated to free resources on the serverside
}

```

```

        if (m_Catalog != null && m_SessionID != 0)
        {
            OGC_CatalogService.CG_TerminateRequest r= new
            OGC_CatalogService.CG_TerminateRequest();
            r.base = BaseGet();

            OGC_CatalogService.CG_TerminateResponse resp;
            try
            {
                setStatus("Terminating Session");
                resp = m_Catalog.terminateSession(r);

                if(resp.status == OGC_CatalogService.CG_Status.success)
                {
                    diagnosticArea.append("Terminate Session Status =
success \n");
                    return(true);
                }
                else
                {
                    if(resp.status ==
OGC_CatalogService.CG_Status.successresultsAvailable)
                    {
                        diagnosticArea.append("Terminate Session
Status = successresultsAvailable \n");
                        return(true);
                    }
                    else
                    {
                        if(resp.status ==
OGC_CatalogService.CG_Status.processingNormal)
                        {
                            diagnosticArea.append("Terminate
Session Status = processingNormal \n");
                            return(false);
                        }
                        else
                        {
                            if (resp.status ==
OGC_CatalogService.CG_Status.processingpausedOrSuspended)
                            {

                                diagnosticArea.append("Terminate Session Status =
processingpausedOrSuspended
\n");
                                return(false);
                            }
                            else
                            {

                                diagnosticArea.append("Terminate Session Status =
failure \n");
                                return(false);
                            }
                        }
                    }
                }
            catch (Exception e)    {
                ExceptionReport("terminateSession" , e);
                return(false);
            }
        }
        else
        {
            return(false);
        }

// -----
// Method:    addString
// -----
// Purpose:   add element to the Global Vector which server for testing the order
// -----
private void addString(String name)
{

```

```

        m_OrderVector.addElement(name);
    }

// -----
// Method:    showStatus
// -----
// Purpose:   Show status in statuswindow or diagnostic area
// -----
private void showStatus(String s, OGC_CatalogService.CG_Status status)
{
    if(status == OGC_CatalogService.CG_Status.success)
        diagnosticArea.append(s +": Status = success \n");
    else
        if(status == OGC_CatalogService.CG_Status.processingNormal)
            diagnosticArea.append(s +": Status = Processing Normal \n");
        else
            if (status ==
OGC_CatalogService.CG_Status.processingpausedOrSuspended)
                diagnosticArea.append(s +": Status = Processing Paused or
Suspended \n");
            else
                diagnosticArea.append(s +": Status = failure \n");
}

// -----
// Method:    getAttribute
// -----
// Purpose:   Obtain the attribute to be ordered for inserting it into the Global Vector
// -----
private void getAttribute(OGIS.NVPair p, int Level, String Attribute)
{
    try
    {
        if (p.value.type().kind() == org.omg.CORBA.TCKind.tk_string)
            if (p.name.indexOf(Attribute)!=-1)
                addString(p.value.extract_string());
            else{}
        else
        {
            OGIS.NVPair[] Value =
OGIS.NVPairSeqHelper.read(p.value.create_input_stream());
            for (int i = 0; i < Value.length; i++)
            {
                getAttribute(Value[i], Level + 1, Attribute);
            }
        }
    }
    catch(Exception e){
        System.out.println("Ignoring, getAttribute :" + e);
    }
}

// -----
// Method:    testOrder
// -----
// Purpose:   Testing if the Order returned is OK
//             the m_OrderVector variable contains the String[] for testing
// -----
public boolean testOrder(String Order)
{
    boolean correct = true;
    int i;

    if(Order.indexOf("ascending")!= -1)
    {
        for (i=0;i<m_OrderVector.size()-1;i++)
        {

if(m_OrderVector.elementAt(i).toString().compareTo(m_OrderVector.elementAt(i+1).to
String())>0)
            correct = false;
}
}
}

```

```

        }
    }
    else // Order=="descending"
    {
        for (i=0;i<m_OrderVector.size()-1;i++)
        {
            if(m_OrderVector.elementAt(i).toString().compareTo(m_OrderVector.elementAt(i+1).toString())<0)
                correct = false;
        }
    /*
    System.out.println("Vector size: " + m_OrderVector.size());
    for (i=0;i<m_OrderVector.size();i++)
        System.out.println("Vector[i] :" +
m_OrderVector.elementAt(i).toString());
 */
    }

    return(correct);
}

// -----
// Method:      NVAdd
// -----
// Purpose:     Extract results with the NV Format and shows them in the
//              Area Results. This function calls itself to add subrecords
// -----
private void NVAdd(OGIS.NVPair p, int Level)
{
    try
    {
        if (p.value.type().kind() == org.omg.CORBA.TCKind.tk_string)
        {
            String Indent = "";
            for (int i = 1; i < Level; i++)
            {
                Indent = Indent + "  ";
            }
            addResults(Indent + p.name + ":" + p.value.extract_string()
+ "\n");
        }
        else
        {
            //System.out.println(" p.value.type != "
org.omg.CORBA.TCKind.tk_string);
            OGIS.NVPair[] Value =
OGIS.NVPairSeqHelper.read(p.value.create_input_stream());
            for (int i = 0; i < Value.length; i++)
            {
                NVAdd(Value[i], Level + 1);
            }
        }
    }
    catch(Exception e){
        System.out.println("Ignoring");
    }
}

// -----
// Method:      addResults
// -----
// Purpose:     Write results in the resultsArea
// -----
private void addResults(String s)
{
    try
    {
        resultsArea.append(s);
    }
    catch(Exception e){
        System.out.println("Ignoring");
    }
}

```

```

        }

// -----
// Method:      getLanguage
// -----
// Returns:     query Language selected depending on the radio button selected
// -----
OGC_CatalogService.CG_QueryLanguage getLanguage()
{
    if (language1.isSelected() == true)
        return(OGC_CatalogService.CG_QueryLanguage.OCG_Common);
    else
        if (language2.isSelected() == true)
            return(OGC_CatalogService.CG_QueryLanguage.Z3950_TypeOne);
        else
            return(OGC_CatalogService.CG_QueryLanguage.SQL3_SimpleFeature);
}

// -----
// Method:      isCorrect
// -----
// Return:      true if status is successfull, else false
// -----
boolean isCorrect(OGC_CatalogService.CG_Status status)
{
    boolean correct = true;

    if((status == OGC_CatalogService.CG_Status.success) || (status ==
OGC_CatalogService.CG_Status.successresultsAvailable))
        return(true);
    else
        return(false);
}

// -----
// Method:      testFormatQuery
// -----
// Purpose:    this function shows the supported Formats (XML, HTML, TXT, NV)
//             in the query function
// -----
void testFormatQuery(OGC_CatalogService.CG_QueryRequest r,
OGC_CatalogService.CG_QueryResponse resp)
{
    try
    {
        if (r.returnFormat == resp.retrievedData.encoding)
        {
            if (resp.retrievedData.encoding ==
OGC_CatalogService.CG_MessageFormat.XML)
            {
                try
                {
                    String s =
resp.retrievedData.payload.extract_string();

                    // Does the "<?xml" string exists?
                    if(s.indexOf("<?xml") != -1)
                        setStatusCapability(14,1); // xml
                    else
                        setStatusCapability(14,0); // xml
                }
                catch( Exception e ){
                    setStatusCapability(14,0);
                }
            }
            else if (resp.retrievedData.encoding ==
OGC_CatalogService.CG_MessageFormat.NV)
            {
                try
                {

```

```
OGIS.NVPair[] Meta =
OGIS.NVPairSeqHelper.read(resp.retrievedData.payload.create_input_stream());
// Add all hits
for (int i = 0; i < Meta.length; i++)
{
    NVAdd(Meta[i], 0);
    getAttribute(Meta[i], 0,"Title");
//Store title in m_OrderVector for testing the order
}

if((Meta.length == correctHits)&&(Meta.length
== resp.hits))
{
    setStatusCapability(18,1);// number
of Hits
    setStatusCapability(17,1);// NV
    setStatusCapability(9,1);//query
}
else
{
    if (Meta.length == correctHits)// correct NV and query, wrong number of Hits
    {
        setStatusCapability(18,0);// number of Hits
        setStatusCapability(17,1);// NV
        setStatusCapability(9,1);//query
    }
}
else
{
    if (Meta.length == resp.hits)
    {
        setStatusCapability(18,1);// number of Hits
        setStatusCapability(17,0);// NV
        setStatusCapability(9,0);//query
    }
}

}
catch (Exception e){
    setStatusCapability(17,0);
}
else
{
    if (resp.retrievedData.encoding ==
OGC_CatalogService.CG_MessageFormat.HTML)
    {
        try
        {
            String s =
resp.retrievedData.payload.extract_string();
            // Does the "<html>" string exists?
            if((s.indexOf("<html>") != -
1)&&(s.indexOf("</html>") != -1))
                setStatusCapability(15,1);// html
            else
                setStatusCapability(15,0);// html
        }
        catch( Exception e ){
            setStatusCapability(15,0);
        }
    }
    else
    if (resp.retrievedData.encoding ==
OGC_CatalogService.CG_MessageFormat.TXT)
    {

```

```

        try
        {

            String s =
resp.retrievedData.payload.extract_string();
                                setStatusCapability(16,1);
}
catch( Exception e ){
                                setStatusCapability(16,0);}

}
else
diagnosticArea.append("Unsupported
encoding: Error in data Format (query)\n");
}
else
{
    if (r.returnFormat == OGC_CatalogService.CG_MessageFormat.TXT)
    {
        diagnosticArea.append("Query: TXT format not supported \n");
        setStatusCapability(16,0);
    }
    else
        if (r.returnFormat ==
OGC_CatalogService.CG_MessageFormat.HTML)
        {
            diagnosticArea.append("Query: HTML format not
supported \n");
            setStatusCapability(15,0);
        }
    else
        if(r.returnFormat ==
OGC_CatalogService.CG_MessageFormat.XML)
        {
            diagnosticArea.append("Query: XML format not
supported \n");
            setStatusCapability(14,0);
        }
    else// NV
    {
        diagnosticArea.append("Query: NV format not
supported \n");
        setStatusCapability(17,0);
    }
}
catch(Exception e){
    ExceptionReport("any", e);
}
}

public void removeVector()
{
    m_OrderVector.removeAllElements();
}

// -----
// Method:      testFormatPresent
// -----
// Purpose:     Test the status of the formats in the present() function
// -----
void testFormatPresent(OGC_CatalogService.CG_PresentRequest r,
OGC_CatalogService.CG_PresentResponse resp)
{
    try
    {
        if (r.returnFormat == resp.retrievedData.encoding)
        {
            if (resp.retrievedData.encoding ==
OGC_CatalogService.CG_MessageFormat.XML)
            {
                try

```

```

    {

        String s =
resp.retrievedData.payload.extract_string();
        if (s.indexOf("<?xml")!= -1)
            setStatusCapability(30,1);

        else
            setStatusCapability(30,0);
    }
    catch( Exception e ){
        setStatusCapability(30,0);
    }
    else if (resp.retrievedData.encoding ==
OGC_CatalogService.CG_MessageFormat.NV)
    {
        try
        {
            OGIS.NVPair[] Meta =
OGIS.NVPairSeqHelper.read(resp.retrievedData.payload.create_input_stream());

            // Add all hits
            for (int i = 0; i < Meta.length; i++)
            {
                getAttribute(Meta[i], 0,"Title"); //
            }

            if((Meta.length == correctHits)&&(Meta.length
== resp.hits))
            {
                setStatusCapability(34,1);// number
                setStatusCapability(33,1);// NV
                setStatusCapability(29,1);//present
            }
            else
            {
                if (Meta.length == correctHits)// correct NV and query, wrong number of Hits
                {
                    setStatusCapability(34,0);///
                    setStatusCapability(33,1);// NV
                    NV
                    setStatusCapability(29,1);//present
                }
                else
                {
                    if (Meta.length == resp.hits)
                    {
                        setStatusCapability(34,1);// number of Hits
                        setStatusCapability(33,0);// NV
                        setStatusCapability(29,0);//present
                    }
                }
            }
        }
        catch (Exception e){
            setStatusCapability(33,0);
        }
    }
    else
        if (resp.retrievedData.encoding ==
OGC_CatalogService.CG_MessageFormat.HTML)
    {
        try

```

```

        {

resp.retrievedData.payload.extract_string();
1)&&(s.indexOf("</html>")!= -1))
{
    }
else
{
    if (resp.retrievedData.encoding ==
OGC_CatalogService.CG_MessageFormat.TXT)
    {
        try
        {
            String s =
                if ((s.indexOf("<html>")!= -
                    setStatusCapability(31,1);
                else
                    setStatusCapability(31,0);
            }
            catch( Exception e ){
                setStatusCapability(31,0);}

        }
        if (resp.retrievedData.encoding ==
    {
        try
        {
            String s =
                if ((s.indexOf("<html>")!= -
                    setStatusCapability(32,1);
                else
                    setStatusCapability(32,0);}

        }
        else
diagnosticArea.append("Unsupported
encoding: Error in data Format (present) \n");
    }
    else
    {
        if (r.returnFormat == OGC_CatalogService.CG_MessageFormat.TXT)
        {
            diagnosticArea.append("Present: TXT format not supported
\n");
            setStatusCapability(32,0);
        }
        else
            if (r.returnFormat ==
OGC_CatalogService.CG_MessageFormat.HTML)
            {
                diagnosticArea.append("Present: HTML format not
supported \n");
                setStatusCapability(31,0);
            }
            else
                if(r.returnFormat ==
OGC_CatalogService.CG_MessageFormat.XML)
                {
                    diagnosticArea.append("Present: XML format
not supported \n");
                    setStatusCapability(30,0);
                }
                else // NV Format
                {
                    diagnosticArea.append("Present: NV format not
supported \n");
                    setStatusCapability(33,0);
                }
            }
        catch(Exception e){
            ExceptionReport("any", e);
        }
    }

// -----

```

```

// Method:      noDataReturned
// -----
// Purpose:    test if retrievedData contains data
// Return:     true  -> if the retrieveData of the Queryresponse contains no Data
//             false -> retrieveData contains data
// -----
boolean noDataReturned(OGC_CatalogService.CG_QueryResponse resp)
{
    try
    {
        /*
         OGIS.NVPair[] Meta =
OGIS.NVPairSeqHelper.read(resp.retrievedData.payload.create_input_stream());
        if (Meta.length != 0)
        {
            System.out.println("Meta.length no es 0 hay datos en retrieve
data...");
            return(false);
        }
        else
        {
            System.out.println("Meta.length es 0... ie, no hay datos en
retrieve data");
            return(true);
        }
        */
        String s = resp.retrievedData.payload.extract_string();

        if (s.compareTo("")==0)
        {
            return(true);
        }
        else
            return(false);
    }
    catch(Exception e)
    {
        System.out.println("Exception en noDataReturned: " + e);
        return (false);
        //return (true) I know the exception is due a null string, so it is
empty...
    }
}

// -----
// Method:      numberHits
// -----
// Purpose:    Reads the number of hits from the NV pairs
// Return:     Number of hits in response
// -----
int numberHits(OGC_CatalogService.CG_QueryResponse resp)
{
    try
    {
        OGIS.NVPair[] Meta =
OGIS.NVPairSeqHelper.read(resp.retrievedData.payload.create_input_stream());
        return(Meta.length);
    }
    catch(Exception e)
    {
        return (-1);
    }
}

// -----
// Method:      queryTest
// -----
// Purpose:    it makes the default query testing all the capabilities
//             of the server creating diferents query request...
// -----
void queryTest(String thequery)

```

```

{
    OGC_CatalogService.CG_QueryLanguage language;
    OGC_CatalogService.CG_QueryRequest r= new
    OGC_CatalogService.CG_QueryRequest();
    r.base = BaseGet();
    language = getLanguage();
    int localHits;
    boolean empty;

    r.queryExpression = new
    OGC_CatalogService.CG_QueryExpression(thequery, "",language);
    r.iteratorSize = 100;
    r.cursor = 0;
    r.asynchronous = false;
    r.maxLevel = 2;
    r.queryScope = OGC_CatalogService.CG_QueryScope.local;

    r.presentation = new OGC_CatalogService.CG_PresentationDescription();

    r.presentation.presentationType(OGC_CatalogService.CG_PredefinedPresentationType.f
ull);

    r.sortField = new OGC_CatalogService.CG_SortField[0];

    r.collectionID = new OGC_CatalogService.CG_CollectionName();
    r.collectionID.collectionID("");
    r.catalogType = OGC_CatalogService.CG_CatalogEntryType.collection;

    r.resultType = OGC_CatalogService.CG_ResultType.results;
    r.returnFormat = OGC_CatalogService.CG_MessageFormat.XML;

    OGC_CatalogService.CG_QueryResponse resp;

    setStatus("Making query");

    try
    {
        resp = m_Catalog.query(r);
        try
        {
            if (isCorrect(resp.status))
            {
                System.out.println("Query
resp.resultsetID.collectionID():"+resp.resultSetID.collectionID());
                setStatusCapability(10,1);
            }
            else
                setStatusCapability(10,0);
        }
        catch(Exception e)
        {
            setStatusCapability(10,0);
        }

        if (isCorrect(resp.status))
        {
            testFormatQuery(r,resp);
            setStatusCapability(21,1); // results
        }
        else
            setStatusCapability(9,0);
    }
    catch(Exception e)
    {
        ExceptionReport("Testing collectionID in Query ", e);
        setStatusCapability(9,0);
        setStatusCapability(21,0); // results
    }

    r.collectionID.collectionName("");
    try
    {
        resp = m_Catalog.query(r);
    }
}

```

```

try
{
    if (isCorrect(resp.status))
    {

        System.out.println("resp.resultsetID.collectionID():" + resp.resultSetID.collectionName());
                setStatusCapability(11,1);
                setStatusCapability(21,1); // results
    }
    else
        setStatusCapability(11,0);
}
catch(Exception e)
{
    setStatusCapability(11,0);
}
catch(Exception e)
{
    setStatusCapability(21,0); // results
    ExceptionReport("Testing collectionName in Query ", e);
}

r.collectionID.collectionID("");
r.returnFormat = OGC_CatalogService.CG_MessageFormat.HTML;
try
{
    resp = m_Catalog.query(r);
    if(isCorrect(resp.status))
    {
        testFormatQuery(r,resp);
        setStatusCapability(21,1); // results
    }
}
catch(Exception e)
{
    ExceptionReport("query", e);
    ExceptionReport("Testing HTML format in Query ", e);
}

r.returnFormat = OGC_CatalogService.CG_MessageFormat.TXT;
try
{
    resp = m_Catalog.query(r);
    if(isCorrect(resp.status))
    {
        testFormatQuery(r,resp);
        setStatusCapability(21,1); // results
    }
}
catch(Exception e)
{
    ExceptionReport("query", e);
    ExceptionReport("Testing TXT Format in Query ", e);
}

r.returnFormat = OGC_CatalogService.CG_MessageFormat.NV;
r.sortField = new OGC_CatalogService.CG_SortField[1];
r.sortField[0] = new OGC_CatalogService.CG_SortField ("title",
OGC_CatalogService.CG_SortOrder ascending);
try
{
    resp = m_Catalog.query(r);

    if(isCorrect(resp.status))
    {
        testFormatQuery(r,resp);
        localHits = numberHits(resp); //in distributed scope this
number has to be greater

        diagnosticArea.append("Local query has returned " +
resp.hits + " hits of " + correctHits +" expected\n");
    }
}

```

```

        if (localHits != correctHits)
        {
            setStatusCapability(25,0); //scope=local
            setStatusCapability(21,0); //results
        }
        else
        {
            setStatusCapability(25,1); //scope=local
            setStatusCapability(21,1); //results
            setStatusCapability(12,1); //OGC_COMMON
        }
        // Testing Order
        if (getStatusCapability(17)==1) // if NV format is working
        {
            if(testOrder("ascending"))
                setStatusCapability(19,1); // ascerding order
            else
                setStatusCapability(19,0);
        }
        else{}
    }
    else
    {
        setStatusCapability(21,0); //results
        setStatusCapability(12,0); //language OGC_COMMON
        setStatusCapability(25,0); //scope=local
    }
}
catch (Exception e)
{
    ExceptionReport("Testing NV Format in Query ", e);
}

// Testing the GeographicBoundingBox query -> correctHitsCoord = 2
thequery ="intersects(GeographicBoundingBox, envelope(-
24.17,40.71,71.26,27.63))"; //west, east, north, south
r.queryExpression = new
OGC_CatalogService.CG_QueryExpression(thequery, "",language);
try
{
    resp = m_Catalog.query(r);

    if(isCorrect(resp.status))
    {
        if (numberHits(resp)!= correctHitsCoord)
        {
            setStatusCapability(27,0); //Query
GeographicBoundingBox
        }
        else
            setStatusCapability(27,1);
    }
    else
    {
        setStatusCapability(27,0);
    }
}
catch (Exception e)
{
    ExceptionReport("Testing GeographicBoundingBox Query ", e);
}

// Testing the invalidate query
thequery = "abstract lkdd 'river%'";
r.queryExpression = new
OGC_CatalogService.CG_QueryExpression(thequery, "",language);
try
{
    resp = m_Catalog.query(r);
}

```

```

        if(isCorrect(resp.status))
            setStatusCapability(28,0); //If the server send a righth
status is not working
        else
            setStatusCapability(28,1);
    }
    catch (Exception e) {
        setStatusCapability(28,1); // Exception because is a invalid query
        ExceptionReport("Testing invalid Query ", e);
    }

    //query with descending order, with the default query
    thequery =getQuery();
    r.queryExpression = new
    OGC_CatalogService.CG_QueryExpression(thequery,"",language);
    r.sortField[0] = new OGC_CatalogService.CG_SortField ("title",
    OGC_CatalogService.CG_SortOrder.descending);
    removeVector(); //remove all the elements for testing now the descending
mode.
    try
    {
        resp = m_Catalog.query(r);
        if(isCorrect(resp.status))
        {
            // Testing Order
            if (getStatusCapability(17)==1)
            {
                OGIS.NVPair[] Meta =
OGIS.NVPairSeqHelper.read(resp.retrievedData.payload.create_input_stream());
                // Add all hits
                for (int i = 0; i < Meta.length; i++)
                {
                    getAttribute(Meta[i], 0,"Title"); // for
testing the order
                }
                if(testOrder("descending"))
                    setStatusCapability(20,1);
                else
                    setStatusCapability(20,0);
            }
            else{}
        }
        else
            setStatusCapability(20,0);
    }
    catch (Exception e) {
        //setStatusCapability(20,0);
        ExceptionReport("Testing descending order in Query ", e);
    }

    r.sortField = new OGC_CatalogService.CG_SortField[0];
    r.queryScope = OGC_CatalogService.CG_QueryScope.distributed;
    try
    {
        resp = m_Catalog.query(r);
        if(isCorrect(resp.status))
        {
            diagnosticArea.append("Distributed query has returned " +
resp.hits + " hits of "+(correctHits*2)+" expected\n");
            if (numberHits(resp)==(correctHits*2)) // 2 distributed
servers for testing
                setStatusCapability(26,1); // scope distributed
            else
                setStatusCapability(26,0); // scope distributed
        }
        else
        {
            setStatusCapability(26,0); // scope distributed
        }
    }
}

```

```

        }

        catch (Exception e)    {
            ExceptionReport("Testing distributed Query ", e);}

        //Testing functionality of differents languages.

        language = OGC_CatalogService.CG_QueryLanguage.Z3950_TypeOne;
        thequery = "Europe[1,4]";
        r.queryExpression = new
        OGC_CatalogService.CG_QueryExpression(thequery,"",language);
        r.queryScope = OGC_CatalogService.CG_QueryScope.local;

        try
        {
            resp = m_Catalog.query(r);
            if(isCorrect(resp.status))
            {
                if (numberHits(resp)==correctHits)
                    setStatusCapability(13,1);//language z39.50

                else
                    setStatusCapability(13,0);//language z39.50

            }
            else
                setStatusCapability(13,0);
        }
        catch(Exception e)
        {
            ExceptionReport("Testing z39.50 language in Query ", e);
        }

        // now we test the resultType: results, hits, validate, resultSetD
        language = getLanguage(); // OGC_Common Language again
        thequery =getQuery();
        r.returnFormat = OGC_CatalogService.CG_MessageFormat.TXT;
        r.queryExpression = new
        OGC_CatalogService.CG_QueryExpression(thequery,"",language);
        r.resultType = OGC_CatalogService.CG_ResultType.validate;
        r.catalogType = OGC_CatalogService.CG_CatalogEntryType.collection;
        try
        {
            resp = m_Catalog.query(r);
            empty = noDataReturned(resp);
            if(isCorrect(resp.status) && (empty))
            {
                setStatusCapability(22,1); //validate
            }
            else
            {
                setStatusCapability(22,0);
            }
        }
        catch (Exception e)
        {
            setStatusCapability(22,0);
            ExceptionReport("Testing resultType=validate", e);
        }

        r.resultType = OGC_CatalogService.CG_ResultType.hits;
        try
        {
            resp = m_Catalog.query(r);
            //Testing that retrieveData == empty
            empty = noDataReturned(resp);
            if(isCorrect(resp.status) && (empty))
            {
                setStatusCapability(23,1); //hits
            }
            else
            {
                setStatusCapability(23,0);
            }
        }
    }
}

```

```

        catch (Exception e)
        {
            setStatusCapability(23,0);
            ExceptionReport("Testing resultType=hits", e);
        }

        r.resultType = OGC_CatalogService.CG_ResultType.resultSetID;
        try
        {
            resp = m_Catalog.query(r);
            OGC_CatalogService.CG_CollectionName resultSetID = new
            OGC_CatalogService.CG_CollectionName();
            resultSetID = resp.resultSetID;

            empty = noDataReturned(resp);
            if(isCorrect(resp.status) && empty)
            {
                setStatusCapability(24,1); //resultSetID
            }
            else
            {
                setStatusCapability(24,0);
            }

            if (getStatusCapability(9)==1)
            {
                diagnosticArea.append("Query general status = success \n");
            }
            else
                diagnosticArea.append("Query general status = failure \n");

            present(resultSetID);
        }
        catch (Exception e)
        {
            setStatusCapability(24,0);
            ExceptionReport("Testing resultType=resultSetID", e);
        }
    }

    // -----
    // Method:      query
    // -----
    // Purpose:     the query when the user click the submit button
    // -----
    void query(String thequery)           //OGC_CatalogService.CG_CollectionName query(String
    thequery)
    {
        OGC_CatalogService.CG_QueryLanguage language;
        OGC_CatalogService.CG_QueryRequest r= new
        OGC_CatalogService.CG_QueryRequest();
        r.base = BaseGet();
        language = getLanguage();
        r.queryExpression = new
        OGC_CatalogService.CG_QueryExpression(thequery,"",language);
        r.resultType = OGC_CatalogService.CG_ResultType.results;
        r.iteratorSize = 100;
        r.cursor = 0;
        r.returnFormat = OGC_CatalogService.CG_MessageFormat.NV; // Name-value
        pairs
        r.presentation = new OGC_CatalogService.CG_PresentationDescription();

        r.presentation.presentationType(OGC_CatalogService.CG_PredefinedPresentationType.f
        ull);
        r.sortField = new OGC_CatalogService.CG_SortField[0];
        r.queryScope = OGC_CatalogService.CG_QueryScope.local;
        r.collectionID = new OGC_CatalogService.CG_CollectionName();
        r.collectionID.collectionID("");
        r.catalogType = OGC_CatalogService.CG_CatalogEntryType.collection;
        r.asynchronous = false;
        r.maxLevel = 2;

        OGC_CatalogService.CG_QueryResponse resp;
    }
}

```

```

try
{
    setStatus("Processing query...");
    resp = m_Catalog.query(r);
    diagnosticArea.append("Query has returned " + resp.hits + " hits.
\n");

    if(isCorrect(resp.status))
    {
        try
        {
            diagnosticArea.append("Query retrievedData Format:
NV \n");
            OGIS.NVPair[] Meta =
OGIS.NVPairSeqHelper.read(resp.retrievedData.payload.create_input_stream());
            // Add all hits
            for (int i = 0; i < Meta.length; i++)
                NVAdd(Meta[i], 0);
        }
        catch ( Exception e ){
            ExceptionReport("Reading data", e);
        }
    }
    showStatus("Query",resp.status);
}
catch (Exception e) {
    ExceptionReport("User query", e);
}
}

// -----
// Method:   explainCollection
// -----
// Purpose:  for retrieving the schema of the DB and test the capabilities
// -----
void explainCollection()
{
    int excp = 0;
    OGC_CatalogService.CG_ExplainCollectionRequest r = new
OGC_CatalogService.CG_ExplainCollectionRequest();
    r.base = BaseGet();

    r.collectionID = new OGC_CatalogService.CG_CollectionName();
    r.collectionID.collectionID("");
    r.returnFormat = OGC_CatalogService.CG_MessageFormat.NV;
    r.attributeCategory = OGC_CatalogService.CG_AttributeCategory.both; // {queriable,
presentable, both}

    OGC_CatalogService.CG_ExplainCollectionResponse resp;

    setStatusCapability(1,0); // If no success in none request it will be not supported
    setStatus("Explain Collection");
    try
    {
        resp = m_Catalog.explainCollection(r);

        if (isCorrect(resp.status))
        {
            setStatusCapability(1,1);
            setStatusCapability(7,1); //Format NV
            try
            {
                System.out.println("Testing collectionID.collectionID :"
+resp.collectionID.collectionID());
                setStatusCapability(2,1); // collectionID
            }
            catch(Exception e)
            {
                setStatusCapability(2,0); // collectionID
            }
        }
        // we can not test the dataModel in a consistent way
    }
}

```

```
        }
    catch(Exception e)
    {
        ExceptionReport("testing collectionID in ExplainCollection", e);
        excp++;
    }

    r.collectionID.collectionName("");
    try
    {
        resp = m_Catalog.explainCollection(r);

        if (isCorrect(resp.status))
        {
            setStatusCapability(1,1);
            try
            {
                System.out.println("Testing
collectionID.collectionName :" +resp.collectionID.collectionName());

                setStatusCapability(3,1); //collectionName
            }
            catch(Exception e)
            {
                setStatusCapability(3,0); //collectionName
                excp++;
            }
        }
    }
    catch(Exception e){
        ExceptionReport("testing collectionName in ExplainCollection", e);
        //setStatusCapability(1,0);
        excp++;
    }

    r.returnFormat = OGC_CatalogService.CG_MessageFormat.XML;
    r.collectionID.collectionID("");
    try
    {
        resp = m_Catalog.explainCollection(r);
        if (isCorrect(resp.status))
        {
            setStatusCapability(1,1);
            setStatusCapability(4,1); //Format XML
        }
    }
    catch(Exception e){
        ExceptionReport("testing XML Format in ExplainCollection", e);
        excp++;
    }

    r.returnFormat = OGC_CatalogService.CG_MessageFormat.HTML;
    try
    {
        resp = m_Catalog.explainCollection(r);
        if (isCorrect(resp.status))
        {
            setStatusCapability(1,1);
            setStatusCapability(5,1); //Format HTML
        }
    }
    catch(Exception e){
        ExceptionReport("testing HTML Format in ExplainCollection", e);
        excp++;
    }

    r.returnFormat = OGC_CatalogService.CG_MessageFormat.TXT;
    try
    {
        resp = m_Catalog.explainCollection(r);
        if (isCorrect(resp.status))
        {
            setStatusCapability(1,1);
```

```

                setStatusCapability(6,1); //Format TXT
            }
        }
        catch(Exception e){
            ExceptionReport("testing TXT Format in ExplainCollection", e);
            excp++;
        }

        diagnosticArea.append(excp +" exceptions occured during the Explain
collection. \n");
        if (getStatusCapability(1)==1)
        {
            diagnosticArea.append("Explain collection general status = success
\n");
        }
        else
            diagnosticArea.append("Explain collection general status = failure
\n");
    }

// -----
// Method:    present
// -----
// Purpose:   present()call with the resultSetID obtained in the query()
//             and test the differents parameters in the present "struct"
// -----
void present(OGC_CatalogService.CG_CollectionName resultID)
{
    int excp = 0;
    OGC_CatalogService.CG_PresentRequest r = new
    OGC_CatalogService.CG_PresentRequest();
    r.base = BaseGet();

    r.iteratorSize = 100;
    r.cursor = 0;
    r.presentation = new OGC_CatalogService.CG_PresentationDescription();
    r.presentation.presentationType(OGC_CatalogService.CG_PredefinedPresentationType.f
ull);
    r.resultSetID = resultID;
    r.sortField = new OGC_CatalogService.CG_SortField[0];

    OGC_CatalogService.CG_PresentResponse resp;

    setStatusCapability(29,0);
    r.returnFormat = OGC_CatalogService.CG_MessageFormat.XML;
    setStatus("Present");
    try
    {
        resp = m_Catalog.present(r);
        if(isCorrect(resp.status))
        {
            setStatusCapability(29,1);
            showStatus("Present",resp.status);
            testFormatPresent(r,resp);
        }
    }
    catch (Exception e)    {
        ExceptionReport("testing XML Format in present", e);
        excp++;
    }

    r.returnFormat = OGC_CatalogService.CG_MessageFormat.HTML;
    try
    {
        resp = m_Catalog.present(r);
        if(isCorrect(resp.status))
        {
            setStatusCapability(29,1);
            testFormatPresent(r,resp);
        }
    }
    catch (Exception e)    {
        ExceptionReport("testing HTML Format in present", e);
    }
}

```

```

        excp++;
    }
    r.returnFormat = OGC_CatalogService.CG_MessageFormat.TXT;
    try
    {
        resp = m_Catalog.present(r);
        if (isCorrect(resp.status))
        {
            setStatusCapability(29,1);
            testFormatPresent(r,resp);
        }
    }
    catch (Exception e)
    {
        ExceptionReport("testing TXT Format in present", e);
        excp++;
    }

    r.returnFormat = OGC_CatalogService.CG_MessageFormat.NV; // Name-value pairs
    r.sortField = new OGC_CatalogService.CG_SortField[1];
    r.sortField[0] = new OGC_CatalogService.CG_SortField ("title",
OGC_CatalogService.CG_SortOrder ascending);
    try
    {
        resp = m_Catalog.present(r);
        diagnosticArea.append("Present has returned " + resp.hits + " hits of "
+correctHits+" expected\n");
        if(isCorrect(resp.status))
        {
            setStatusCapability(29,1);
            removeVector();
            testFormatPresent(r,resp);
            // Testing Order
            if(getStatusCapability(33)==1)
            {
                if(testOrder("ascending"))
                    setStatusCapability(35,1);
                else
                    setStatusCapability(35,0);
            }
        }
    }
    catch (Exception e)
    {
        ExceptionReport("testing NV Format in present", e);
        excp++;
    }

    r.sortField[0] = new OGC_CatalogService.CG_SortField ("title",
OGC_CatalogService.CG_SortOrder.descending);
    try
    {
        resp = m_Catalog.present(r);
        if(isCorrect(resp.status))
        {
            setStatusCapability(29,1);
            // Testing Order
            removeVector();
            if(getStatusCapability(33)==1)
            {
                OGIS.NVPair[] Meta =
OGIS.NVPairSeqHelper.read(resp.retrievedData.payload.create_input_stream());
                // Add all hits
                for (int i = 0; i < Meta.length; i++)
                {
                    getAttribute(Meta[i], 0,"Title"); // for
testing the order
                }
                if(testOrder("descending"))
                    setStatusCapability(36,1);
                else
                    setStatusCapability(36,0);
            }
        }
    }
}

```

```

        }
        catch (Exception e)      {
            excp++;
        }

        diagnosticArea.append(excp + " exception occurred during Present \n");
        if (getStatusCapability(29)==1)
        {
            diagnosticArea.append("Present general status = success \n");
        }
        else
            diagnosticArea.append("Present general status = failure \n");
    }

// -----
// Method:   explainServer
// -----
// Purpose:  Make explainServer request
// -----
void explainServer()
{
    OGC_CatalogService.CG_ExplainServerRequest r= new
    OGC_CatalogService.CG_ExplainServerRequest();
    OGC_CatalogService.CG_ExplainServerResponse resp;

    r.base = BaseGet();
    r.capabilities = new OGC_CatalogService.CG_CapabilityType[9];
    r.capabilities[0] = OGC_CatalogService.CG_CapabilityType.ctAllSupportedRequest;
    //Asking for all the capabilities
    r.capabilities[1] = OGC_CatalogService.CG_CapabilityType.ctDefaults;
    r.capabilities[2] = OGC_CatalogService.CG_CapabilityType.ctDefaultTimeOut;
    r.capabilities[3] = OGC_CatalogService.CG_CapabilityType.ctExplain;
    r.capabilities[4] = OGC_CatalogService.CG_CapabilityType.ctMessaging;
    r.capabilities[5] = OGC_CatalogService.CG_CapabilityType.ctQuery;
    r.capabilities[6] = OGC_CatalogService.CG_CapabilityType.ctSession;
    r.capabilities[7] = OGC_CatalogService.CG_CapabilityType.ctSoftwareInformation;
    r.capabilities[8] = OGC_CatalogService.CG_CapabilityType.ctSupportedCollections;

    setStatus("Explain Server");
    try
    {
        resp = m_Catalog.explainServer(r);

        diagnosticArea.append("Number of capabilities returned by ExplainServer: "
+ resp.capabilities.length + "\n");

        if (resp.capabilities.length >0)//we can not test isCorrect(resp.status)-
>no status in the explainServer response
        {
            setStatusCapability(8,1);
        }
        else
            setStatusCapability(8,0);
    }
    catch (Exception e)
    {
        setStatusCapability(8,0);
        ExceptionReport("seeCapabilities",e);
    }
}

// -----
// Method:   Submit
// -----
// Purpose:  Make a query() call when submit button has been pressed
// -----
public void Submit()
{
    String s;
    String args[] = new String[1];
}

```

```

        args[0]="";
        boolean succ;

        resultsArea.setText("");
        diagnosticArea.setText("");

        setStatus("Processing...");
        if (initSession(args))
        {
            diagnosticArea.append("Init Session status = success\n");
            s = getQuery();
            query(s);
            succ = terminateSession();
            setStatus("Waiting for a new query ...");
        }
        else
            diagnosticArea.append("Init Session status = failure\n");
    }

// -----
// Method:      getQuery
// -----
// Purpose:     reads the queryText field
// -----
public String getQuery()
{
    return(queryText.getText());
}

public void showCapabilities()
{
    int i;

    for (i=0;i<(38+QueryVector.size());i++)
        addCapability(tableCapabilities.name[i],tableCapabilities.available[i]);
}

// -----
// Method:      setStatus
// -----
// Purpose:     Show in the StatusText field which is the status indicated
//              by the String
// -----
public void setStatus(String s)
{
    statusText.setText(s);
}

// -----
// Method:      queryList
// -----
// Purpose:     Section which checks list of queries from file query.txt
// -----
public void queryList()
{
    setStatus("Queries");
    //queryText.setText((String) QueryVector.elementAt(0));
    OGC_CatalogService.CG_QueryRequest r2= new OGC_CatalogService.CG_QueryRequest();
    r2.base = BaseGet();
    int localHits;
    boolean empty;
    for (int i=0; i<QueryVector.size(); ++i)
    {
        try
        {
            String StringLanguage = (String)QueryLanguageVector.elementAt(i);
            OGC_CatalogService.CG_QueryLanguage language2;
            if (StringLanguage.compareTo("OGC_Common") == 0)
                language2 = OGC_CatalogService.CG_QueryLanguage.UGC_Common;
            else if (StringLanguage.compareTo("Z3950_TypeOne") == 0)

```

```

        language2 =
OGC_CatalogService.CG_QueryLanguage.Z3950_TypeOne;
        else throw new Exception("Bad query language " + StringLanguage);

        r2.queryExpression = new
OGC_CatalogService.CG_QueryExpression((String)QueryVector.elementAt(i), "",language2);
        r2.iteratorSize = 100;
        r2.cursor = 0;
        r2.asynchronous = false;
        r2.maxLevel = 2;
        r2.queryScope = OGC_CatalogService.CG_QueryScope.local;
        r2.presentation = new
OGC_CatalogService.CG_PresentationDescription();

        r2.presentation.presentationType(OGC_CatalogService.CG_PredefinedPresentationType.
full);
        r2.sortField = new OGC_CatalogService.CG_SortField[0];
        r2.collectionID = new OGC_CatalogService.CG_CollectionName();
        r2.collectionID.collectionID("");
        r2.catalogType = OGC_CatalogService.CG_CatalogEntryType.collection;

        r2.resultType = OGC_CatalogService.CG_ResultType.results;
        r2.returnFormat = OGC_CatalogService.CG_MessageFormat.NV;
        OGC_CatalogService.CG_QueryResponse resp2;

        resp2 = m_Catalog.query(r2);
        localHits = numberHits(resp2);
        int correctHits =
Integer.parseInt((String)NumberofHits.elementAt(i));
        try
        {
            if (localHits != correctHits)
            {
                setStatusCapability(38+i,0);
            }
            else
                setStatusCapability(38+i,1);
        }
        catch(Exception e)
        {
            setStatusCapability(38+i,0);
        }
    }
    catch(Exception e)
    {
        setStatusCapability(38+i,0);
        e.printStackTrace();
    }
}
}

// -----
// Method:    main
// -----
// Purpose:   Main class starting up the testsuite
// -----
static public void main(String args[])
{
    String theQuery = new String();
    boolean succ;
    OGC_CatalogService.CG_CollectionName resultID;

    try
    {
        // Add the following code if you want the Look and Feel of the native
        // system.
        /*try
        {UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());}
        catch (Exception e) {}*/
        catalogCertification C = new catalogCertification();
        C.setVisible(true);

        // --- Certification code ---

```

```

        if (C.initSession(args))
        {
            C.diagnosticArea.append("Init Session status = success\n");
            C.setStatusCapability(0,1);
            C.explainCollection();
            C.explainServer();
            theQuery = C.getQuery();
            C.queryTest(theQuery);
            C.queryList();
            if (C.terminateSession())
            {
                C.setStatusCapability(37,1);
            }
            else
            {
                C.setStatusCapability(37,0);
            }
        }
        else
        {
            C.diagnosticArea.append("Init Session status = failure\n");
            C.setStatusCapability(0,0);
        }

        C.showCapabilities();

        //Query enable, Certification process finished
        submitButton.setEnabled(true);
        C.setStatus("Waiting for a new query...");

    }
    catch (Throwable t)
    {
        t.printStackTrace();
        //Ensure the application exits with an error condition.
        System.exit(1);
    }
} //-----End main()

// Used by addNotify
boolean frameSizeAdjusted = false;
/**
 * Notifies this component that it has been added to a container
 * This method should be called by <code>Container.add</code>, and
 * not by user code directly.
 * Overridden here to adjust the size of the frame if needed.
 * @see java.awt.Container#removeNotify
 */
public void addNotify()
{
    // Record the size of the window prior to calling parents addNotify.
    Dimension size = getSize();

    super.addNotify();

    if (frameSizeAdjusted)
        return;
    frameSizeAdjusted = true;

    // Adjust size of frame according to the insets and menu bar
    JMenuBar menuBar = getRootPane().getJMenuBar();
    int menuBarHeight = 0;
    if (menuBar != null)
        menuBarHeight = menuBar.getPreferredSize().height;
    Insets insets = getInsets();
    setSize(insets.left + insets.right + size.width, insets.top + insets.bottom
+ size.height + menuBarHeight);
}

void exitApplication()
{
    try

```

```
{  
    int reply = JOptionPane.showConfirmDialog(this,"Do you really want to  
exit?","OGC Catalog Certification." ,  
                                              JOptionPane.YES_NO_OPTION,  
                                              JOptionPane.QUESTION_MESSAGE);  
    if (reply == JOptionPane.YES_OPTION)  
    {  
        this.setVisible(false);      // hide the Frame  
        this.dispose();            // free the system resources  
        System.exit(0);           // close the application  
    }  
}  
catch (Exception e) {}  
}  
  
class SymWindow extends WindowAdapter  
{  
    public void windowClosing(WindowEvent event)  
    {  
        Object object = event.getSource();  
        if (object == catalogCertification.this)  
            Client_windowClosing(event);  
    }  
}  
  
void Client_windowClosing(java.awt.event.WindowEvent event)  
{  
    // to do: code goes here.  
    Client_windowClosing_Interaction1(event);  
}  
  
void Client_windowClosing_Interaction1(WindowEvent event)  
{  
    try  
    {  
        this.exitApplication();  
    }  
    catch (Exception e) {}  
}  
  
class submitQuery implements ActionListener  
{  
    public void actionPerformed(ActionEvent event)  
    {  
        Object object = event.getSource();  
        Submit();  
    }  
}  
}
```