Open Geospatial Consortium Inc.

Date: 2010-02-15

Reference number of this document: OGC 10-070

Version: **1.0.0**

Category: OpenGIS® Implementation Standard

Editor: Peter Schut

OpenGIS® Table Joining Service Implementation Standard

Warning

This document is not an OGC Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OpenGIS® Standard
Document subtype:	
Document stage:	Draft proposed version 1.0
Document language:	English

Contents

Page

i. Preface

Suggested additions, improvements, and comments on this specification are welcome and encouraged. Such suggestions may be submitted by email.

ii. Document terms and definitions

This document uses the standard terms defined in Subclause 5.3 of [OGC 06-121r3], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

iii. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization

iv. Revision history

Date	Release	Editor	Primary clauses modified	Description
	1.0.0	P. Schut	all	Preliminary draft of version 1.0

v. Changes to the OGC Abstract Specification

The OpenGIS® Abstract Specification does not require changes to accommodate the technical contents of this document.

vi. Future work

Improvements in this document are desirable to standardize the way that classification information is encoded (i.e. the XML structure of the ClassificationURL parameter of the JoinData request).

Foreword

This specification is the result of work first undertaken to support the Canadian Geospatial Data Infrastructure (CGDI), and in particular the Canadian Soil Information System (CanSIS), and the National Forest Information Service (NFIS). The standard was implemented as a prototype in 2002 by Agriculture and Agri-Food Canada (AAFC). Version 0.8.1 was approved by OGC in 2004 as a set of two discussion documents (04-010: Geolinked Data Access Service (GDAS), and 04-011: Geolinking Service (GLS)) which towards the end of 2005 were found to be among the top 10 downloads from the OGC website. An Interoperability Experiment was subsequently initiated to further refine the concept and associated interfaces, and resulted in the production of version 0.10.2. The final report of the Interoperability Experiment recommended a number of additional improvements to the specifications including that they be merged into a single interface specification and was published as version 0.12.0 (08-006: Geographic Linkage Service (GLS)) as an OGC Request for Comments (RFC). This version incorporates recommendations received as part of the RFC and replaces those earlier documents.

This document includes three annexes; Annexes A and B are normative, and Annex C is informative.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The OGC shall not be held responsible for identifying any or all such patent rights.

Introduction

This document is the specification for a Table Joining Service (TJS). It defines a simple way to describe and exchange tabular data that contains information about geographic objects.

Almost all corporate databases contain some kind of geographic identifier, regardless of whether or not the database is housed in a computing environment that supports a Geographic Information System (GIS). Geographic identifiers can include postal codes, municipality names, telephone area codes, or more special purpose identifiers such as school districts. Geospatial linking technology allows this corporate data to be found and used for mapping or spatial analysis.

The geographic identifiers used in corporate databases usually reference a spatial framework. A spatial framework in this context is a partitioning of the surface of the earth into a set of management units. Municipalities, postal codes, telephone area codes, ecoregions, and watersheds are all examples of spatial frameworks. These frameworks all have one thing in common – they contain a descriptor that can be used to uniquely identify any individual management unit –and that same identifier is found in the corporate database.

Here are some typical database contents along with their spatial frameworks:

- Sales by retail outlet or by municipality
- Insurance payments by postal code
- Telephone numbers by area code
- Farms by Census Agricultural Region
- Students by school district

TJS offers a way to expose this corporate data to other computers, so that it can be found and accessed, and a way to merge that data with the spatial data that describes the framework, in order to enable mapping or geospatial analysis.

Tabular data to be exchanged via TJS must include a geographic identifier. The geographic identifier refers to a spatial feature found in a separate geospatial data set. In order to join tabular data to another dataset, both datasets must contain the same geographic identifier (i.e. key field). An example of such tabular data is a collection of population counts by city. The table includes the city name, but does not include any other geographic identifier. The city names can be used to join the population data to a layer in a GIS that contains the spatial coordinates for each city, in order to map the information or perform some sort of geospatial analysis.

TJS provides a simple standardized way to exchange attribute information that applies to a well-known geospatial dataset (a Framework dataset). Attribute information delivered from a TJS can be used in a variety of ways, including use by models to perform calculations, or visualization as a web map.

The advantages of the TJS specification is that it allows organizations to house their corporate data on systems that are optimized for the management of that data, and yet to take advantage of GIS technology to examine and analyze that data. In addition, it is

- Simple to implement
- Easy to use
- Lightweight
- Highly scalable
- Multi-purpose

TJS allows corporate data to be maintained closest to source, and yet allows the latest data to be obtained when analysis is being performed, regardless of whether or not the geospatial system can make a direct connection to the corporate data management system. Exposing the data through a TJS allows the corporate data managers to change their underlying database design and security safeguards without compromising access to data that should be accessible by other systems. In effect, TJS supports both distributed data management, as well as the distributed processing of geospatial data.

TJS includes two related sets of operations. Tabular data is served up to other computers on the network by implementing the GetData and related operations. The response to a GetData operation is an XML file, in a format known as GDAS (Geographic Data Attribute Set). Tabular data is ingested at some other node on the network by a TJS configured to support the JoinData and related operations. These operations allow a computer to join tabular data in GDAS format into a local spatial framework dataset. This local dataset can then be used to provide maps via a WMS, or served up through some other mechanism that provides access to the joined data.

OpenGIS® Table Joining Service Implementation Standard

1 Scope

This specification applies to the creation and use of the Table Joining Service (TJS). It includes the definition of all TJS requests and responses, including the specification of the Geographic Data Attribute Set (GDAS) encoding format using eXtensible Markup Language [XML 1.0].

This specification does not address the archival, cataloging, discovery or retrieval of GDAS or other TJS XML documents.

2 Compliance

Compliance with this standard shall be checked using all the relevant tests specified in Annex A (normative).

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

ISO 19105:2000, *Geographic information — Conformance and Testing*

OGC 06-121r3, *OpenGIS® Web Services Common Specification*

NOTE This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

In addition to this document, this standard includes several normative XML Schema Document files as specified in Annex B.

4 Terms and definitions

For the purposes of this standard, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 06-121r3] shall apply. In addition, the following terms and definitions apply.

4.1
attribute
a set of values that describe some aspect of a **spatial framework**. Population, temperature, and income are all examples of attributes that could apply to a **spatial framework**.

4.2
capabilities
service-level metadata describing the **operations** and supporting content available from a **server**

4.3
client
software component that can invoke an **operation** from a **server**

4.4
geographic identifier
unique reference for a spatial object found in a **spatial framework** dataset

4.5
tabular data
a table of data consisting of a collection of **attributes** each with a **geographic identifier** that enables those **attributes** to be joined to a **spatial framework**.

4.6
operation
a transformation or query that a **server** may be requested to execute

4.7
request
invocation of an **operation** by a **client**

4.8
response
result of an **operation** returned from a **server** to a **client**

4.9
server
physical instantiation of a **service**

4.10
service
a suite of functionality that is provided by an entity through a standardized set of **operations** defined by **interfaces**

4.11
spatial framework
a GIS representation, either point, line, or polygon, of any collection of physical or conceptual geographic objects.   Municipalities, postal code areas, telephone area codes, ecoregions, watersheds, road segments, fire stations, and lighthouses are all examples of spatial frameworks

5 Conventions

5.1 Abbreviated terms

Many of the abbreviated terms listed in Subclause 5.1 of the OWS Common Implementation Specification [OGC 06-121r3] apply to this document, plus the following abbreviated terms.

5.2 Used parts of other documents

This document uses significant parts of the OWS Common Implementation Specification [OGC 06-121r3]. To reduce the need to refer to that document, this document copies some of those parts with small modifications. To indicate those parts to readers of this document, the largely copied parts are shown with a light grey background (15%).

5.3 Platform-neutral and platform-specific standards

As specified in Clause 10 of OGC Abstract Specification Topic 12 “OpenGIS Service Architecture” (which contains ISO 19119), this document includes both Distributed Computing Platform-neutral and platform-specific standards. This document first specifies each operation request and response in platform-neutral fashion. This is done using a table for each data structure, which lists and defines the parameters and other data structures contained.

The specified platform-neutral data could be encoded in many alternative ways, each appropriate to one or more specific DCPs. This document now specifies encoding appropriate for use of HTTP GET transfer of operations requests (using KVP encoding), and for use of HTTP POST transfer of operations requests using XML encoding. However, the same operation requests and responses (and other data) could be encoded for other specific computing platforms, including SOAP/WSDL.

5.4 Data dictionary tables

The data dictionary is specified herein in a series of tables. The contents of the columns in these tables are described in Table 1.

Table 1 — Contents of data dictionary tables

Column title	Column contents
Name (left column)	Specifies the name of the parameter or association (or data structure). This name uses the XML encoding capitalization specified in Subclause 11.6.2 of [OGC 06-121r3]. Some names in the tables may appear to contain spaces, but no names contain spaces.
Definition (second column)	Specifies the definition of this parameter (omitting un-necessary words such as “a”, “the”, and “is”). If the parameter value is the identifier of something, not a description or definition, the definition of this parameter should read something like “Identifier of TBD”.
Data type and value (third column) or Data type (if are no second items are included in rows of table)	Normally contains two items: The mandatory first item is often the data type used for this parameter, using data types appropriate in a UML model, in which this parameter is a named attribute of a UML class. Alternately, the first item can identify the data structure (or class) referenced by this association, and references a separate table used to specify the contents of that class (or data structure). The optional second item in the third column of each table should indicate the source of values for this parameter, the alternative values, or other value information, unless the values are quite clear from other listed information.
Multiplicity and use (right or fourth column) or Multiplicity (if are no second items are included in rows of table)	Normally contains two items: The mandatory first item specifies the multiplicity and optionality of this parameter in this data structure, either “One (mandatory)”, “One or more (mandatory)”, “Zero or one (optional)”, or “Zero or more (optional)”. The second item in the right column of each table should specify how any multiplicity other than “One (mandatory)” shall be used. If that parameter is optional, under what condition(s) shall that parameter be included or not included? If that parameter can be repeated, for what is that parameter repeated?

The data type of many parameters, in the third table column, is specified as “Character String type, not empty”. In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

The contents of these data dictionary tables are normative, including any table footnotes.

6 Table Joining Service overview

The Table Joining Service enables the publication of tabular information about geographic features, and the joining of these tables to their geographic features so that the information can be mapped or processed in a geographic information system.

The TJS operations have similarities to many other OGC Web Services, including the WMS, WFS, and WCS. The interface aspects that are common with other OWSs are specified in the OpenGIS® Web Services Common Implementation Specification [OGC 06-121r3]. Some of these common aspects are normatively referenced herein, instead of being repeated in this standard.

GDAS data encoding is described in clause 7, while clauses 9 through 15 of this document describe the TJS service operations.

6.1 Operations

Like all OGC services, the Table Joining Service interface specifies a service discovery operation:

a) GetCapabilities (required implementation by all servers) – This operation allows a client to request and receive back service metadata (or Capabilities) documents that describe the abilities of the specific server implementation. This operation also supports negotiation of the standard version being used for client-server interactions.

In addition, TJS specifies two distinct sets of operations that can be requested by a client and performed by a server: *data access* operations and *data joining* operations.

6.1.1 Data access operations

Operations related to encoding and delivering geo-tabular data in a format that can be ingested by other services include:

b) DescribeFrameworks – This operation allows a client to obtain a list of the spatial frameworks for which geo-tabular data is available from the server.

c) DescribeDatasets – This operation allows a client to obtain general descriptions of the attribute data tables which are available from the server.

d) DescribeData – This operation allows a client to obtain a list describing the specific data contents of the attribute data tables (i.e. the attributes) which are available from the server.

e) GetData – This operation allows a client to obtain a specific set of geo-tabular data.

If a server supports data access, then all of the data access operations are mandatory.

6.1.2 Data joining operations

Operations related to ingesting geo-tabular data and joining it to its spatial framework (i.e data joining) include:

f) DescribeJoinAbilities – This operation allows a client to obtain the list of spatial frameworks to which the server can join geo-tabular data, and the forms of output products supported that are supported by the server.

g) DescribeKey – This operation allows a client to obtain the list of geographic identifiers for a spatial framework supported by the server.

h) JoinData – This operation allows a client to request the joining of a specified geo-tabular dataset to its spatial framework and receive references to the products of that join.

If a server supports data joining, then all of the data joining operations are mandatory.

6.2 XML encoding of tabular information

The data access operations (b,c,d, and e above) return selected parts or all of an XML encoded data file, in a format known as Geographic Data Attribute Set (GDAS). The GetData operation returns an entire GDAS file including attribute data and its associated metadata, while DescribeFrameworks, DescribeDatasets, and DescribeData each return selectively larger portions of the metadata for the geographic attributes that can be obtained from the TJS server. At the other end of the data exchange process, the JoinData operation accepts attribute data in GDAS format.

The purpose of the Geographic Data Attribute Set (GDAS) encoding format is to package attribute information and its associated metadata so that it can be transmitted across a network. The GDAS format offers the following benefits:

- It is simple to understand and implement. The core of the encoding has a similar structure to the encoding for an HTML Table. This makes the file easy to generate from corporate database technology, and easy to manipulate into other encodings such as HTML using XSLT. It also facilitates debugging.
- It is lightweight. The encoding was designed to be as light-weight as possible, in order to minimize network transmission times for large datasets without introducing the overhead of data compression.
- It is highly scaleable. The encoding can carry multiple attributes simultaneously, with minimal additional overhead.
- It is multi-purpose. The encoding can represent all types of attribute data, allowing a generic interface to handle all types of attribute data.
- It is unambiguous. The encoding uniquely references the framework dataset to which the attributes apply.
- It contains substantial metadata, which makes it possible to automatically generate tables, charts, documentation, maps, and legends from the same data stream.
- It supports the identification of null values in the dataset to facilitate their exclusion from calculations and legends
- It includes attributes to support the joining of tabular data to geometry in an N:1 or N:N fashion.
- It specifies an attribute typology that facilitates the automated determination of appropriate forms of mapping and charting.

The GDAS format is designed to support simple as well as rich and complicated attribute databases that may not always be easy to interpret. The metadata included in the encoding is designed to ensure that the user knows exactly what the content of the dataset is as well as which spatial framework it references, and has easy access to any associated documentation. GDAS is only ever produced in its entirety as the response to a TJS GetData request, but portions of GDAS underlie all TJS operation requests and responses.

6.3 Uses of GDAS and TJS

GDAS was designed to support the joining of tabular data to its spatial framework, for the purpose of populating OGC Web Map Service (WMS) or Web Feature Service (WFS) servers with additional attribute information. TJS can be configured to convert attribute and related information from GDAS format into the databases and configuration files required to display maps via a WMS, or to serve up attributes included with their geographic coordinates in Geographic Markup Language (GML) via a WFS.

GDAS can also be used as the response to a WMS GetFeatureInfo request. For example, once GDAS data has been sent and joined to a layer supporting a WMS server, this server can response a WMS GetFeatureInfo request with the requested subset of records in GDAS format. To accomplish this, GetFeatureInfo must specify "INFO_FORMAT=text/xml; subtype=gdas/1.0" as part of the WMS GetFeatureInfo request.

The utility of TJS goes beyond the simple extraction and joining of database content, since data in GDAS format can be readily manipulated as part of a service chain. Services such as the OGC's Web Processing Service (WPS) can be used to perform calculations, comparisons, or other analysis on attribute data in GDAS format before it is accessed by a JoinData operation. Since GDAS is quite a simple encoding, it can easily be used as an input or generated as an output from any spatial model. Furthermore, the incorporation of an XSL reference allows the same GDAS file to be used to generate web pages as well as to provide the input to models or automated mapping services.

Although TJS was originally intended be configured with the tabular data housed on one server and the spatial framework data housed on another server, the Interoperability Experiment conducted by OGC in 2007 demonstrated that in some circumstances TJS is useful even when the attributes and spatial frameworks are housed on the same server. This is especially true for very large datasets, such as Census data, where user-triggered automated mapping can be deployed through the implementation of TJS. The data access operations are designed to support access to very large volumes of metadata without overloading the network connection.

Note that while GDAS files can be served up dynamically via a TJS, for performance reasons it may be preferable to store the content of specific GetData responses and host them as static files. Deployment of such static GDAS files are also a valid way to provide canned or pre-processed data interpretations without the effort and expense of supporting a full-blown TJS implementation.

6.4 Definition of types of attributes

Critical for the automation of mapping tasks using TJS is the correct identification of the type of attribute that is being described and transferred. GDAS supports four different types of attributes, which cover the entire spectrum of mapping types (see Table 18 — GDAS data encoding: Values data structure).

This concept is important both for performing calculations on attribute information, and for mapping purposes. Embedding of this concept in the TJS specification ensures that clients can determine appropriate operations to users, processing services can validate calculations, and mapping services can become fully automated.

6.4.1 Nominal and Ordinal attributes

Nominal and ordinal attributes are essentially the same: both provide some kind of naming, the only difference between them is that ordinal data contains some kind of ranking information, while nominal data is unranked.

Examples of nominal data include

- Name of a feature, such as the name of a county (Oxford)
- Name of a classification, such as the name of a type of soil (Clay)
- Name of a population, such as the name of an ethnic group (Hispanic)

Examples of ordinal data include

- Categories of the intensity of a phenomenon, such as the intensity of rainfall (low, medium, high).
- Categories of the depth of a feature, such as the depth to the water table (<2m, 2-10m, >10m)
- Categories of dates, such as the date of first occurrence (before 1900, 1900-2000, post 2000).

6.4.2 Count and Measure attributes

Count and measure data are both numeric in nature.

Count data is a cumulative total for an entire region (i.e. polygon). Count data is normally discrete units, such as the number of people or houses. When geospatial regions are merged, count data is cumulative (i.e. it must be summed to form a new total). For example, population is a count of the number of people in a region. When two adjacent towns merge to become a city, the population of the new city will be the sum of the populations of the two original towns.

Measure data is a measurement of some aspect of a region, which normally results from some calculation or measurement. In contrast to count data, measure data must be averaged when geospatial regions are merged (i.e. it is amalgamative). For example, rainfall is a measure of the amount of water that falls on an area. When the two adjacent towns merge to become a city, the rainfall of the new city can be calculated as a spatially weighted average of the rainfalls of the original towns.

Note that this differentiation can be confusing, because it is possible to convert between count and measure data through the introduction or removal of a unit area factor. For example, total population for a polygon is a count, but when population is presented as population per square kilometer, it is a measure. Likewise, depth of rainfall is a measure, but it can be converted to the total volume of water that has fallen on a polygon, in which case it becomes a count. Neither the presence of fractions nor the presence of area in the units of measurement is a definitive indicator of the type of numeric data. The true test is whether or not values are additive when polygons are merged.

Some examples are shown in Table 2 below:

Table 2 — Examples of some Count and Measure Attributes

Count (values are additive)	**Measure (values are amalgamative)**
Total population of Africa	Population density of Africa (e.g. per square kilometer)
Total number of cars in Boston	Number of cars per person in Boston
Gross Domestic Product for Switzerland	Average family income in Switzerland
Polygon area (i.e. total number of square kilometers).	Percentage of farmland in a polygon
Total volume of water flowing out of a polygon (e.g. watershed) in July	Total depth of rainfall for a polygon in July

7 Shared aspects of TJS operations

7.1 Introduction

This clause specifies aspects of the Table Joining Service that are shared by multiple requests and responses, including the GDAS encoding format.

7.2 Document conventions

The following conventions apply to all tables in this document which describe parameters and data structures.

The contents of the "Names" columns in the tables in this document follow the name capitalization rules for OGC services as specified in Subclause 11.6.2 of [OGC 06-121r3]. None of these names contain spaces, although they may appear to do so in this document.

In the "Data type and values" columns, the data type of many parameters is specified as "Character String type, not empty". In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

The "Multiplicity and use" columns specify whether a parameter or data structure must be present and populated in an operation request or response. All TJS servers shall implement each "mandatory" and "optional" parameter and data structure, checking that each request parameter and data structure is received with any allowed value(s). Similarly, all TJS clients shall implement each "mandatory" and "optional" parameter and data structure, using specified values.

To identify commonality between requests and responses, only the first definition of a commonly-used parameter or data structure is shown with a white background. All subsequent copies are shown with a light gray background.

7.3 Shared request parameters

The parameters shown in Table 3 below are used by all TJS requests.

Table 3 — Parameters common to most TJS operation requests

Names	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is "TJS"	One (mandatory)
request	Operation name	Character String type, not empty Value is operation name (e.g., "GetCapabilities")	One (mandatory)
version	Service version identifier	Character String type, not empty Value is "1.0" [a]	Zero or one (optional) When included, return the matching version [b] When omitted, return latest supported version
language	Language identifier for human readable text.	Character String type, not empty Value is a two or five character IETF RFC 4646 language identifier	Zero or one (optional) For use see section 7.6
NOTE 1 The parameters shown in gray above are largely copied from Table 3 of [OGC 06-121r3]. a TJS uses a version identification system of the form "X.Y.Z" as defined in [OGC 06-121r3], where "X" identifies the major version, "Y" identifies the minor version, and "Z" identifies the corrigendum for the TJS schemas. All TJS requests use an abbreviated two-part version identifier of the form "X.Y", while all TJS responses include the full three-part version identifier. At time of publication, the only valid version identifier for TJS requests is "1.0". b If the server does not support the requested version then it shall return a VersionNegotiationFailed exception as defined in [OGC 06-121r3].			

7.4 Geographic Data Attribute Set (GDAS) encoding format

This clause specifies the GDAS encoding format, portions of which are found in most TJS requests and responses.

The general structure of the GDAS XML encoding is as follows.

```
<GDAS>
   <Framework>
      ...              [spatial framework metadata]
      <Dataset>
         ...           [attribute dataset metadata]
         <Attribute>
            ...        [attribute metadata]
         </Attribute>
         <Rowset>
            ...        [attribute data]
         </Rowset>
      </Dataset>
   </Framework>
</GDAS>
```

The complete structure of a GDAS data is shown starting in Table 4. Note that in this and subsequent tables all names are shown in CamelCase: XML elements begin with capital letters, while XML attribute names begin with lowercase letters.

Table 4 — GDAS data encoding structure

Name	Definition	Data type and values	Multiplicity and use
GDAS	Parent element containing attribute data and all associated metadata	GDAS data structure, see Table 5	One (mandatory)

Table 5 — GDAS data encoding: GDAS data structure

Name	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty	One (mandatory) Value is "TJS"
version	TJS specification and schema version	Character String type, not empty [a]	One (mandatory) Value is "1.0"
lang	Language of the human-readable text (e.g. "en-CA")	Character String type, not empty	One (mandatory) RFC 4646 language tag
capabilities	GetCapabilities request URL for this service implementation	URL type	One (mandatory)
Framework	Description of a spatial framework for which data is available	Framework data structure, see Table 6	One (mandatory)
a This version identifier shall be a complete three-part version identifier of the form X.Y.Z			

Table 6 — GDAS data encoding: Framework data structure

Name	Definition	Data type and values	Multiplicity and use
FrameworkURI	The URI that uniquely references the spatial framework. Normally a URL, but a URN may be used.	Character String type, not empty	One (mandatory) Example: "http://agr.ca/slc/v1"
Organization	The name of the organization that is responsible for maintaining the framework dataset.	Character String type, not empty	One (mandatory)
Title	A human readable sentence fragment that might form a title if the framework dataset were displayed in map form.	Character String type, not empty	One (mandatory)
Abstract	A complete description or abstract that describes the framework dataset	Any (Character String type or XHTML)	One (mandatory)
ReferenceDate	The date to which the framework dataset applies.	DateTime type [a] ReferenceDate data structure, see Table 7	One (mandatory)
Version	Version identifier for this framework.	Character String type, not empty	One (mandatory)
Documentation	Web-accessible resource containing documentation describing the framework dataset.	URL type	Zero or one (optional)
FrameworkKey	Description of the key field(s) within the framework dataset (i.e. in the GIS) that are use to join data to this Framework.	Character String type, not empty. FrameworkKey structure, see Table 8	One (mandatory)
Bounding Coordinates	Bounding Coordinates of the spatial framework.	BoundingCoordinates data structure, see Table 10	One (mandatory)
Describe Datasets Request	URL reference to the DescribeDatasets request for datasets that apply to this framework.	href data structure, see Table 11	One (mandatory)
Dataset	Description of a dataset that contains attribute information that applies to the spatial framework.	Dataset data structure, see Table 12.	One (mandatory)

a This element shall be of the form defined at http://www.w3.org/TR/xmlschema-2/#dateTime, namely YYYY-MM-DDTHH:MM:SSzzzzzz or the related right truncated representations equivalent to date, gYearMonth or gYear.

Table 7 — GDAS data encoding: ReferenceDate data structure

Name	Definition	Data type and values	Multiplicity and use
startDate	Start of a time period to which the framework or dataset applies.	Date type [a]	Zero or One (optional)
a If “startDate” is included then “ReferenceDate” describes a range of time (from “startDate” to “ReferenceDate”) to which the framework/dataset applies. If “startDate” is not included then “ReferenceDate” describes a representative point in time to which the framework/dataset applies. The data type is as defined for “ReferenceDate”.			

Table 8 — GDAS data encoding: FrameworkKey data structure

Name	Definition	Data type	Multiplicity and use
Column	Description of a key field within the framework.	Column data structure, see Table 9	One or more (mandatory)

Table 9 — GDAS data encoding: Column data structure

Name	Definition	Data type	Multiplicity and use
name	Name of the key field within the framework dataset.	String	One (mandatory)
type	Datatype of the field, as defined by XML schema e.g. “http://www.w3.org/TR/xmlschema-2/#integer”	URI type	One (mandatory)
length	Length of the field, in characters or digits	Positive Integer	One (mandatory)
decimals	Number of digits after the decimal, for decimal numbers with a fixed number of digits after the decimal.	Non-Negative Integer	Zero or one (optional) Include for decimal numbers.

Table 10 — GDAS data encoding: BoundingCoordinates data structure

Name	Definition	Data type	Multiplicity and use
North	WGS84 latitude of the northernmost coordinate of the spatial framework.	Decimal	One (mandatory)
South	WGS latitude of the southernmost coordinate of the spatial framework	Decimal	One (mandatory)
East	WGS84 longitude of the easternmost coordinate of the spatial framework.	Decimal	One (mandatory)
West	WGS84 longitude of the westernmost coordinate of the spatial framework.	Decimal	One (mandatory)

Table 11 — GDAS data encoding: href data structure

Name	Definition	Data type and values	Multiplicity and use
href	HTTP reference.	URL type	One (mandatory)

Table 12 — GDAS data encoding: Dataset data structure

Name	Definition	Data type and values	Multiplicity and use
DatasetURI	A URI that uniquely identifies the attribute dataset. Normally a URL, but a URN may be used.	Character String type, not empty	One (mandatory) Example: “http://nrcan.gc.ca/slc/permafrost”
Organization	The name of the organization that is responsible for maintaining the dataset.	Character String type, not empty	One (mandatory)
Title	A human readable sentence fragment that might form a title if the dataset were displayed in tabular form.	Character String type, not empty	One (mandatory)
Abstract	An abstract that describes the purpose and contents of the dataset	Any (Character String type or XHTML)	One (mandatory)
ReferenceDate	The date to which the tabular data applies.	Date type [a] ReferenceDate data structure, see Table 7	One (mandatory)
Version	Version identifier for this dataset. The combination of DatasetURI and Version shall identify any changes, including corrections, to the published content of the Columnset and Rowset elements.	Character String type, not empty	One (mandatory)
Documentation	Web-accessible resource containing documentation describing the attribute dataset.	URL type	Zero or one (optional) Include when further documentation is available on-line.
DescribeData Request	URL reference to the DescribeData request for data found in this dataset.	href data structure, see Table 11	One (mandatory)
Columnset	Structure containing descriptions of the dataset columns (i.e. the contents of each Row in the Rowset structure) of the dataset being exchanged.	Columnset data structure, see Table 13	One (mandatory)
Rowset	Structure containing the attribute values for the dataset being exchanged.	Rowset data structure, see Table 29	One (mandatory)
Note 1: When multiple attributes are included, the ordering of the Attribute elements inside the Dataset element shall be identical to the order in which they were listed in the GetData request. Furthermore, the ordering of the Attribute elements inside the Dataset element shall be identical to the ordering of the V elements inside the Row element of the Framework/Dataset/Rowset data structure (see Table 30). a This element shall be of the form defined at http://www.w3.org/TR/xmlschema-2/#dateTime, namely YYYY-MM-DDTHH:MM:SSzzzzzz or the related right truncated representations equivalent to date, gYearMonth or gYear.			

Table 13 — GDAS data encoding: Columnset data structure

Name	Definition	Data type and values	Multiplicity and use
Framework Key	Description of the key dataset column (the "K" element in each "Row" of the Rowset data structure) that is used to join the Rowset contents to the spatial framework.	FrameworkKey data structure, see Table 14	One (mandatory)
Attributes	Describes a data column (a "V" element found in each Row of the Rowset data structure).	Attributes data structure, see Table 15	One or more (mandatory)

Table 14 — GDAS data encoding: FrameworkKey data structure

Name	Definition	Data type and values	Multiplicity and use
complete	Identifies that there is at least one record in the Attribute dataset for every identifier in the Framework dataset. [a]	Boolean	One (mandatory)
relationship	Identifies if the relationship between the Framework and the Attribute datasets are 1:1 or 1:many. [b]	Character String type, containing "one" or "many"	One (mandatory)
Column	Describes a column that forms part of the linkage key. [c]	FrameworkKey/ Column data structure, see Table 9	One or more (mandatory) [d]
a "one" indicates that there is at most one record in the Attribute dataset for every identifier in the Framework dataset. "many" indicates that there may be more than one record in the Attribute dataset for every identifier in the Framework dataset. b "true" indicates that there is at least one record in the Attribute dataset for every identifier in the Framework dataset. "false" indicates that some identifiers in the Framework dataset cannot be found in the Attribute dataset. c The primary key for the dataset may not necessarily be the geographic key. The primary key may be comprised of multiple columns and include, for example, a temporal variable in addition to the geographic identifier. d When multiple columns are present then all of the columns shall be required to join the attribute table to the spatial framework (i.e. they form a composite key). In this case, the names of the each of the columns in the Dataset/Columnset/FrameworkKey structure shall match those of the equivalent columns in the Framework/FrameworkKey structure. When only one column is present, the name of this column may be different from the name of the column found in the Framework/FrameworkKey structure.			

Table 15 — GDAS data encoding: Attributes data structure

Name	Definition	Data type and values	Multiplicity and use
Column	Describes a column that contains attribute data. [c]		One or more (mandatory) [d]

Table 16 — GDAS data encoding: Attributes/Column data structure

Name	Definition	Data type and values	Multiplicity and use
name	The name of the attribute. Generally the name by which the attribute is identified within the RDBMS database.	Character String type, not empty [a]	One (mandatory)
type	Datatype, as defined by XML schema e.g. "http://www.w3.org/TR/xmlschema-2/#string"	URI type	One (mandatory)
length	Length of the field, in characters	Positive Integer	One (mandatory)
decimals	Number of digits after the decimal, for decimal numbers with a fixed number of digits after the decimal.	Positive Integer	Zero or one (optional) Include for decimal numbers where the number of decimal places present is important.
purpose	Purpose, indicating whether the attribute contains data, a linkage key, or other content.	Character String type, restricted to content shown in Table 17	One (mandatory)
Title	A human readable sentence fragment that might form a title if the attribute were displayed in tabular form.	Character String type, not empty	One (mandatory) [b]
Abstract	An abstract that describes the purpose and contents of the attribute.	Any (Character String type or XHTML)	One (mandatory)
Documentation	Web-accessible resource containing documentation describing the attribute.	URL type	Zero or one (optional) Include when further documentation is available on-line.
Values	Description of the content of the attribute.	Values data structure, see Table 18	One (mandatory)
GetDataRequest	URL reference to the GetData request for this column.	href data structure, see Table 11	Zero or One (optional) [c] Include when column "purpose" is "Attribute"

NOTE 1 The parameters shown in gray above are copied from Table 9.

a Attribute names may not contain a comma (",").

b Note that the unit of measure should not appear in the Title. Instead, it should appear in the UOM element.

c This GetData request shall be constructed such that it includes all additional columns in the dataset that provide context for this column (i.e. those columns where the "purpose" is other than "Attribute") .

Table 17 — GDAS data encoding: purpose data contents

Character String	Definition
SpatialComponentIdentifier	Column contains an abstract nominal or ordinal identifier for spatial components found within the feature. This value is for use when KeyRelationship = “many”.
SpatialComponentProportion	Column contains a proportion (from 0 to 1) of the spatial feature (i.e. the object identified by the PrimarySpatialIdentifier) to which the component applies. For use when KeyRelationship = “many”.
SpatialComponentPercentage	Column contains a percentage (from 0 to 100) of the spatial feature (i.e. the object identified by the PrimarySpatialIdentifier) to which the component applies. For use when KeyRelationship = “many”.
TemporalIdentifier	Column contains a nominal or ordinal identifier that indicates the temporal positioning of the data (e.g. first). For use when KeyRelationship = “many”.
TemporalValue	Column contains a date/time measure (e.g. 2001). For use when KeyRelationship = “many”.
VerticalIdentifier	Column contains a nominal or ordinal identifier that indicates the depth or elevation of the data (e.g. lowest). For use when KeyRelationship = “many”.
VerticalValue	Column contains a numeric measure of the depth or elevation of the data (e.g. 120). For use when KeyRelationship = “many”.
OtherSpatialIdentifier	Column contains a geographic linkage key for some other spatial Framework (i.e. not the one identified in the parent “Framework” element). For use when other spatial identifiers are present in the tabular data, as may be the case for data which applies to a spatial hierarchy.
NonSpatialIdentifer	Column contains a nonspatial identifier - i.e. a relate key used to perform a table join to a table which does not contain spatial geometery.
Attribute	Column contains attribute data describing a geographic object in the spatial Framework (i.e. suitable for mapping).

Table 18 — GDAS data encoding: Values data structure

Name	Definition	Data type and values	Multiplicity and use
Nominal	Attribute consists of Nominal data (i.e. names, such as "Canada")	Nominal data structure, see Table 19	Zero or one (conditional) [a]
Ordinal	Attribute consists of Ordinal data (i.e. rankings, such as "high=1, medium=2, low=3"	Ordinal data structure, see Table 23	Zero or one (conditional) [a]
Count	Attribute consists of Count data (i.e. total number of objects, such as "number of cows")	Count & Measure data structure, see Table 26	Zero or one (conditional) [a]
Measure	Attribute consists of Measure data (i.e. numeric measurement or calculated value such as "degrees Celsius")	Count & Measure data structure, see Table 26	Zero or one (conditional) [a]
a One and only one of these four items shall be included. For further information, see section 6.4.			

Table 19 — GDAS data encoding: Nominal data structure

Name	Definition	Data type and values	Multiplicity and use
Classes	Nominal classes (i.e. attribute contains classification information of some kind). (e.g. attribute contains information regarding geological classifications)	Nominal / Classes data structure, see Table 20	One (optional) [a] Use when names are classes of objects (i.e. names repeat in multiple rows)
Exceptions	Valid exception classes for this attribute	Exceptions data structure, see Table 22	Zero or one (optional) Include when the data contents can include exception identifiers.
a Normally only used when the Columnset/Attributes/Column purpose is "Attribute".			

Table 20 — GDAS data encoding: Nominal / Classes data structure

Name	Definition	Data type and values	Multiplicity and use
Title	Human-readable short description suitable to display on a pick list, legend, and/or on mouseover.	Character String type, not empty	One (mandatory)
Abstract	One or more paragraphs of human-readable relevant text suitable for display in a pop-up window.	Any (Character String type or XHTML)	One (mandatory)
Documentation	URL reference to a web-accessible resource which contains further information describing this object.	URL type	Zero or one (optional) Include when further documentation is available on-line.
Value	Description of a valid (non-null) nominal class that is found in this attribute field	Nominal / Classes / Value data structure, see Table 21	One or more (mandatory)

Table 21 — GDAS data encoding: Value & Null data structure

Name	Definition	Data type and values	Multiplicity and use
Identifier	Content of the attribute field for one or more records in the dataset.	Character String type	One (mandatory)
Title	A human readable sentence fragment that might form a title if the attribute were displayed in tabular form.	Character String type, not empty	One (mandatory)
Abstract	An abstract that describes the meaning of the value	Any (Character String type or XHTML)	One (mandatory)
Documentation	Web accessible resource containing documentation describing the value.	URL type	Zero or one (optional) Use when further documentation is available on-line.
color	Hex code for a color that is suggested for cartographic portrayal of this value (e.g. “FFCCFF”).	Hex code	One (optional) [a] Include when available.
a Not used for attributes of type “Measure” and “Count”.			

Table 22 — GDAS data encoding: Exceptions data structure

Name	Definition	Data type and values	Multiplicity and use
Null	Description of a nominal class that is found in this attribute field	Null data structure, see Table 21	One or more (mandatory)

Table 23 — GDAS data encoding: Ordinal data structure

Name	Definition	Data type and values	Multiplicity and use
Classes	Ordinal classes. (e.g. attribute contains information regarding geological classifications)	Ordinal / Classes data structure, see Table 24	Zero or one (optional) Use when attribute "purpose" is "Data"
Exceptions	Valid exception classes for this attribute	Exceptions data structure, see Table 22	Zero or one (optional) Use when the data contents can include exception identifiers.

Table 24 — GDAS data encoding: Ordinal / Classes data structure

Name	Definition	Data type and values	Multiplicity and use
Title	Human-readable short description suitable to display on a pick list, legend, and/or on mouseover.	Character String type, not empty	One (mandatory)
Abstract	One or more paragraphs of human-readable relevant text suitable for display in a pop-up window.	Any (Character String type or XHTML)	One (mandatory)
Documentation	URL reference to a web-accessible resource which contains further information describing this object.	URL type	Zero or one (optional)
Value	Description of an ordinal class that is found in this attribute field	Ordinal / Classes / Value data structure, see Table 25	One or more (mandatory)
Note 1: The shaded elements in this table are identical to those in Table 20.			

Table 25 — GDAS data encoding: Ordinal / Classes / Value data structure

Name	Definition	Data type and values	Multiplicity and use
Identifier	Content of the attribute field for one or more records in the dataset.	Character String type	One (mandatory)
Title	A human readable sentence fragment that might form a title if the attribute were displayed in tabular form.	Character String type, not empty	One (mandatory)
Abstract	An abstract that describes the meaning of the value	Any (Character String type or XHTML)	One (mandatory)
Documentation	Web accessible resource containing documentation describing the value.	URL type	Zero or one (optional)
rank	Ordinal ranking of this value	Non-negative integer	One (mandatory)
color	Hex code for a color that is suggested for cartographic portrayal of this value (e.g. "FFCCFF").	Hex code	Zero or one (optional) Include when available.
Note 1: The shaded elements in this table are identical to those in Table 21.			

Table 26 — GDAS data encoding: Count & Measure data structure

Name	Definition	Data type and values	Multiplicity and use
UOM	Unit of Measure	Character String type, not empty; UOM data structure, see Table 27	One (mandatory)
Uncertainty	Standard measure of uncertainty (u_i) equal to the positive square root of the estimated variance for this data. See http://physics.nist.gov/cuu/Uncertainty/	See Table 28 — GDAS data encoding: Uncertainty data structure	Zero or one (optional) Include when known.
Exceptions	Valid exception classes for this attribute	Exceptions data structure, see Table 22	Zero or one (optional) Use when the data contents can include exception identifiers.

Table 27 — GDAS data encoding: UOM data structure

Name	Definition	Data type and values	Multiplicity and use
reference	Reference to a description of the unit of measure.	URI type	Zero or One (optional) Use when a definition is available on-line.
ShortForm	Short text description of the unit of measure, suitable for charts or legends (e.g. "°C").	Character String Type, not empty	One (mandatory)
LongForm	Long form of the unit of measure, suitable for complete text descriptions (e.g. "degrees centigrade").	Character String Type, not empty	One (mandatory)

Table 28 — GDAS data encoding: Uncertainty data structure

Name	Definition	Data type and values	Multiplicity and use
gaussian	Distribution form of the data. When "true" the distribution of the data is Gaussian and Independent and Identically Distributed (IID)	"true", "false", or "unknown". Default is "unknown". Character String type, not empty	Zero or one (optional) Include when known.

Rowset follows the Attribute element(s). It contains 1 or more Row elements, one for each record of data requested by the client, as indicated in Table 29

Table 29 — GDAS data encoding: Rowset data structure

Name	Definition	Data type and values	Multiplicity and use
Row	Data row containing selected attributes from the RDBMS.	Row data structure, see Table 30 [a]	One or more (mandatory)
a This element shall be ordered from lowest to highest according to the value of the "K" child element.			

Each Row consists of one or more <K> (key) elements which contain the value of the FrameworkKey for this record, followed by one <V> (value) tag for each selected attribute. The order of the <V> elements inside each <Row> is important. The first <V> value in a row corresponds to the first Attribute listed in the Dataset element, the second <V> value corresponds to the second attribute and so on.

Table 30 — GDAS data encoding: Row data structure

Name	Definition	Data type and values	Multiplicity and use
K	Key (e.g. polygon identifier) that references the spatial feature to which this row applies.	Character String type, not empty [a] K data structure, see Table 31	One or more (mandatory)
V	Value of a specified attribute for the spatial feature identified by the key for this Row.	Character String type, not empty; V data structure, see Table 32 [b]	One or more (mandatory)
a This identifier shall be a member of a Columnset/FrameworkKey column described in Table 13. b When this value is null it must be indicated by setting its null attribute to "true". An identification of the reason may be included in the content of this element, but it must be described in the Exceptions element, see Table 22. When null is set to "true" the content of this element may differ from that identified in the corresponding "type" definition found in the attribute description. E.g. for an attribute of type "http://www.w3.org/TR/xmlschema-2/#integer", when null="true" the element may actually contain a string.			
Note 1: For multiple attributes, the order of the V elements within the XML file shall be identical to the order of the Column elements in the Dataset/Columnset/Attributes data structure (see Table 12). Note 2: There shall only be one "V" for any particular attribute within a Row. Where multiple "V"s exist within a dataset for any identifier, these shall be represented each in a separate Row, and the relationship attribute must contain the value "many" as indicated in Table 14.			

Table 31 — GDAS data encoding: K data structure

Name	Definition	Data type and values	Multiplicity and use
aid	FrameworkKey identifier, namely the corresponding Columnset/Framework/Column name	Character string type, not empty	Zero or one (optional) Include for debugging or facilitation of XML transformation.

Table 32 — GDAS data encoding: V data structure

Name	Definition	Data type and values	Multiplicity and use
aid	Attribute identifier, namely the corresponding AttributeName	Character string type, not empty	Zero or one (optional) Include for debugging or facilitation of XML transformation.
null	Boolean value, when present and "true" indicates that this particular value is missing for some reason, and the contents of the element must be processed accordingly.	Boolean, default is false.	Zero or one (optional) Include for all missing values.
Note 1. The null attribute in this structure is redundant with the null element described in Table 22 whenever null values are coded. However, the explicit identification of null values for every "V" element simplifies: 1) numeric calculations in XML, 2) the separation of real data from null values, and 3) the counting of null values.			

7.5 Operation request encoding

The encoding of operation requests shall use HTTP GET with KVP encoding and HTTP POST with XML or SOAP with XML encoding as specified in Clause 11 of [OGC 06-121r3]. Table 33 summarizes the TJS operations and their encoding methods defined in this standard.

Table 33 — Operation request encoding

Operation name	Request encoding
GetCapabilities (required)	KVP and optional XML
DescribeFrameworks	KVP and optional XML
DescribeDatasets	KVP and optional XML
DescribeData	KVP and optional XML
GetData	KVP and optional XML
DescribeJoinAbilities	KVP and optional XML
Describekey	KVP and optional XML
JoinData	KVP and optional XML

7.6 Multilingual support

The languages supported by a TJS server are identified in the GetCapabilities operation response.

The “language” parameter, an optional parameter for all TJS requests, identifies the client's preferred language. Its value is an RFC 4646 language tag. This tag must match the contents of one of the Language elements listed in the GetCapabilities response. For each human language text string in the server's response, the server shall return the string in the language requested.

If the “language” parameter is not present in a request, it is recommended that the server attempt to honour the AcceptLanguage MIME header in the HTTP request (usually passed to the process by the web server by means of the HTTP_ACCEPT_LANGUAGES environment variable) instead.

The language of the response shall be indicated by the mandatory “lang” attribute which is found in the root element of all TJS service responses.

8 GetCapabilities operation (mandatory)

8.1 Introduction

Like most OGC services, TJS includes a mandatory GetCapabilities operation. This operation allows clients to retrieve service metadata. The response to a GetCapabilities request shall be an XML document containing service metadata, including specific information about the operations supported by a particular server. This clause specifies the XML document that a TJS server shall return to describe its capabilities.

8.2 GetCapabilities operation request

The GetCapabilities operation request shall be as specified in Subclauses 7.2 and 7.3 of [OGC 06-121r3]. The value of the "service" parameter shall be "TJS". The allowed set of service metadata (or Capabilities) XML document section names and meanings shall be as specified in Tables 3 and 7 of [OGC 06-121r3], with the additions listed in Table 34 below.

Table 34 — Additional Section name values and meanings

Section name	Meaning
Languages	Return Languages section in service metadata document
WSDL	Return WSDL section in service metadata document

The "Multiplicity and use" column in Table 1 of [OGC 06-121r3] specifies the optionality of each listed parameter in the GetCapabilities operation request. Table 35 specifies the implementation of those parameters by TJS clients and servers.

Table 35 —GetCapabilities operation request URL parameters

Names	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “TJS”	One (mandatory)
request	Operation name	Character String type, not empty Value is “GetCapabilities”)	One (mandatory)
Accept Versions	Prioritized sequence of one or more specification versions accepted by client, with preferred versions listed first	Character String type, not empty Value is “1.0” [a]	Zero or one (optional) When included, return the latest supported matching version When omitted, return latest supported version
language	Language identifier for human readable text	Character String type, not empty Value is a two or five character IETF RFC 4646 language identifier	Zero or one (optional) For use see section 7.6

a The identification of versions in the GetCapabilities request takes the abbreviated two-part form (“X.Y”) instead of the three-part form “X.Y.Z” defined in [OGC 06-121r3]. As such it will not validate correctly using the XML schemas originally defined by OWS Common 1.1, so an adjusted schema document is included in the zip files bundled with this document.

NOTE 1 The parameters shaded in gray above are largely copied from Table 3 in Subclause 7.3 of this document.

All TJS servers shall implement HTTP GET transfer of the GetCapabilities operation request, using KVP encoding. Servers may also implement HTTP POST transfer of the GetCapabilities operation request, using XML encoding only.

EXAMPLE 1 To request a TJS capabilities document, a client could issue the following KVP encoded GetCapabilities operation request with minimum contents:

http://foo.bar/foo&service=TJS&request=GetCapabilities

8.3 GetCapabilities operation response

8.3.1 Normal response

The service metadata document shall contain the optional sections specified in Table 36. Depending on the values in the Sections parameter of the GetCapabilities operation request, any combination of these sections can be requested and shall be returned when requested.

Table 36 — Section name values and contents

Section name	Contents
ServiceIdentification	Metadata about this specific server. The schema of this section shall be the same as for all OWSs, as specified in Subclause 7.4.3 and owsServiceIdentification.xsd of [OGC 06-121r3].
ServiceProvider	Metadata about the organization operating this server. The schema of this section shall be the same for all OWSs, as specified in Subclause 7.4.4 and owsServiceProvider.xsd of [OGC 06-121r3].
OperationsMetadata	Metadata about the operations specified by this service and implemented by this server, including the URLs for operation requests. The basic contents and organization of this section shall be the same for all OWSs, as specified in Subclause 7.4.5 and owsOperationsMetadata.xsd of [OGC 06-121r3].
Languages	Metadata about the languages supported by this server. See section 8.3.3

In addition to these sections, each service metadata document shall include the mandatory "version" and optional updateSequence parameters specified in Table 6 in Subclause 7.4.1 of [OGC 06-121r3].

8.3.2 OperationsMetadata section standard contents

For the TJS, the OperationsMetadata section shall be the same as for all OGC Web Services, as specified in Subclause 7.4.5 and owsOperationsMetadata.xsd of [OGC 06-121r3]. The mandatory values of various (XML) attributes shall be as specified in Table 37. Similarly, the optional attribute values listed in Table 38 shall be included or not depending on whether that operation is implemented by that server. In Table 37 and Table 38, the "Attribute name" column uses dot-separator notation to identify parts of a parent item. The "Attribute value" column references an operation parameter, in this case an operation name, and the meaning of including that value is listed in the right column.

Table 37 — Required values of OperationsMetadata section attributes

Attribute name	Attribute value	Meaning of attribute value
Operation.name	GetCapabilities	The GetCapabilities operation is implemented by this server.

Table 38 — Optional values of OperationsMetadata section attributes

Attribute name	Attribute value	Meaning of attribute value
Operation.name	DescribeFrameworks	The DescribeFrameworks operation is implemented by this server.
	DescribeDatasets	The DescribeDatasets operation is implemented by this server.
	DescribeData	The DescribeData operation is implemented by this server.
	GetData	The GetData operation is implemented by this server.
	DescribeJoinAbilities	The DescribeJoinAbilities operation is implemented by this server.
	DescibeKeys	The DescibeKeys operation is implemented by this server.
	JoinData	The JoinData operation is implemented by this server.

The GetData section includes two optional attributes: *GeolinkIdsLimit* and *AttributeLimit*,

The optional *GeolinkIdsLimit* constraint is a positive integer indicating the maximum number of **GeolinkIds** a client is permitted to include in a single GetData Request. If the value of this element is “0”, this feature is not supported and requests cannot include the *GeolinkIds* parameter. If this element is absent, the server imposes no limit.

The optional *AttributeLimit* constraint is a positive integer indicating the maximum number of **Attribute**s a client is permitted to include in a single GetData Request. If this element is absent, the server imposes no limit. Zero is not an allowable value. An example follows.

```
    <ows:Constraint name=“GeolinkidsLimit”>
        <ows:AllowedValues>
            <ows:Range>
                <ows:MaximumValue>1000</ows:MaximumValue>
            </ows:Range>
        </ows:AllowedValues>
        <ows:Meaning>The maximum number of GeolinkIds that can be specified as
    part of the GeolinkIds element in a GetData request.</ows:Meaning>
    </ows:Constraint>
    <ows:Constraint name=“AttributeLimit”>
        <ows:AllowedValues>
            <ows:Range>
                <ows:MaximumValue>1</ows:MaximumValue>
            </ows:Range>
        </ows:AllowedValues>
        <ows:Meaning>The maximum number of Attributes that can be requested in
    one GetData request.</ows:Meaning>
    </ows:Constraint>
</ows:Operation>
```

The JoinData section includes an optional *AttributeLimit* constraint, which is a positive integer indicating the maximum number of **Attribute**s permitted by a client in a single

JoinData Request. If this element is absent, the server imposes no limit. Zero is not an allowable value. An example follows.

```
<ows:Constraint name="AttributeLimit">
    <ows:AllowedValues>
        <ows:Range>
            <ows:MaximumValue>1</ows:MaximumValue>
        </ows:Range>
    </ows:AllowedValues>
    <ows:Meaning>The maximum number of Attributes that can be included as
part of one JoinData request.</ows:Meaning>
</ows:Constraint>
```

Note that this value is also provided as part of the DescribeJoinAbilities operation response.

8.3.2.1 Encodings for operation requests

All TJS servers shall specify the encodings that may be sent using HTTP POST transfer of operation requests. Specifically, an ows:Constraint element shall be included, with "PostEncoding" as the value of the "name" attribute and specifying different allowed values for each allowed encoding:

a) The value "SOAP" shall indicate that SOAP encoding is allowed, as specified in Subclause 11.8.

b) The value "XML" shall indicate that XML encoding is allowed (without SOAP message encapsulation).

c) The value "KVP" shall indicate that KVP encoding is allowed, when using HTTP POST transfer.

If the HTTP POST connect point URL is different for different encodings of the operation requests, this ows:Constraint element shall be included in each Post element. If the connect point URL is the same for all encodings of all operation requests, this ows:Constraint element shall be included in the OperationsMetadata element.

8.3.3 Identification of languages supported

The mandatory <Languages> element in the service metadata lists the languages (as RFC 4646 language tags) that this server is able to fully support. That is, if one of the listed languages is requested using the "Language" parameter, all text strings contained in the response are guaranteed to be in that language. Table 39 indicates the structure of this section of the Capabilities document.

Table 39 — GetCapabilities response encoding: Languages data structure

Name	Definition	Data type and values	Multiplicity and use
defaultLanguage	Default language of all service responses e.g. "en-CA"	Character String type, not empty	One (mandatory)
Language	Language which can be selected for all service responses	Character String type, not empty	One or more (optional)
Note: the default language must be present in the set of Language elements. This approach facilitates the creation of a language pick list.			

An example of the languages element follows

```
<tjs:Languages defaultLanguage="en-CA">
        <ows:Language>en-CA</ows:Language>
        <ows:Language>fr-CA</ows:Language>
</tjs:Languages>
```

See also section 7.6

8.3.4 Identification of WSDL location

The optional <WSDL> element in the service metadata identifies the location from which a Web Services Description Language (WSDL) document describing the service can be retrieved. Table 40 indicates the structure of this section of the Capabilities document.

Table 40 — GetCapabilities response encoding: WSDL data structure

Name	Definition	Data type and values	Multiplicity and use
href	The URL from which the WSDL document can be retrieved.	URL type, not empty	One (mandatory)

An example of the WSDL element follows

```
<tjs:WSDL xlink :href="http://foo.bar/foo"/>
```

8.3.5 Capabilities document XML encoding

The XML schema for a TJS service metadata document extends ows:CapabilitiesBaseType in owsCommon.xsd of [OGC 06-121r3], and is found on the OGC website at http://schemas.opengis.net/, and replicated in Appendix A.

As indicated, this XML Schema Document uses the owsServiceIdentification.xsd, owsServiceProvider.xsd, and owsOperationsMetadata.xsd schemas specified in [OGC

06-121r3]. It also uses an XML Schema Document for the “Contents” section of the TJS Capabilities XML document, which shall be as attached in the tjsCapabilitiesContents.xsd file. All these XML Schema Documents contain documentation of the meaning of each element, attribute, and type, and this documentation shall be considered normative as specified in Subclause 11.6.3 of [OGC 06-121r3].

8.3.6 Exceptions

When a TJS server encounters an error while performing a GetCapabilities operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed exception codes shall include those listed in Table 5 of Subclause 7.4.1 of [OGC 06-121r3].

9 DescribeFrameworks operation (optional)

9.1 Introduction

The *DescribeFrameworks* operation returns an XML document that describes one or more spatial frameworks for which data is available from the server. This description includes information that uniquely identifies each spatial framework, and descriptive information about each framework. This information can be used to populate a user interface to identify the spatial framework for which the user would like to obtain attribute data. When the optional FrameworkURI parameter is included, the response can be used to populate an “about” page for a specific framework.

This operation shall be supported for all servers that implement data access, as identified in section 6.1.1.

9.2 DescribeFrameworks operation request

9.2.1 DescribeFrameworks request parameters

A request to perform the *DescribeFrameworks* operation shall include the parameters listed in Table 41. This table also specifies the source of values and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

Table 41 — Parameters in DescribeFrameworks operation request

Names	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is "TJS"	One (mandatory)
request	Operation name	Character String type, not empty Value is "DescribeFrameworks")	One (mandatory)
version	Service version identifier	Character String type, not empty Value is "1.0"	Zero or one (optional) When included, return the matching version When omitted, return latest supported version
language	Language identifier for human readable text	Character String type, not empty Value is a two or five character IETF RFC 4646 language identifier	Zero or one (optional) For use see section 7.6
Framework URI	URI of a spatial framework to which the attribute data can be joined.	URI type, not empty Value is a URI as found in the *DescribeFrameworks* response.	Zero or one (optional) [a]
a The list of frameworks supported by a server is normally unknown during an initial DescribeFrameworks request to a server, and therefore this parameter is normally absent during such a request. If this parameter is absent, the response shall include descriptions for all frameworks for which data access is supported by the server. NOTE 1 The parameters shaded in gray above are largely copied from Table 3 in Subclause 7.3 of this document.			

9.2.2 DescribeFrameworks request KVP encoding (mandatory)

Servers shall implement HTTP GET transfer of the *DescribeFrameworks* operation request, using KVP encoding. The KVP encoding of this request shall use the parameters specified in Table 41.

EXAMPLE An example DescribeFrameworks operation request KVP encoded for HTTP GET is:

```
http://foo.bar/foo?
    Service=TJS&
    Version=1.0&
    Request=DescribeFrameworks&
    Language=en-CA
```

9.2.3 DescribeFrameworks request XML encoding (optional)

Servers may also implement HTTP POST transfer of the *DescribeFrameworks* operation request, using XML encoding only, in accordance with the *tjsDescribeFrameworks_request.xsd* XML schema. A valid DescribeFrameworks operation request encoded in XML is shown in Annex C.

9.3 DescribeFrameworks operation response

9.3.1 DescribeFrameworks normal response parameters

The normal response to a valid DescribeFrameworks operation request shall be a FrameworkDescriptions data structure, which contains descriptions of one or more Frameworks for which data is available on this server. The structure of this response is shown in Table 42.

Table 42 — *DescribeFrameworks* response data structure

Name	Definition	Data type	Multiplicity and use
Framework Descriptions	Full description of all spatial frameworks for which data is available	FrameworkDescriptions data structure, see Table 43	One (mandatory)

Table 43 — *DescribeFrameworks* response: FrameworkDescriptions data structure

Name	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty	One (mandatory) Value is “TJS”
version	TJS specification and schema version	Character String type, not empty	One (mandatory) Value is “1.0”
lang	RFC 4646 language tag of the human-readable text (e.g. “en-CA”)	Character String type, not empty	One (mandatory)
capabilities	GetCapabilities request URL for this service implementation	URL type	One (mandatory)
Framework	Description of a spatial framework for which data is available	Partial data structure defined in Table 6, excluding the Framework/Dataset element.	One or more (mandatory)
Note 1: The values for the shaded elements in this table are identical to those submitted as part of the *DescribeFrameworks* request and documented in Table 41.			

9.3.2 DescribeFrameworks normal response XML encoding

The *DescribeFrameworks* operation response shall be encoded in XML in accordance with the *tjsDescribeFrameworks_response.xsd* XML schema (see Annex B). A valid DescribeFrameworks operation response encoded in XML is shown in Annex C.

9.3.3 DescribeFrameworks exceptions

When a TJS server encounters an error while performing a DescribeFrameworks operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 20 of that document. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 44.

Table 44 — Exception codes for DescribeFrameworks operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
NOTE The values listed above are copied from Table 20 in Subclause 8.3 of [OGC 06-121r3].		

10 DescribeDatasets operation (optional)

10.1 Introduction

The *DescribeDatasets* operation returns an XML document that identifies all of the attribute datasets for which data is available from the service instance. There may be one or more attribute datasets that apply to any spatial framework. The response includes descriptive information about each dataset available from the service instance, and information that uniquely identifies each dataset. This description can be used to automatically build a user interface to identify the dataset from which the user would like to obtain attribute data. When the optional DatasetURI parameter is included, the response can be used to populate an "about" page for a specific dataset.

This operation shall be supported for all servers that implement data access, as identified in section 6.1.1.

10.2 DescribeDatasets operation request

10.2.1 DescribeDatasets request parameters

A request to perform the *DescribeDatasets* operation shall include the parameters listed in Table 45. This table also specifies the source of values and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

Table 45 — Parameters in DescribeDatasets operation request

Names	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “TJS”	One (mandatory)
request	Operation name	Character String type, not empty Value is “DescribeDatasets”)	One (mandatory)
version	Service version identifier	Character String type, not empty Value is “1.0”	Zero or one (optional) When included, return the matching version When omitted, return latest supported version
language	Language identifier for human readable text	Character String type, not empty Value is a two or five character IETF RFC 4646 language identifier	Zero or one (optional) For use see section 7.6
Framework URI	URI of a spatial framework to which the attribute data can be joined.	URI type, not empty Value is a URI as found in the *DescribeFrameworks* response.	Zero or one (optional) [a]
DatasetURI	URI of an attribute table, which can be joined to the spatial framework identified by the *FrameworkURI*.	URI type, not empty Value is a DatasetURI as found in the *DescribeDatasets* response.	Zero or one (optional) [b]

a If this parameter is absent, the response shall include descriptions for all frameworks for which data access is supported by the server.

b DatasetURIs are normally unknown during an initial DescribeDatasets request to the server, and therefore this parameter is normally absent from such a request. If this parameter is absent, the response shall include descriptions for all datasets available for the specified framework(s).

NOTE 1 The parameters shaded in gray above are largely copied from Table 3 in Subclause 7.3 of this document.

NOTE 2: The values for all fields shall be encoded using the standard Internet practice for encoding URLs [IETF RFC 1738].

10.2.2 DescribeDatasets request KVP encoding (mandatory)

Servers shall implement HTTP GET transfer of the *DescribeDatasets* operation request, using KVP encoding. The KVP encoding of this request shall use the parameters specified in Table 45.

EXAMPLE A valid DescribeDatasets operation request KVP encoded for HTTP GET is:

```
http://foo.bar/foo?
    Service=TJS&
    Version=1.0&
    Request=DescribeDatasets&
    FrameworkURI=http://foo.bar2/Provinces/2001&
    Language=en-CA
```

NOTE: For clarity, the parameter values in the above example have not been URL encoded.

10.2.3 DescribeDatasets request XML encoding (optional)

Servers may also implement HTTP POST transfer of the *DescribeDatasets* operation request, using XML encoding only, in accordance with the tjsDescribeDatasets_request.xsd XML schema.

10.3 DescribeDatasets operation response

10.3.1 DescribeDatasets normal response parameters

The normal response to a valid *DescribeDatasets* operation request shall be a DatasetDescriptions data structure, which contains descriptions of one or more data tables for which data is available on this server. The structure of this response is shown in Table 46. Note that the Framework data structure portion of this response is similar to the equivalent structure in the *DescribeFrameworks* response.

Table 46 — *DescribeDatasets* response data structure

Name	Definition	Data type	Multiplicity and use
Dataset Descriptions	General description of all datasets available on this server	DatasetDescriptions data structure, see Table 47	One (mandatory)

Table 47 — *DescribeDatasets* response: DatasetDescriptions data structure

Name	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “TJS”	One (mandatory)
version	TJS specification and schema version	Character String type, not empty Value is “1.0”	One (mandatory)
lang	Language of the human-readable text (e.g. “en-CA”)	Character String type, not empty Value is an RFC 4646 language tag	One (mandatory)
capabilities	GetCapabilities request URL for this service implementation	URL type	One (mandatory)
Framework	Description of a spatial framework for which data is available	Partial data structure defined in Table 6, excluding the Framework/Dataset/Columnset and Framework/Dataset/Rowset elements.	One or more (mandatory)
Note 1: The shaded elements in this table are identical to those in Table 43.			

10.3.2 DescribeDatasets normal response XML encoding

The *DescribeDatsets* operation response shall be encoded in XML in accordance with the *tjsDescribeDatasets_response.xsd* XML schema (see Annex B).

10.3.3 DescribeDatasets exceptions

When a TJS server encounters an error while performing a *DescribeDatsets* operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 20 of that document. For each listed exceptionCode, the contents of the “locator” parameter value shall be as specified in the right column of Table 48.

Table 48 — Exception codes for DescribeDatasets operation

exceptionCode value	Meaning of code	“locator” value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit “locator” parameter
NOTE The values listed above are copied from Table 20 in Subclause 8.3 of [OGC 06-121r3].		

11 DescribeData operation (optional)

11.1 Introduction

The *DescribeData* operation returns an XML document that describes all of the attributes for any dataset identified in the request. The response includes descriptive information about each attribute available from the service instance, and information that uniquely identifies that attribute. This description can be used to automatically build a user interface to identify the attribute data that the user would like to obtain from the service.

This operation shall be supported for all servers that implement data access, as identified in section 6.1.1.

11.2 DescribeData operation request

11.2.1 DescribeData request parameters

A request to perform the *DescribeData* operation shall include the parameters listed in Table 49. This table also specifies the source of values and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

Table 49 — Parameters in DescribeData operation request

Names	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “TJS”	One (mandatory)
request	Operation name	Character String type, not empty Value is “DescribeData”)	One (mandatory)
version	Service version identifier	Character String type, not empty Value is “1.0”	Zero or one (optional) When included, return the matching version When omitted, return latest supported version
language	Language identifier for human readable text	Character String type, not empty Value is a two or five character IETF RFC 4646 language identifier	Zero or one (optional) For use see section 7.6
Framework URI	URI of a spatial framework to which the attribute table can be joined.	URI type, not empty Value is a FrameworkURI as found in the *DescribeFrameworks* response.	One (mandatory)
DatasetURI	URI of an attribute table, which can be joined to the spatial framework identified by the *FrameworkURI*.	URI type, not empty Value is a DatasetURI as found in the *DescribeDatasets* response.	One (mandatory)
Attributes	The names of the attributes for which descriptions are requested from the server.	Character String type, not empty Value is one or more Columnset/Attributes/Column names as found in the *DescribeData* response, in comma-delimited format.	Zero or one (optional) [a]

a Attribute names are normally unknown during an initial DescribeData request to the server, and therefore this parameter is normally absent from such a request. If this parameter is absent, the response shall include descriptions of all attributes available for the specified dataset.

NOTE 1 The parameters shaded in gray above are largely copied from Table 3 in Subclause 7.3 of this document.

NOTE 2: The values for all fields shall be encoded using the standard Internet practice for encoding URLs [IETF RFC 1738].

11.2.2 DescribeData request KVP encoding (mandatory)

Servers shall implement HTTP GET transfer of the *DescribeData* operation request, using KVP encoding. The KVP encoding of this request shall use the parameters specified in Table 49.

EXAMPLE A valid DescribeData operation request KVP encoded for HTTP GET is:

```
http://foo.bar/foo?
    Service=TJS&
    Version=1.0&
    Request=DescribeData&
    FrameworkURI=http://foo.bar2/Provinces/2001&
    DatasetURI=http://foo.bar/2001Census/2001&
    Language=en-CA
```

NOTE: For clarity, the parameter values in the above example have not been URL encoded.

11.2.3 DescribeData request XML encoding (optional)

Servers may also implement HTTP POST transfer of the *DescribeData* operation request, using XML encoding only, in accordance with the tjsDescribeData_request.xsd XML schema.

11.3 DescribeData operation response

11.3.1 DescribeData normal response parameters

The normal response to a valid *DescribeData* operation request shall be a DataDescriptions data structure, which contains descriptions of one or more Attributes for which data is available on this service. The structure of this response is shown in Table 50. Note that the Framework and Dataset data structure portions of this response are similar to the equivalent structures in the *DescribeDatasets* response.

Table 50 — *DescribeData* response data structure

Name	Definition	Data type	Multiplicity and use
DataDescriptions	General description of all attribute data available on this server	DataDescriptions data structure, see Table 51	One (mandatory)

Table 51 — *DescribeData* response: DataDescriptions data structure

Name	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is "TJS"	One (mandatory)
version	TJS specification and schema version	Character String type, not empty Value is "1.0"	One (mandatory)

Name	Definition	Data type and values	Multiplicity and use
lang	Language of the human-readable text (e.g. "en-CA")	Character String type, not empty Value is an RFC 4646 language tag	One (mandatory)
capabilities	GetCapabilities request URL for this service implementation	URL type	One (mandatory)
Framework	Description of a spatial framework for which data is available	Partial data structure defined in Table 6, excluding the Framework/Dataset/Rowset element.	One (mandatory)
Note 1: The shaded elements in this table are identical to those in Table 43.			

11.3.2 DescribeData normal response XML encoding

The *DescribeData* operation response shall be encoded in XML in accordance with the *tjsDescribeData_response.xsd* XML schema (see Annex B).

11.3.3 DescribeData exceptions

When a TJS server encounters an error while performing a *DescribeData* operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 20 of that document. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 52.

Table 52 — Exception codes for DescribeDatasets operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
NOTE The values listed above are copied from Table 20 in Subclause 8.3 of [OGC 06-121r3].		

12 GetData operation (optional)

12.1 Introduction

The *GetData* operation provides a means for a client to retrieve attribute data. The response includes descriptive information about each attribute included in the response, as well as the attribute values for all selected records. The attribute values are returned to a client as an XML document, in GDAS format (see section 7.4).

Users of the service can safely assume that any caches or analyses based on the GetData response will need to be updated if and only if a DescribeData request to the service indicates that an updated Version is available for the identified dataset.

This operation shall be supported for all servers that implement data access, as identified in section 6.1.1.

12.2 GetData operation request

12.2.1 GetData request parameters

A request to perform the *GetData* operation shall include the parameters listed in Table 53. This table also specifies the source of values and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

Table 53 — Parameters in GetData operation request

Names	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “TJS”	One (mandatory)
request	Operation name	Character String type, not empty Value is “DescribeData”)	One (mandatory)
version	Service version identifier	Character String type, not empty Value is “1.0”	Zero or one (optional) When included, return the matching version When omitted, return latest supported version
language	Language identifier for human readable text	Character String type, not empty Value is a two or five character IETF RFC 4646 language identifier	Zero or one (optional) For use see section 7.6
Framework URI	URI of a spatial framework to which the attribute table can be joined.	URI type, not empty Value is a FrameworkURI as found in the *DescribeFrameworks* response.	One (mandatory)
DatasetURI	URI of an attribute table, which can be joined to the spatial framework identified by the *FrameworkURI*.	URI type, not empty Value is a DatasetURI as found in the *DescribeDatasets* response.	One (mandatory)
Attributes	The names of the attributes requested from the server.	Character String type, not empty Value is one or more AttributeNames as found in the *DescribeData* response, in comma-delimited format.	Zero or one (optional) [a]
Linkage Keys	The DatasetKey identifiers requested by the user.	Character String type, not empty Value is one or more Identifiers in comma-delimited format, where ranges shall be indicated with a minimum value and maximum value separated by a dash (“-”). The same Identifier cannot be requested multiple times.	Zero or one (optional) [b]
Filter Column	The name of a Nominal or Ordinal field in the dataset upon which to filter the contents of the GetData response.	Character String type, not empty Value is a field name as found in the DescribeDataset response.	Zero or one (optional) [c]

FilterValue	The Nominal or Ordinal value which the contents of the GetData response shall match.	Character String type, not empty Value is a text string found in the dataset field identified by the FilterColumn parameter.	Zero or one (optional) [c]
XSL	The URL of a valid XSLT document that shall be referenced in the response.	URI type, not empty Value is a valid URL for a valid stylesheet that can be applied to the *GetData* response. [d]	Zero or one (optional) [e]
aid	Attribute Identifier. See Table 31 and Table 32	Boolean type Value is "true" or "false". When "true", the "aid" attribute shall be included for each "K" and "V" element.	Zero or one (optional) [f]

a If this parameter is absent, the response shall include all attributes available for the requested dataset (i.e. all dataset columns).

b If this parameter is absent, the response shall include all linkage keys available for the requested dataset (i.e all dataset rows) apart from those identified for filtering by the FilterColumn/FilterValue parameters.

c If either of these parameters is absent, the response shall include all linkage keys available for the requested dataset (i.e all dataset rows) apart from those not identified for inclusion by the LinkageKeys parameter.

d Note that the XSL file must be accessible via the same server and port number as the TJS GetData operation, since cross-domain XSLT references are not supported by most web browsers.

e If this parameter is absent, the response shall include the server's choice of either a reference to a default XSLT document, or no XSLT reference.

f If this parameter is absent, the "K" and "V" elements in the response shall not include the "aid" attribute (i.e the default is "false").

NOTE 1 The parameters shaded in gray above are largely copied from Table 3 in Subclause 7.3 of this document.

NOTE 2: The values for all fields shall be encoded using the standard Internet practice for encoding URLs [IETF RFC 1738].

12.2.2 GetData request KVP encoding (mandatory)

Servers shall implement HTTP GET transfer of the *GetData* operation request, using KVP encoding. The KVP encoding of this request shall use the parameters specified in Table 53.

EXAMPLE A valid GetData operation request KVP encoded for HTTP GET is:

```
http://foo.bar/foo?
    Service=TJS&
    Version=1.0&
    Request=DescribeData&
    FrameworkURI=http://foo.bar2/Provinces/2001&
    DatasetURI=http://foo.bar/2001Census&
    Attributes=cattlecalves,cows&
    LinkageKeys=10-13,24,35,46-48&
    XSL=http://foo.bar/xslt
```

NOTE: For clarity, the parameter values in the above example have not been URL encoded.

12.2.3 GetData request XML encoding (optional)

Servers may also implement HTTP POST transfer of the *GetData* operation request, using XML encoding only, in accordance with the tjsGetData_request.xsd XML schema.

12.3 GetData operation response

12.3.1 GetData normal response parameters

The normal response to a valid *DescribeData* operation request shall be a GDAS data structure, as defined in section 7.4

12.3.2 GetData normal response XML encoding

The *GetData* operation response shall be encoded in XML in accordance with the *tjsGetData_response.xsd* XML schema (see Annex B). The MIMEtype shall be identified as: "MIMEtype = text/xml; subtype=gdas/1.0".

12.3.3 GetData exceptions

When a TJS server encounters an error while performing a *GetData* operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 20 of that document. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 54

Table 54 — Exception codes for GetData operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
InvalidAttributeName	Operation request included an attribute identifier not available for the requested dataset.	Name of invalid attribute.
InvalidKey	Operation request included a Key that does not exist for the requested dataset.	Identifier of invalid Key
NOTE The values listed above are copied from Table 20 in Subclause 8.3 of [OGC 06-121r3].		

13 DescribeJoinAbilities operation (optional)

13.1 Introduction

The *DescribeJoinAbilities* operation returns an XML document that identifies all of the spatial frameworks to which attribute data can be joined by the service instance. This description includes information that uniquely identifies each spatial framework, and descriptive information about each framework. This information can be used to support service discovery and populate service registries.

This operation shall be supported for all servers that implement data joining, as identified in section 6.1.2.

13.2 DescribeJoinAbilities operation request

13.2.1 DescribeJoinAbilities request parameters

A request to perform the *DescribeJoinAbilities* operation shall include the parameters listed in Table 55. This table also specifies the source of values and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

Table 55 — Parameters in DescribeJoinAbilities operation request

Names	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “TJS”	One (mandatory)
request	Operation name	Character String type, not empty Value is “DescribeJoinAbilities”)	One (mandatory)
version	Service version identifier	Character String type, not empty Value is “1.0”	Zero or one (optional) When included, return the matching version When omitted, return latest supported version
language	Language identifier for human readable text	Character String type, not empty Value is a two or five character IETF RFC 4646 language identifier	Zero or one (optional) For use see section 7.6
NOTE 1 The parameters shaded in gray above are largely copied from Table 3 in Subclause 7.3 of this document.			

13.2.2 DescribeJoinAbilities request KVP encoding (mandatory)

Servers shall implement HTTP GET transfer of the *DescribeJoinAbilities* operation request, using KVP encoding. The KVP encoding of this request shall use the parameters specified in Table 55.

EXAMPLE An example DescribeJoinAbilities operation request KVP encoded for HTTP GET is:

```
http://foo.bar/foo?
    Service=TJS&
    Version=1.0&
    Request=DescribeJoinAbilities&
    Language=en-CA
```

13.2.3 DescribeJoinAbilities request XML encoding (optional)

Servers may also implement HTTP POST transfer of the *DescribeJoinAbilities* operation request, using XML encoding only, in accordance with the *tjsDescribeJoinAbilities_request.xsd* XML schema.

13.3 DescribeJoinAbilities operation response

13.3.1 DescribeJoinAbilities normal response parameters

The normal response to a valid *DescribeJoinAbilities* operation request shall be a JoinAbilities data structure, which contains descriptions of one or more Frameworks to which data tables can be joined by this server. The structure of this response is shown in Table 56.

Table 56 — DescribeJoinAbilities response data structure

Name	Definition	Data type	Multiplicity and use
JoinAbilities	Full description of all spatial frameworks to which data can be joined	FrameworkDescriptions data structure, see Table 57	One (mandatory)

Table 57 — *DescribeJoinAbilities* response: JoinAbilities data structure

Name	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “TJS”	One (mandatory)
version	TJS specification and schema version	Character String type, not empty Value is “1.0”	One (mandatory)
lang	Language of the human-readable text (e.g. “en-CA”)	Character String type, not empty Value is an RFC 4646 language tag	One (mandatory)

Name	Definition	Data type and values	Multiplicity and use
capabilities	GetCapabilities request URL for this service implementation	URL type	One (mandatory)
Update Supported	Boolean that identifies if existing JoinData products can be updated by this service. If "true" then subsequent identical JoinData requests will update existing JoinData products that were created by this service. These updated products will then be available via the existing URLs of those products.	Boolean "true" or "false"	One (mandatory)
Spatial Frameworks	Full description of all spatial frameworks to which attribute data can be joined.	SpatialFrameworks data structure, see Table 58	One (mandatory)
AttributeLimit	Maximum number of attributes that can be joined simultaneously as part of a JoinData request.	Positive Integer	One (mandatory)
Output Mechanisms	List of mechanisms by which the attribute data will be accessible once it has been joined to the spatial framework.	OutputMechanisms data structure, see Table 59	One (mandatory)
OutputStylings	Unordered list of display or content styling instructions supported by the server and that can be applied if the AccessMechanisms of the requested output includes WMS.	OutputStyling data structure, see Table 61	Zero or one (optional) [a]
Classification SchemaURL	URL that returns an XML Schema document specifying an XML structure for describing data classifications. When included with a JoinData request, the contents of such an XML file can be applied by the server in order to produce the data classes used in the JoinData output.	URI type Value is a valid URL for a XML schema definition of the classification document	Zero or one (optional) [b]

a This element shall be included when the server supports one or more WMS styling mechanisms. If WMS is not supported by the server then this element shall not be present. If WMS is supported and this element is not present, a default styling will be applied to all WMS outputs.

b This element shall be included when the server supports a mechanism for classifying data. This standard does not define any such mechanisms, so their definition may be specific to an individual server.

Note 1: The shaded elements in this table are identical to those in Table 41 for the *DescribeFrameworks* request.

Table 58 — *DescribeJoinAbilities* response: <u>SpatialFrameworks</u> data structure

Name	Definition	Data type and values	Multiplicity and use
Framework	Description of a spatial framework for which data is available.	Partial data structure defined in Table 6 — GDAS data encoding: <u>Framework</u> data structure, excluding the Framework/Dataset element.	One or more (mandatory)
Note 1: The shaded element in this table is identical to that in Table 43 for the *DescribeFrameworks* response.			

Table 59 — *DescribeJoinAbilities* response: <u>OutputMechanisms</u> data structure

Name	Definition	Data type and values	Multiplicity and use
Mechanism	Mechanism by which the attribute data can be accessed once it has been joined to the spatial framework	Mechanism data structure, see Table 60	One or more (mandatory)

Table 60 — *DescribeJoinAbilities* response: <u>Mechanism</u> data structure

Name	Definition	Data type and values	Multiplicity and use
Identifier	Name that uniquely identifies this type of access mechanism supported by this server.	Character string type, not empty.	One (mandatory)
Title	Human-readable title that uniquely identifies the type of access mechanism supported by this server. Must be suitable for display in a pick-list to a user.	Character string type, not empty.	One (mandatory)
Abstract	Human-readable description of the type of access mechanism, suitable for display to a user seeking information about this type of access mechanism.	Any (Character String type or XHTML)	One (mandatory)
Reference	URL that defines the access mechanism.	URI type	One (mandatory)

Table 61 — *DescribeJoinAbilities* response: <u>OutputStylings</u> data structure

Name	Definition	Data type and values	Multiplicity and use
Styling	Describes a form of styling instruction supported by this server.	Styling data structure, see Table 62	One or more (mandatory)

Table 62 — *DescribeJoinAbilities* response: Styling data structure

Name	Definition	Data type and values	Multiplicity and use
Identifier	Name that uniquely identifies the type of styling instructions supported by this server.	Character string type, not empty.	One (mandatory)
Title	Human-readable title that uniquely identifies the type of styling instructions supported by this server. Must be suitable for display in a pick-list to a user.	Character string type, not empty.	One (mandatory)
Abstract	Human-readable description of the type of styling instructions, suitable for display to a user seeking information about this type of styling instruction.	Any (Character String type or XHTML)	One (mandatory)
Reference	URL that defines the styling instructions.	URI type Value is a valid URL for a human-readable definition of the styling instructions.	One (mandatory)
Schema	Reference to a definition of XML elements or types supported for this styling instruction (e.g., a URL which returns the XSD for SLD 1.0).	URI type. Value is a valid URL for a valid XML schema that can be used to validate styling instructions.	Zero or one (optional) [a]

a This element shall be included when the styling instructions are XML encoded using an XML schema. When included, the styling instructions shall validate against the referenced XML Schema. This element shall be omitted if this form of styling instruction is not encoded in XML or there is no XML Schema available. If the encoding uses a profile of a larger schema, the server administrator should provide that schema profile for validation purposes.

13.3.2 DescribeJoinAbilities normal response XML encoding

The *DescribeJoinAbilities* operation response shall be encoded in XML in accordance with the *tjsDescribeJoinAbilities_response.xsd* XML schema (see Annex B).

13.3.3 DescribeJoinAbilities exceptions

When a TJS server encounters an error while performing a *DescribeJoinAbilities* operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 20 of that document. For each listed exceptionCode, the contents of the “locator” parameter value shall be as specified in the right column of Table 63.

Table 63 — Exception codes for DescribeJoinAbilities operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
NOTE The values listed above are copied from Table 20 in Subclause 8.3 of [OGC 06-121r3].		

14 DescribeKey operation (optional)

14.1 Introduction

The *DescribeKey* operation returns an XML document that identifies all of the keys for a spatial framework to which data can be joined by the service instance. This description also includes descriptive information about the spatial framework. This listing can be used to debug the *JoinData* operation, or to build a new attribute dataset for the spatial framework.

This operation shall be supported for all servers that implement data joining, as identified in section 6.1.2.

14.2 DescribeKey operation request

14.2.1 DescribeKey request parameters

A request to perform the *DescribeKey* operation shall include the parameters listed in Table 64. This table also specifies the source of values and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

Table 64 — Parameters in DescribeKey operation request

Names	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is "TJS"	One (mandatory)
request	Operation name	Character String type, not empty Value is "DescribeKey")	One (mandatory)
version	Service version identifier	Character String type, not empty Value is "1.0"	Zero or one (optional) When included, return the matching version When omitted, return latest supported version
language	Language identifier for human readable text	Character String type, not empty Value is a two or five character IETF RFC 4646 language identifier	Zero or one (optional) For use see section 7.6
Framework URI	The URI of the spatial framework.	URI type Value is a FrameworkURI found in the *DescribeJoinAbilities* response.	One (mandatory)
NOTE 1 The parameters shaded in gray above are largely copied from Table 3 of this document.			

14.2.2 DescribeKey request KVP encoding (mandatory)

Servers shall implement HTTP GET transfer of the *DescribeKey* operation request, using KVP encoding. The KVP encoding of this request shall use the parameters specified in Table 64.

EXAMPLE An example DescribeKey operation request KVP encoded for HTTP GET is:

```
http://foo.bar/foo?
    Service=TJS&
    Version=1.0&
    Request=DescribeKey&
    FrameworkURI=http://foo.bar/foo&
    Language=en-CA
```

NOTE: For clarity, the parameter values in the above example have not been URL encoded.

14.2.3 DescribeKey request XML encoding (optional)

Servers may also implement HTTP POST transfer of the *DescribeKey* operation request, using XML encoding only, in accordance with the *tjsDescribeKey_request.xsd* XML schema.

14.3 DescribeKey operation response

14.3.1 DescribeKey normal response parameters

The normal response to a valid *DescribeKey* operation request shall be a FrameworkKeyDescription data structure, which lists the entire contents of the FrameworkKey field for which the *JoinData* operation is supported on the server. The structure of this response is shown in Table 65.

Table 65 — DescribeKey response data structure

Name	Definition	Data type	Multiplicity and use
FrameworkKey Description	Response containing full description the FrameworkKey of a spatial framework housed on this server.	FrameworkKeyDescription data structure, see Table 66	One (mandatory)

Table 66 — *DescribeKey* response: FrameworkKeyDescription data structure

Name	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty	One (mandatory) Value is “TJS”
version	TJS specification and schema version	Character String type, not empty	One (mandatory) Value is “1.0”
lang	RFC 4646 language tag of the human-readable text (e.g. “en-CA”)	Character String type, not empty	One (mandatory)
capabilities	GetCapabilities request URL for this service implementation	URL type	One (mandatory)
Framework	Description of a spatial framework for which data is available	Framework data structure, see Table 67	One (mandatory)
Note 1: The shaded elements in this table are identical to those in Table 43 for the *DescribeFrameworks* response.			

Table 67 — *DescribeKey* response: Framework data structure

Name	Definition	Data type and values	Multiplicity and use
FrameworkURI	The URI that uniquely references the spatial framework. Normally a URL, but a URN may be used.	Character String type, not empty	One (mandatory) Example: "http://agr.gc.ca/slc/v1"
Organization	The name of the organization that is responsible for maintaining the framework dataset.	Character String type, not empty	One (mandatory)
Title	A human readable sentence fragment that might form a title if the framework dataset were displayed in map form.	Character String type, not empty	One (mandatory)
Abstract	A complete description or abstract that describes the framework dataset	Any (Character String type or XHTML)	One (mandatory)
ReferenceDate	The date to which the framework dataset applies.	Character String type, not empty; ReferenceDate data structure, see Table 7	One (mandatory)
Documentation	URL reference to a web-accessible resource which contains further information describing this framework.	URL type	Zero or one (optional) [a]
FrameworkKey	The key field within the framework dataset (i.e. in the GIS) that is referenced by all datasets that apply to this Framework.	Character String type, not empty. See Table 8.	One (mandatory)
Bounding Coordinates	Bounding Coordinates of the spatial framework.	BoundingCoordinates data structure, see Table 10	One (mandatory)
Columnset	Structure containing descriptions of the FrameworkKey columns.	Columnset data structure, see Table 13, excluding the Attributes data structure	One (mandatory)
Rowset	Ordered list of all the spatial features for the identified framework.	Rowset data structure, see Table 68	One (mandatory)
a This element should be included when web pages describing this framework are available. Note 1: The shaded elements in this table are identical to those in Table 6.			

Table 68 — *DescribeKey* response: Rowset data structure

Name	Definition	Data type	Multiplicity and use
Row	Feature found in the spatial framework.	Row data structure, see Table 69	One or more (mandatory)

Table 69 — *DescribeKey* response: Row data structure

Name	Definition	Data type	Multiplicity and use
K	Spatial Key (Identifier)	Character String type, not empty	One or more (mandatory)
Title	Human-readable short description suitable to display on a pick-list, legend, and/or on mouse-over.	Character String type, not empty	Zero or one (optional) [a]
a This element should be included when a title is available for each feature comprising the spatial framework.			

14.3.2 DescribeKey normal response XML encoding

The *DescribeKey* operation response shall be encoded in XML in accordance with the *tjsDescribeKey_response.xsd* XML schema (see Annex B).

14.3.3 DescribeKey exceptions

When a TJS server encounters an error while performing a *DescribeKey* operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 20 of that document. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 70.

Table 70 — Exception codes for DescribeKey operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
NOTE The values listed above are copied from Table 20 in Subclause 8.3 of [OGC 06-121r3].		

15 JoinData operation (optional)

15.1 Introduction

The *JoinData* operation instructs the server to merge an attribute data table encoded in GDAS format (i.e. obtainable via a GetData request) with its spatial framework. The server performs the join and prepares the output in the form requested by the client. The response includes the connection information required to access the output(s).

This operation shall be supported for all servers that implement data joining, as identified in section 6.1.2.

15.2 JoinData operation request

15.2.1 JoinData request parameters

A request to perform the *JoinData* operation shall include the parameters listed in Table 71. This table also specifies the source of values and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

Table 71 — Parameters in JoinData operation request

Names	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “TJS”	One (mandatory)
request	Operation name	Character String type, not empty Value is “DescribeKey”)	One (mandatory)
version	Service version identifier	Character String type, not empty Value is “1.0”	Zero or one (optional) When included, return the matching version When omitted, return latest supported version
language	Language identifier for human readable text	Character String type, not empty Value is a two or five character IETF RFC 4646 language identifier	Zero or one (optional) For use see section 7.6
update	Identifies if JoinData products should by updated/updatable.	Boolean type Value is “true” or “false”. Default is “false”.	Zero or one (optional) [a]
Framework URI	The URI of the spatial framework.	URI type Value is a FrameworkURI found in the *DescribeJoinAbilities* response.	One (mandatory)
GetDataURL	A URL which returns the attribute data table encoded in GDAS format.	URI type Value is a URL that returns a valid GDAS document compliant with this version of this specification. [b]	One (mandatory)
StylingURL	URL that contains the styling information to be applied to any WMS output.	URI type Value is a URL that returns valid styling information.	Zero or one (optional) [c]
Styling Identifier	Name that identifies the type of styling to be invoked.	Character String type, not empty Value is an Identifier listed in the *DescribeJoinAbilities* response.	Zero or one (optional) [d]
Classification URL	URL that returns a file describing a data classification to be applied to the output (e.g. the classification to be used for a legend in the case where the output is a WMS). [e]	URI type Value is a URL that returns valid classification information.	Zero or one (optional) [f]

a Include when the user will wish to update the products of this request in the future and requires that the URLs of the output products shall not change. If this parameter is true then the service shall attempt to update an existing set of JoinData products produced earlier by an earlier identical request. If no such products exist it shall prepare JoinData products for this request in such a way that they can be updated by subsequent identical JoinData requests. URLs for such products shall not vary for subsequent update requests. b Note that this may be a TJS GetData request (via HTTP GET), a stored response to a GetData request, or a web process that returns content compliant with the GetData response schema. If more than one attribute of type “Data” exists in the response to this URL, the server shall apply the JoinData operation to the first attribute of type “Data”. c If this parameter is absent, the server will apply its default styling to any WMS output. d If this parameter is absent, the server will assume that the StylingURL, if present, is the default form of styling supported by the server. e This file must be encoded in compliance with the XML Schema identified in the ClassificationSchemaURL element of the DescribeJoinAbilities response. f If this parameter is absent, the server will apply a default classification as determined by the server.
NOTE 1 The parameters shaded in gray above are largely copied from Table 3 of this document.

15.2.2 JoinData request KVP encoding (mandatory)

Servers shall implement HTTP POST transfer of the *JoinData* operation request, using KVP encoding. The KVP encoding of this request shall use the parameters specified in Table 71.

EXAMPLE An example JoinData operation request KVP encoded for HTTP POST is:

```
Service=TJS&
Version=1.0&
Request=JoinData&
Language=en-CA&
FrameworkURI=http://foo.bar/foo&
GetDataURL=http://foo.bar2/foo&
StylingURL=http://foo.bar3/foo&
StylingIdentifier=SLD_1.0
```

NOTE: For clarity, the parameter values in the above example have not been URL encoded.

15.2.3 JoinData request XML encoding (optional)

Servers may also implement HTTP POST transfer of the *JoinData* operation request, using XML encoding, in accordance with the *tjsJoinData_request.xsd* XML schema. Note that the use of this XML encoding provides additional capabilities for this request by providing options for the GetData portion of the *JoinData* request via either HTTP GET or POST. The structure of this request is shown in Table 72.

Table 72 — *JoinData* request data structure

Name	Definition	Data type	Multiplicity and use
JoinData	Request to a TJS to perform the JoinData operation. This operation allows a client to join attribute data to a spatial framework. In this XML encoding, no "request" parameter is included, since the element name specifies the specific operation.	JoinData data structure, see Table 73	One (mandatory)

Table 73 — *JoinData* request: JoinData data structure

Name	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is "TJS"	One (mandatory)
request	Operation name	Character String type, not empty Value is "JoinData")	One (mandatory)
version	Service version identifier	Character String type, not empty Value is "1.0"	Zero or one (optional) When included, return the matching version When omitted, return latest supported version
language	Language identifier for human readable text	Character String type, not empty Value is a two or five character IETF RFC 4646 language identifier	Zero or one (optional) For use see section 7.6
FrameworkURI	The URI that uniquely references the spatial framework. Normally a URL, but a URN may be used.	Character String type, not empty	One (mandatory)
AttributeData	Attribute data to be joined to the spatial framework.	AttributeData data structure, see Table 74	One (mandatory)
MapStyling	Styling that shall be applied if the AccessMechanisms of the requested output includes WMS. If WMS is not supported, this element shall not be present. If WMS is supported and this element is not present, a default styling will be applied to the WMS layer.	WebMapStyling data structure, see Table 76	Zero or one (optional)
Note 1: The shaded elements in this table are identical to those in Table 41 for the *DescribeFrameworks* request.			

Table 74 — *JoinData* request: <u>AttributeData</u> data structure

Name	Definition	Data type	Multiplicity and use
GetDataURL	URL which returns a valid TJS 1.0 GetData response. Note that this may be a TJS GetData request (via HTTP GET), a stored response to a GetData request, or a web process that returns content compliant with the GetData response schema.	URI type	Zero or one (conditional) [a]
GetDataXML	GetData request in XML encoding, including the name of the TJS server to be queried. Note that since XML encoding of the GetData request is optional for TJS servers, this choice should not be used unless it is known that the TJS server supports this request method.	GetDataXML data structure, see Table 75	Zero or one (conditional) [a]
a One and only one of these two items shall be included.			

Table 75 — *JoinData* request: <u>GetDataXML</u> data structure

Name	Definition	Data type	Multiplicity and use
getDataHost	Base URL of the TJS server to which the GetData XML encoded request shall be passed.	URI type	One (mandatory)
GetData	Contents of the GetData request that shall be passed to the getDataHost URL in order to obtain the attribute data to be joined to its spatial framework.	GetData operation request structure, but excluding the “aid” and “XSL” parameters, see Table 53 and section 12.2.3.	One (mandatory)

Table 76 — *JoinData* request: <u>WebMapStyling</u> data structure

Name	Definition	Data type	Multiplicity and use
Styling Identifier	Name that identifies the type of styling to be invoked. Must be an Identifier listed in the DescribeJoinAbilities response.	Character String type, not empty	One (mandatory)
StylingURL	Reference to a web-accessible resource that contains the styling information to be applied. This attribute shall contain a URL from which this input can be electronically retrieved.	URI type	One (mandatory)

15.3 JoinData operation response

15.3.1 JoinData normal response parameters

The normal response to a valid *JoinData* operation request shall be a JoinDataResponse data structure, which contains a copy of the request, execution status, and references to the outputs created by the service. The structure of this response is shown in Table 77.

Table 77 — *JoinData* response data structure

Name	Definition	Data type	Multiplicity and use
JoinData Response	Response to a JoinData request.	JoinDataResponse data structure, see Table 78	One (mandatory)

Table 78 — *JoinData* response: <u>JoinDataResponse</u> data structure

Name	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty	One (mandatory) Value is "TJS"
version	TJS specification and schema version	Character String type, not empty	One (mandatory) Value is "1.0"
lang	Language of the human-readable text (e.g. "en-CA")	Character String type, not empty	One (mandatory) RFC 4646 language tag
capabilities	GetCapabilities request URL for this service implementation	URL type	One (mandatory)
Status	Execution status of the JoinData request.	Status data structure, see Table 79	One (mandatory)
DataInputs	Descriptions of the framework, dataset, and attributes used to generate the outputs of the JoinData operation.	Framework data structure, see Table 80	One (mandatory)
JoinedOutputs	List of outputs resulting from the JoinData operation. There must be at least one output when the operation has completed successfully.	JoinedOutputs data structure, see Table 81	One (mandatory)
Note 1: The shaded elements in this table are identical to those in Table 43 for the *DescribeFrameworks* response.			

Table 79 — *JoinData* response: Status data structure

Name	Definition	Data type and values	Multiplicity and use
href	HTTP reference to location where current JoinDataResponse document is stored	URL type	One (mandatory) [a]
creationTime	The time (UTC) that the JoinData operation finished. If the operation is still in progress, this element shall contain the creation time of this document.	dateTime type	One (mandatory)
Accepted	Indicates that this request has been accepted by the server, but has not yet completed.	Character string type, not empty [c]	Zero or one (conditional) [b]
Completed	Indicates that this request has completed execution with at lease partial success.	Character string type, not empty [d]	Zero or one (conditional) [b]
Failed	Indicates that execution of the JoinData operation failed, and includes error information.	Character string type, not empty [e]	Zero or one (conditional) [b]

a The JoinDataResponse should be found at this URL as soon as the process returns the initial response to the client. It should persist at this location for as long as the outputs are accessible from the server. The outputs may be stored for as long as the implementer of the server decides. If the JoinData operation takes a long time, this URL can be repopulated on an ongoing basis in order to keep the client updated on progress. Before the operation has completerd (or failed), the JoinDataResponse contains information about the status of the operation. It may also optionally contain any output results. When the JoinData operation has completed, the JoinDataResponse found at this URL shall contain information for all of the outputs.

b One and only one of these three elements can be present

c The contents of this human-readable text string is left open to definition by each server implementation, but is expected to include any messages the server may wish to let the clients know. Such information could include when completion is expected, or any warning conditions that may have been encountered. The client may display this text to a human user.

d The contents of this human-readable text string is left open to definition by each server, but is expected to include any messages the server may wish to let the client know, such as how long the operation took to execute, or any warning conditions that may have been encountered. The client may display this text string to a human user. The client should make use of the presence of this element to trigger automated or manual access to the results of the operation. If manual access is intended, the client should use the presence of this element to present the results as downloadable links to the user.

e The client may display this text string to a human user. The presence of this element indicates that the operation completely failed and no Outputs were produced.

Table 80 — *JoinData* response: Framework data structure

Name	Definition	Data type	Multiplicity and use
Framework	Description of the spatial framework and the attributes now joined to it.	Framework data structure defined in Table 6, excluding the Framework/Dataset/Rowset element.	One (mandatory)

Table 81 — *JoinData* response: JoinedOutputs data structure

Name	Definition	Data type and values	Multiplicity and use
Output	Unordered list of all the outputs that have been or will be produced by this operation.	Output data structure, see Table 82	One or more (mandatory)
Note 1: The number of occurrences of this element is dependent on the number of attributes in the JoinData request as well as the number and type of AccessMechanisms requested. There is one Output element required for each Mechanism requested. For some type of Mechanism, there is one Output element required for each attribute in the JoinData request, while others require only a single Output incorporates all of the elements included in the request. For example, where WMS is requested as an output mechanism, each attribute identified in the request might be used to create a separate WMS layer, and thus each layer would be represented by the contents of a separate Output element. Where GIS is requested as an output mechanism, all of the attributes might be bundled into a single GIS file, thus requiring only one Output element. This specification does not provide guidance on this aspect beyond the requirement that all outputs must be identified in this data structure.			

Table 82 — *JoinData* response: Output data structure

Name	Definition	Data type and values	Multiplicity and use
Mechanism	Mechanism by which the joined attribute data is available	Mechanism data structure, see Table 60	One (mandatory)
Resource	Reference to a web-accessible resource that was created by the JoinData operation.	Resource data structure, see Table 83	Zero or one (conditional) [a]
Exception Report	Unordered list of one or more errors encountered during the JoinData operation for this output. These Exception elements shall be interpreted by clients as being independent of one another (not hierarchical). This element is populated when the production of this output did not succeed.	Exception data structure, see Table 85 [b]	Zero or one (conditional) [a]
a One and only one of these two elements shall be present. b This is similar to an OWS Common ExceptionReport as defined in [OGC 06-121r3], except that it is not stand-alone.			

Table 83 — *JoinData* response: Resource data structure

Name	Definition	Data type and values	Multiplicity and use
URL	URL from which this resource can be electronically retrieved, or from which a document can be retrieved that indicates access details for the resource (such as an OGC Capabilities document).	URL type	One (mandatory) For OGC web services this shall be the complete GetCapabilities URL.

Name	Definition	Data type and values	Multiplicity and use
Parameter	Parameter that may need to be included the HTTP requests to a web service identified by the URL parameter above.	Character String type, not empty; with Parameter data structure, see Table 84	Zero or more (optional) For a WMS output, there shall be one occurrence of this element, and it shall be populated with the name of the layer produced by the JoinData operation.

Table 84 — *JoinData* response: <u>Parameter</u> data structure

Name	Definition	Data type and values	Multiplicity and use
name	Identifier for this parameter as defined by the service delivering the output of the JoinData operation.	Character String type, not empty	One (mandatory) For a WMS output this attribute shall be populated with the string "layers"

Table 85 — *JoinData* response: <u>ExceptionReport</u> data structure

Name	Definition	Data type and values	Multiplicity and use
Exception	Error encountered during processing that prevented successful production of this output.	Exception data structure, see Subclause 8 of OWS Common [OGC 06-121r3]	One or more (mandatory)

15.3.2 JoinData normal response XML encoding

The *JoinData* operation response shall be encoded in XML in accordance with the *tjsJoinData_response.xsd* XML schema (see Annex B).

15.3.3 JoinData exceptions

When a TJS server encounters an error while performing a *JoinData* operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 20 of that document. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 86.

Table 86 — Exception codes for DescribeJoinAbilities operation

exceptionCode value	Meaning of code	“locator” value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit “locator” parameter
InvalidFramework	The Framework identified in the GetData request did not match a Framework available for the JoinData operation on this server.	None, omit “locator” parameter
GetDataFailed	The Getdata request failed.	None, omit “locator” parameter
InvalidKey	The Join operation was unable to complete because the contents of the relate keys did not match correctly.	None, omit “locator” parameter
NOTE The values listed above are copied from Table 20 in Subclause 8.3 of [OGC 06-121r3].		

Annex A
(normative)

Abstract test suite

A.1 General

This abstract test suite specifies at a high level how server and client implementations of this standard shall be tested for conformance to this standard. The framework for such abstract test suites is specified in ISO 19105: Geographic information – Conformance and testing, especially Clauses 7 and 9.

An abstract test suite contains multiple abstract tests, grouped into one or more test modules. This abstract test suite consists of two top-level test modules:

a) Client test module – Abstract tests for checking conformance of client implementations with the requirements of this standard that are normatively referenced by an OWS Implementation Specification.

b) Server test module – Abstract tests for checking conformance of server implementations with the requirements of this standard that are normatively referenced by an OWS Implementation Specification.

Any of these modules could contain lower-level test modules. At this time, there are no lower-level test modules.

In the client and server test modules, all operations specified and implemented shall be tested, including KVP HTTP GET and SOAP HTTP POST transfer of each operation request. In the standard test module, all operations specified shall be tested, including KVP HTTP GET and SOAP HTTP POST transfer of operation requests. All mandatory operation request and response parameters specified or implemented shall be tested. Any optional item implemented by a server shall be tested. All items implemented by a client shall be tested.

A.2 Client test module

A.2.1 GetCapabilities operation request

a) Test Purpose: Verify that a client satisfies all requirements for a GetCapabilities operation request.

b) Test Method: Generate an adequate sample of GetCapabilities operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 8.2

d) Test Type: Basic

A.2.2 DescribeFrameworks operation request

a) Test Purpose: Verify that a client satisfies all requirements for a DescribeFrameworks operation request.

b) Test Method: Generate an adequate sample of DescribeFrameworks operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 9.2

d) Test Type: Basic

A.2.3 DescribeFrameworks operation request

a) Test Purpose: Verify that a client satisfies all requirements for a DescribeFrameworks operation request.

b) Test Method: Generate an adequate sample of DescribeFrameworks operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 10.2

d) Test Type: Basic

A.2.4 DescribeData operation request

a) Test Purpose: Verify that a client satisfies all requirements for a DescribeData operation request.

b) Test Method: Generate an adequate sample of DescribeData operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 11.2

d) Test Type: Basic

A.2.5 GetData operation request

a) Test Purpose: Verify that a client satisfies all requirements for a GetData operation request.

b) Test Method: Generate an adequate sample of GetData operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 12.2

d) Test Type: Basic

A.2.6 DescribeJoinAbilities operation request

a) Test Purpose: Verify that a client satisfies all requirements for a DescribeJoinAbilities operation request.

b) Test Method: Generate an adequate sample of DescribeJoinAbilities operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 13.2

d) Test Type: Basic

A.2.7 DescribeKey operation request

a) Test Purpose: Verify that a client satisfies all requirements for a DescribeKey operation request.

b) Test Method: Generate an adequate sample of DescribeKey operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 14.2

d) Test Type: Basic

A.2.8 JoinData operation request

a) Test Purpose: Verify that a client satisfies all requirements for a JoinData operation request.

b) Test Method: Generate an adequate sample of JoinData operation requests from the client, and verify that each is a valid request.

c) Reference: Subclause 15.2

d) Test Type: Basic

A.3 Server test module

A.3.1 GetCapabilities operation response

a) Test Purpose: Verify that a server satisfies all requirements for a GetCapabilities operation response.

b) Test Method: Submit an adequate sample of GetCapabilities operation requests, and verify that in each case the server provides a valid response.

c) Reference: Subclause 8.3

d) Test Type: Basic

A.3.2 DescribeFrameworks operation response

a) Test Purpose: Verify that a server satisfies all requirements for a DescribeFrameworks operation response.

b) Test Method: Submit an adequate sample of DescribeFrameworks operation requests, and verify that in each case the server provides a valid response.

c) Reference: Subclause 9.3

d) Test Type: Basic

A.3.3 DescribeDatasets operation response

a) Test Purpose: Verify that a server satisfies all requirements for a DescribeDatasets operation response.

b) Test Method: Submit an adequate sample of DescribeDatasets operation requests, and verify that in each case the server provides a valid response.

c) Reference: Subclause 10.3

d) Test Type: Basic

A.3.4 DescribeData operation response

a) Test Purpose: Verify that a server satisfies all requirements for a DescribeData operation response.

b) Test Method: Submit an adequate sample of DescribeData operation requests, and verify that in each case the server provides a valid response.

c) Reference: Subclause 11.3

d) Test Type: Basic

A.3.5 GetData operation response

a) Test Purpose: Verify that a server satisfies all requirements for a GetData operation response.

b) Test Method: Submit an adequate sample of GetData operation requests, and verify that in each case the server provides a valid response.

c) Reference: Subclause 12.3

d) Test Type: Basic

A.3.6 DescribeJoinAbilities operation response

a) Test Purpose: Verify that a server satisfies all requirements for a DescribeJoinAbilities operation response.

b) Test Method: Submit an adequate sample of DescribeJoinAbilities operation requests, and verify that in each case the server provides a valid response.

c) Reference: Subclause 13.3

d) Test Type: Basic

A.3.7 DescribeKey operation response

a) Test Purpose: Verify that a server satisfies all requirements for a DescribeKey operation response.

b) Test Method: Submit an adequate sample of DescribeKey operation requests, and verify that in each case the server provides a valid response.

c) Reference: Subclause 14.3

d) Test Type: Basic

A.3.8 JoinData operation response

a) Test Purpose: Verify that a server satisfies all requirements for a JoinData operation response.

b) Test Method: Submit an adequate sample of JoinData operation requests, and verify that in each case the server provides a valid response.

c) Reference: Subclause 15.3

d) Test Type: Basic

Annex B
(normative)

XML Schema Documents

In addition to this document, this standard includes several normative XML Schema Documents. These XML Schema Documents are bundled in a zip file with the present document. After OGC acceptance of a Version 1.0 of this standard, these XML Schema Documents will also be posted online at the URL http://schemas.opengeospatial.net/TJS/1.0.0/. In the event of a discrepancy between the bundled and online versions of the XML Schema Documents, the online files shall be considered authoritative.

The XML Schema Documents are named:

tjsAll.xsd

tjsDescribeDatasets_request.xsd

tjsDescribeDatasets_response.xsd

tjsDescribeData_request.xsd

tjsDescribeData_response.xsd

tjsDescribeFrameworks_request.xsd

tjsDescribeFrameworks_response.xsd

tjsDescribeJoinAbilities_request.xsd

tjsDescribeJoinAbilities_response.xsd

tjsDescribeKey_request.xsd

tjsDescribeKey_response.xsd

tjsGetCapabilities_request.xsd

tjsGetCapabilities_response.xsd

tjsGetData_request.xsd

tjsGetData_response.xsd

tjsJoinData_request.xsd

tjsJoinData_response.xsd

tjsService.xsd

The TJS XML Schema Documents use and build on the OWS common XML Schema Documents specified [OGC 06-121r3], named:

ows19115subset.xsd

owsCommon.xsd

owsDataIdentification.xsd

owsExceptionReport.xsd

owsGetCapabilities.xsd

owsOperationsMetadata.xsd

owsServiceIdentification.xsd

owsServiceProvider.xsd

All these XML Schema Documents contain documentation of the meaning of each element and attribute, and this documentation shall be considered normative as specified in Subclause 11.6.3 of [OGC 06-121r3].

Annex C
(informative)

Example XML documents

C.1 Introduction

This annex provides an example of a GetData response XML document. Other examples of TJS requests and responses are bundled in a zip file with the present document. After OGC acceptance of a Version 1.0 of this standard, these examples will also be posted online at the URL http://schemas.opengeospatial.net/TJS/1.0.0/examples/.

C.2 GetData response document

```
<GDAS service="TJS" version="1.0"
capabilities="http://sis.agr.gc.ca/pls/meta/tjs_1x0_getcapabilities" xml:lang="en"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.opengis.net/tjs/1.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/tjs/1.0 ../tjsGetData_response.xsd">
	<Framework>
		<FrameworkURI>http://sis.agr.gc.ca/cansis/nsdb/ecostrat/zone/v1</FrameworkURI>
		<Organization>Environment Canada</Organization>
		<Title>Ecozones of Canada</Title>
		<Abstract>Ecozones of Canada, forming part of the Terrestrial Ecological Stratification
of Canada </Abstract>
		<ReferenceDate startDate="1995-01-01">1995-06-03</ReferenceDate>
		<Version>1</Version>
		<Documentation>http://sis.agr.gc.ca/cansis/nsdb/ecostrat/hierarchy.html
</Documentation>
		<FrameworkKey>
			<Column name="ecozone" type="http://www.w3.org/TR/xmlschema-2/#integer"
length="2" decimals="0"/>
		</FrameworkKey>
		<BoundingCoordinates>
			<North>90</North>
			<South>43</South>
			<East>-50</East>
			<West>-145</West>
		</BoundingCoordinates>
		<DescribeDatasetsRequest
xlink:href="http://sis.agr.gc.ca/pls/meta/tjs_1x0_describedatasets?Service=TJS&amp;Version=1.0
&amp;Request=DescribeDatasets&amp;FrameworkURI=http://sis.agr.gc.ca/cansis/nsdb/ecostrat/
zone/v1&amp;Language=en"/>
		<Dataset>
			<DatasetURI>http://sis.agr.gc.ca/cansis/nsdb/ecostrat/zone/v1/population_1991
</DatasetURI>
			<Organization>Agriculture and Agri-Food Canada</Organization>
			<Title>Population 1991</Title>
			<Abstract>The population statistics are a special tabulation prepared for the State
of the Environment Directorate, Environment Canada, by Statistics Canada, based on the 1991
```

```
Census of Canada. Population data was not compiled for the ecodistrict level of the framework.
The population attributes compiled are identified below.</Abstract>
            <ReferenceDate>1999-09-16</ReferenceDate>
            <Version>1</Version>
            <Documentation/>
            <DescribeDataRequest
xlink:href="http://sis.agr.gc.ca/pls/meta/tjs_1x0_describedata?Service=TJS&amp;Version=1.0&a
mp;Request=DescribeData&amp;FrameworkURI=http://sis.agr.gc.ca/cansis/nsdb/ecostrat/zone/v
1&amp;DatasetURI=http://sis.agr.gc.ca/cansis/nsdb/ecostrat/zone/v1/population_1991&amp;
Language=en"/>
            <Columnset>
                <FrameworkKey complete="true" relationship="one">
                    <Column name="ecozone" type="http://www.w3.org/TR/xmlschema-
2/#integer" length="2" decimals="0"/>
                </FrameworkKey>
                <Attributes>
                    <Column name="rurf_91" type="http://www.w3.org/TR/xmlschema-
2/#integer" length="10" decimals="0" purpose="Attribute">
                        <Title>Rural Female Population</Title>
                        <Abstract>Number of females living in rural areas.</Abstract>
                        <Documentation>
http://sis.agr.gc.ca/cansis/nsdb/ecostrat/population.html</Documentation>
                        <Values>
                            <Count>
                                <UOM>
                                    <ShortForm>people</ShortForm>
                                    <LongForm>people</LongForm>
                                </UOM>
                            </Count>
                        </Values>
                        <GetDataRequest
xlink:href="http://sis.agr.gc.ca/pls/meta/tjs_1x0_getdata?Service=TJS&amp;Version=1.0&amp;Re
quest=GetData&amp;FrameworkURI=http://sis.agr.gc.ca/cansis/nsdb/ecostrat/zone/v1&amp;Dat
asetURI=http://sis.agr.gc.ca/cansis/nsdb/ecostrat/zone/v1/population_1991&amp;Attributes=rurf_
91&amp;Language=en"/>
                    </Column>
                </Attributes>
            </Columnset>
            <Rowset>
                <Row>
                    <K>1</K>
                    <V>515</V>
                </Row>
                <Row>
                    <K>2</K>
                    <V>6157</V>
                </Row>
                <Row>
                    <K>3</K>
                    <V>5001</V>
                </Row>
                <Row>
                    <K>4</K>
                    <V>5591</V>
                </Row>
                <Row>
                    <K>5</K>
                    <V>10512</V>
                </Row>
                <Row>
                    <K>6</K>
                    <V>552160</V>
                </Row>
                <Row>
                    <K>7</K>
                    <V>626856</V>
                </Row>
                <Row>
                    <K>8</K>
                    <V>1040917</V>
                </Row>
                <Row>
                    <K>9</K>
                    <V>194603</V>
                </Row>
                <Row>
                    <K>10</K>
                    <V>352418</V>
                </Row>
                <Row>
                    <K>11</K>
                    <V>144</V>
                </Row>
                <Row>
                    <K>12</K>
                    <V>6711</V>
                </Row>
                <Row>
                    <K>13</K>
                    <V>160117</V>
                </Row>
                <Row>
                    <K>14</K>
                    <V>144466</V>
                </Row>
                <Row>
                    <K>15</K>
                    <V>4276</V>
                </Row>
            </Rowset>
        </Dataset>
      </Framework>
</GDAS>
```

Annex D (normative)

SOAP encoding for TJS

This Annex defines how a TJS service shall implement support for SOAP.

D.1 General

Support for SOAP encoding of TJS requests and responses is optional. Servers that do so shall support SOAP version 1.2 encoding, and shall publish WSDL documents as indicated in Annex E of this document.

D.2 Declaration of support for SOAP

A TJS server that supports the SOAP interface for an operation shall declare that support in the OperationsMetadata section of its ServiceMetadata document (i.e the GetCapabilities response) in accordance with OWS Common 1.1, as shown in the following example.

```
<ows:OperationsMetadata>
   <ows:Operation name="GetCapabilities">
      <ows:DCP>
         <ows:HTTP>
            <ows:Post xlink:href="http://foo.bar/foo.cgi?">
               <ows:Constraint name="PostEncoding">
                  <ows:AllowedValues>
                     <ows:Value>SOAP</ows:Value>
                  </ows:AllowedValues>
               </ows:Constraint>
            </ows:Post>
         </ows:HTTP>
      </ows:DCP>
   </ows:Operation>
<ows:OperationsMetadata>
```

If the service has a SOAP binding, there SHALL be a <WSDL> element. The value of the xlink:href attribute SHALL refer to a web accessible WSDL document.

D.3 SOAP encoding of TJS requests

SOAP encodings of TJS requests shall consist of a normal HTTP POST request that forms the content of the Body element of the SOAP document. For example, to request a TJS capabilities document using SOAP, a client could issue the following SOAP encoded GetCapabilities operation request:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
```

```
   <soap:Body>
      <GetCapabilities service="TJS"
            xmlns="http://www.opengis.net/ows/1.1">
         <AcceptVersions>
            <Version>1.0.0</Version>
         </AcceptVersions>
         <AcceptFormats>
            <OutputFormat>text/xml</OutputFormat>
         </AcceptFormats>
      </GetCapabilities>
   </soap:Body>
</soap:Envelope>
```

D.4 SOAP encoding of TJS responses

SOAP encodings of TJS responses shall consist of a normal TJS response that forms the content of the Body element of the SOAP document. The following fragment shows the a GetCapabilities response wrapped in a SOAP envelope:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <Capabilities version="1.0.0"
            xmlns="http://www.opengis.net/tjs/1.0"
            xmlns:ows="http://www.opengis.net/ows/1.1"
            xmlns:xlink="http://www.w3.org/1999/xlink"
            xmlns:gml="http://www.opengis.net/gml">
            ...
         <ows:OperationsMetadata>
            <ows:Operation name="GetCapabilities">
               <ows:DCP>
                  <ows:HTTP>
                     <ows:Post
               xlink:href="http://www.maps.cat/maps.cgi?">
                        <ows:Constraint name="PostEncoding">
                           <ows:AllowedValues>
                              <ows:Value>SOAP</ows:Value>
                           </ows:AllowedValues>
                        </ows:Constraint>
                     </ows:Post>
                  </ows:HTTP>
               </ows:DCP>
            </ows:Operation>
            ...
         <WSDL xlink:role=http://schemas.xmlsoap.org/wsdl/1.0
               xlink:show="none" xlink:type="simple"
               xlink:href="wmtsConcrete.wsdl"/>
      </Capabilities>
   </soap:Body>
</soap:Envelope>
```

D.5 Exceptions in SOAP encoding

If an error is detected while processing an operation request encoded in a SOAP envelope, the TJS server SHALL generate a SOAP 1.2 response message where the content of the Body element is a Fault element containing an ExceptionReport element (as defined in Clause 8.5 of [OGC 06-121r3]). This SHALL be done using the following XML fragment:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <soap:Fault>
         <soap:Code>
            <soap:Value>soap:Receiver</soap:Value>
         </soap:Code>
         <soap:Reason>
            <soap:Text>A server exception was encountered.</soap:Text>
         </soap:Reason>
         <soap:Detail>
            <ows:ExceptionReport
               xmlns:ows="http://www.opengis.net/ows/1.1">
               …
            </ows:ExceptionReport>
         </soap:Detail>
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

The Code element SHALL have the Value "soap:server" indicating that this is a server exception. The Reason element SHALL have the Text "A Server exception was encountered". This fixed string is used since the details of the exception SHALL be specified in the Detail element using an ows:ExceptionReport element.

Annex E
(informative)

WSDL description of the service

This Annex provides an abstract WSDL description for a generic TJS service and guidance on how to create a concrete WSDL description for a particular TJS server instance.

A WSDL document is typically used in combination with SOAP encoding but this annex describes WSDL documents that deal with KVP, XML POST, and SOAP encodings.

E.1 General

The Web Services Description Language (WSDL) is an XML language for describing the computational characteristics of a web service: interface signatures, protocol bindings and network endpoints. TJS servers that publish WSDL documentation of the service shall do so using WSDL version 1.1.

E.2 WSDL Publication

TJS servers that support WSDL shall populate the <WSDL> element within the service metadata document (i.e. the GetCapabilities response).

E.3 Abstract and concrete WSDL documents

WSDL documents are intended to be modularized through the use of import statements since they are structured in an abstract and concrete service instance part. These mechanisms permit the separation of service-specific elements from shared interface definitions. In practice, this separation means that the complete service will always be described by exactly one top-level WSDL. This top-level WSDL file may import a set of WSDL files for specific parts, for instance a WSDL for the abstract part and a WSDL describing only the concrete service instance part of the service.

E.4 Abstract TJS WSDL document

Abstract WSDL document have a modular design that reuses application schemas that are also used by SOAP messages. The abstract WSDL document describes types, messages and portTypes in a generic way and can be imported in the concrete WSDL description of any particular service instance.

E.5 Concrete TJS WSDL document

A concrete WSDL describes a particular server instance. It includes descriptions of five main parts: types, messages, portTypes, bindings and services. The concrete WSDL file imports the abstract WSDL file, identifies a <binding> element for each encoding the server supports and, within the <service> element identifies the port and encodings the server supports.

A HTTP GET <binding> element shall reference a GET portType from the abstract part and make use of the <http:binding verb="GET"> binding as described in the WSDL 1.1 specification. The operation element shall reference the corresponding operation from the abstract part and use <http:urlEncoded/> as input and <mime /> element as output. The <http:operation> element shall be used according to the WSDL 1.1 and WS-I Basic.

A SOAP <binding> element shall reference a SOAP portType from the abstract part and make use of the <soap:binding style="document"> binding as described in the WSDL 1.1 specification. The operation element shall reference the corresponding operation from the abstract part and use <soap:body use="literal"/> as input and output. The <soap:operation> element shall be used according to the WSDL 1.1 and WS-I Basic Profile 1.2 specifications. The soapAction attribute value shall follow the format:

http://www.opengis.net/<serviceType>/requests#<operationName>

<port> elements in the <service> element shall reference bindings from the binding part. In a GET encoding an <http:address location=""> element shall reference the URL of the service (that has to be the same of the <ows:Get xlink:href=""> element attribute in the service metadata document). In a SOAP encoding an <soap:address location=""> element shall reference the URL of the service (that has to be the same value of the <ows:Get xlink:href=""> element attribute in the service metadata document).

E.6 WSDL document examples

Example WSDL documents are bundled in a zip file with the present document. After OGC acceptance of a Version 1.0 of this standard, these examples will also be posted online at the URL http://schemas.opengeospatial.net/TJS/1.0.0/examples/.