Open Geospatial Consortium

Approval Date: 2012-11-23

Publication Date: 2012-12-05

OGC name of this OGC® project document: :http://www.opengis.net/doc/bp/wami/1.0

Reference number of this OGC® project document: OGC 12-032r2

Version: 1.0

Category: OGC® Best Practice

Editors: Rahul Thakkar, Michael Maraist

WAMI Services:

Dissemination Services for Wide Area Motion Imagery

Document type:	OGC® Best Practice
Document subtype:	if applicable
Document stage:	Approved for public release
Document language:	English

License Agreement

Contents

Tables

Figures

i. Abstract

This OGC Best Practice (BP) describes web interface specifications for the access and dissemination of Wide Area Motion Imagery (WAMI) products and metadata. This BP also describes a framework and interface specifications common to all WAMI services.

ii. Keywords

Ogcdoc,ogc, wami, imagery

iii. Preface

This document describes a set of web services for the dissemination of Wide Area Motion Imagery (WAMI) products. It describes a performance centric, scalable grammar and schema that enables globally federated dissemination of WAMI products for high performance, high speed consumption.

The reason for developing the specification was a WAMI community requirement to deliver high performance web services and disseminate WAMI products. The focal point quickly became a grammar that suites performance. While existing web services can be combined or modified to deliver some of the functionality of the services described in this document, by design, they cannot deliver the desired performance.

These services are consumer centric. They deliver only as much as was requested and no more. The services deliver little to a lot of data based on the requests. The services deliver incremental data. Multiple modalities in the quantity of data delivered permits optimal server development/deployment capabilities across networks of various latency and bandwidth behaviours.

Clients can choose from a menu of capabilities and develop a UX to best suite consumer needs. Servers are judged on their ability to deliver raw performance and their ability to scale across an implicitly federated global enterprise. The services are modelled for implementation on inexpensive commodity compute and storage infrastructure permitting cloud vendors to run these at the platform layer.

iv. Submitting organizations

The following organizations submitted this Implementation Specification to the Open Geospatial Consortium Inc. for consideration as a Best Practice Document:

a) PIXIA Corp.

v. Submission contact points

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

CONTACT	COMPANY	WEBSITE
David Barton	PIXIA Corp.	http://www.pixia.com
Rahul Thakkar	PIXIA Corp.	http://www.pixia.com

vi. Revision history

Date	Release	Authors	Paragraph modified	Description
3/16/2012	Draft	Rahul Thakkar, Michael Maraist, Christopher Bason, Scott Pakula, Brian Angelica, John Merrifield	All	Original specifications
3/16/2012	Draft	David Barton	Footer, Cover	Added OGC document number
5/08/2012	Draft	Rahul Thakkar, Michael Maraist, Christopher Bason, John Merrifield, Ted Yezek	RS addendum, editor change	Added Raw Service (RS) as Section 5 and Addendum 1. CS, IS, VS, etc. unchanged. Changed editor.
8/30/2012	Ready for OGC	Rahul Thakkar, Michael Maraist	Title, Section 2/CS, Appendix	Added 3 parameters + schema to CS, RS removed for further research, edited for a Best Practice document

vii. Changes to the OGC® Abstract Specification

The OGC® Abstract Specification does not require changes to accommodate this OGC®best practice document.

Foreword

PIXIA Corp authored various specifications for the dissemination of Wide Area Motion Imagery (WAMI) between 2007 and 2009. PIXIA Corp has advocated open standards to promote interoperability and healthy technology growth. Since these specifications were modelled to be open, and due to positive response from the US Government and the industry in favour of these specifications, we are approaching the OGC to consider these specifications for standardization.

This document is a direct consequence of that work and is an HTTP-based specification written for a high performance, and scalable requirement from the WAMI customer base. The success of a commercial or open product based on an interoperable standard depends heavily on the specification of that standard.

Upon review of this document, it will be obvious of the many applications that this specification can spawn and support a very clean, open, and performance-centric development framework with a strong consumer focus.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights. However, to date, no such rights have been claimed or identified.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

Introduction

This document is comprised of multiple sections.

The first section describes the general model of web interface specifications for disseminating Wide Area Motion Imagery (WAMI) data and metadata. This section provides interface specifications common to all WAMI services. The interface specifications are designed to be modular and extensible. They use HTTP 1.1 as its base reference standard. The specifications are divided into a set of services. Each service performs a specialized task, is stateless and RESTful. The services are read-only (i.e. server is not permitted to change the data).They draw heavily from the Open Geospatial Consortium, Inc.® or OGC® to define the grammar. They reference OGC's Web Services Common Standard Version 2.0.0 document number 06-121r9.

The second section describes HTTP interface specifications for a Collection Service or CS. A *collection* is defined as a set of WAMI data such that each element or frame of WAMI data is temporally sequential from the previous element. An *element* or *frame* of WAMI data can be a single image or a group of images that constitute a single logical picture of an instant in time. How these elements are stored on the server side is implementation and vendor specific and not associated with the specifications themselves. The CS allows a client to incrementally discover what collections of WAMI data are being served.

The third section describes HTTP interface specifications for an Image Service or IS. An Image Service serves a client a requested area of interest (AOI) from a collection of WAMI data and delivers it as an image file of known and supported format. The IS also provides metadata about the same AOI. Metadata is reported as formatted text. The web service is image format agnostic.

The fourth section describes HTTP interface specifications for a Video Service or VS. A Video Service serves a client a requested area of interest (AOI) from a collection of WAMI data and delivers it as a video stream or file of known and supported format. The VS also provides metadata about the same AOI. Metadata is reported as part of the video stream. The video stream is delivered in a well-known format. At least one video stream must be supported. It is recommended that at least one video stream format be *MISB* compliant to follow the *MISP*. The web service is image format agnostic.

1 Scope

This document describes web interface specifications for the dissemination of WAMI data and metadata. It also provides the framework and interface specifications common to all WAMI services. It includes:

1. Operation request and response contents, most partial
2. Parameters and data structures included in operation requests and responses
3. XML and KVP encoding of operation requests and responses

2 Conformance

Conformance with this specification shall be checked using all the relevant abstract tests specified in an Abstract Test Suite. While highly desirable, a test suite is not part of this document at this time. It will be developed as part of transitioning this specification from a best practice document to a potential standard.

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of OGC 12-032r2. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of OGC 12-032r2are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

- [RFC 2119] “Key words for use in RFCs to Indicate Requirement Levels” http://www.ietf.org/rfc/rfc2045.txt
- [RFC 2045] “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”
- [RFC 2141] “URN Syntax” http://www.ietf.org/rfc/rfc2141.txt
- [RFC 2396] “Uniform Resource Identifiers (URI)” http://www.ietf.org/rfc/rfc2396.txt
- [RFC 2616] “Hypertext Transfer Protocol – HTTP/1.1” http://www.ietf.org/rfc/rfc2616.txt
- [RFC 4646] “Tags for Identifying Languages” http://www.ietf.org/rfc/rfc4646.txt
- [RFC 2387] “MIME Multipart/Related Content-type” http://www.ietf.org/rfc/rfc2387.txt
- [RFC 2557] “MIME Encapsulation of Aggregate Documents” http://www.ietf.org/rfc/rfc2557.txt

- [RFC 2111] “Content-ID and Message-ID Uniform Resource Locators” http://www.ietf.org/rfc/rfc2111.txt
- [RFC 2518] “HTTP Extensions for Distributed Authoring – WEBDAV” http://www.ietf.org/rfc/rfc2518.txt
- [RFC 3986] “Uniform Resource Identifier (URI): Generic Syntax”
- [ISO 8601:2000(E)] “Data elements and interchange formats – Information interchange – Representation of dates and times” http://www.iso.org/iso/support/faqs/faqs_widely_used_standards/widely_used_standards_other/date_and_time_format.htm
- [ISO 19115:2003] “Geographic information – Metadata” http://www.iso.org/iso/catalogue_detail.htm?csnumber=26020
- OGC resources:
  - OGC website: http://www.opengeospatial.org.
  - OGC standards: http://www.opengeospatial.org/standards.
  - [OGC WSCS] OGC Web Services Common Standard. Document 06-121r9: http://www.opengeospatial.org/standards/common.
- [WAMI XSD] This is the WAMI XML Schema Definition file (XSD), currently wami_1_0_1.xsd. It is a detailed part of the WAMI specification
- [WAMI OVERVIEW] This document

4 Terms and definitions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC 2119].

AOI	Area of interest
WAMI	Wide area motion imagery
Collection	A set of elements or frames of data
Element or Frame	A picture of a reasonably wide geographic area on the ground, with raster information and associated metadata.

5 Section 1: General Information and Common Specifications

This section provides general information about WAMI the model of these specifications. It also describes the grammar that is common to all specifications herein.

6 WAMI

WAMI or Wide-Area Motion Imagery is a relatively new field. It refers to data collected by sensors operating in various bands of the electromagnetic spectrum. These sensors are generally of the order of several tens of megapixels to several gigapixels in terms of their cumulative ability to capture a single snapshot or a large area of interest at an instant in time. These sensors capture such data at rates that are generally greater than 1-Hertz.

There are several names by which WAMI is referred to. For example, WALF or Wide-Area Large Format, WAPS or Wide-Area Persistent Surveillance, WAAS or Wide-Area Airborne Surveillance, LVSD or Large Volume Streaming Data, and WAS or Wide-Area Surveillance. There are other acronyms for this type of data that are not included as part of this document. Such acronyms may be used interchangeably with WAMI.

The sensor collecting this data may be looking at or “staring” at one location on the ground or it may be changing its look-at point based on mission requirements.

National Geospatial-Intelligence Agency (NGA) has the Geospatial Intelligence Standards Working Group (GWG). It is working towards defining standards for WAMI data. Within Motion Imagery Standards Board or MISB is part of the GWG and is involved in defining some of those standards.

Products based on the specifications outlined in this document are in successful operational use across a global enterprise.

7 Performance

The grammar must permit a high performance scalable implementation.

The implementation of a server based on these web services as two central elements of focus: Performance and Scale. The service implementation must perform to a consumer or customer imposed metric and not a client or server imposed limit. WAMI data is not only delivered to limited viewers that play at 2, 5 or 10 frames per second. WAMI data is delivered to automatic image processing algorithms that may reside locally or globally. They cannot starve for data, otherwise the cost of having such algorithms scale increases rapidly, at times exponentially.

The performance of these services depends on the available infrastructure and implementation quality. Considering the nature of the data being served, good performance is critical. On the server-side, performance may depend on factors such as the quality of implementation of the services, the compute nodes, the network and the storage sub-systems.

On the client-side, performance may depend on factors such as the network connection (latency and speed) between the client and the server, the power of the client compute- nodes and the quality and platform of implementation of the client software is also a metric.

While these variables are outside the scope of the specification itself, the specification can be developed to assist in efficiently managing the speed and latency of the connection between the client and the server.

For example, if a network is really high speed, but has high latency, sending a large number of HTTP requests would be detrimental to performance. Instead, the client is better off sending fewer requests but receiving as much data as possible in those few requests.

As another example, low latency, medium or low speed networks need to have the ability to send small chucks of data back and forth to minimize quantity of data sent across the network. Such a setup is useful for thin clients without a lot of compute power behind them and for clients that do not do as intensive work as a thick client.

WAMI web services are designed to tackle high performance requirements under all forms of networks, enabling both thin and thick client applications to keep up with the consumer demand of fast analytics of WAMI raster data, search and discovery over WAMI metadata, faster than real-time visualization and processing of WAMI raster data, either as a flipbook or full motion video.

In addition, the specification provides basic non-linear editing capabilities to enable the generation of products based on analytics.

8 Services categories

WAMI services fall under two broad categories:

1. Ingest services
2. Dissemination services

8.1 Ingest Services

These services acquire and organize data from a WAMI collection source. They may prepare or process the data to make it ready for dissemination. Ingest services are generally vendor specific and outside the scope of this document. This document only deals with dissemination services.

8.2 Dissemination Services

These services present WAMI data in a variety of formats such that an exploitation application can utilize the data for processing and/or visualization.

The data dissemination services were divided into three types:

1. **Core services** manage and access WAMI data directly; they may consult each other
2. **Alert services** focus on providing the status of WAMI data and services availability
3. **Derived services** enrich other services to provide an enhanced perspective of the data

Table 1: WAMI dissemination services

Name	About
Core services	
Collection Service (CS)	An implicitly federated service informs a client of all WAMI data being served with links to content deliver services. It presents a hierarchical view of WAMI data with incremental dissemination capability.
Image Service (IS)	Delivers derived WAMI content from bounded areas of interest (AOI) across time over multiple collects as a flipbook of one or more maps and metadata.
Video Service (VS)	It is similar to IS. It delivers WAMI content as video.
Raw Service (RS)	Delivers original raw WAMI data as it was acquired, unblemished.
Alert services	
GeoRSS	Implements a pull-based alert service. Uses the GeoRSS schema over RSS.
XMPP	Implements a push-based alert service. Uses the GeoRSS schema over XMPP.
Derived services	
Video Control and Streaming Service (VCSS)	It is similar to VS. It provides Digital Video Recorder (DVR) controls over HTTP delivering video windows over different AOIs via HTTP, TCP, or UDP. Consumers can play forward, backward, pause across all data, and go-to-live. They can pan around; zoom in and out. (In development).
Query Service (QS)	This service provides a search capability atop multiple CS. Use of existing negotiated NoSQL industry standard interfaces is recommended.
Virtual File System (VFS)	Provides a POSIX file system view atop WAMI frames as a set of folders and files using HTTP 1.1 WebDAV. Circumvents vendor lock-in and supports file-based apps to function, giving them migration option and opportunity to survive in an increasingly SOA environment.

8.2.1 Collection Service

The collection service or CS allows a client to discover what collections of WAMI data are being served by a server implementation. A collection is defined in the **Collection Service** specification, version 1.0.2.

8.2.2 Image Service

An image service or IS allows a client to request areas of interest from individual elements from a collection and receive them as a sequence of still images in a known format. It can also provide metadata associated with the prior request.

8.2.3 Video Service

A video service or VS allows a client to request areas of interest from individual elements from a collection and receive them as video code streams in a known format. It can also provide metadata associated with the prior request.

8.2.4 Raw Service

A raw service or RS provides a simple spatiotemporal interface to deliver the original content, unmodified. The specification for this service is not part of this document.

8.2.5 Query Service

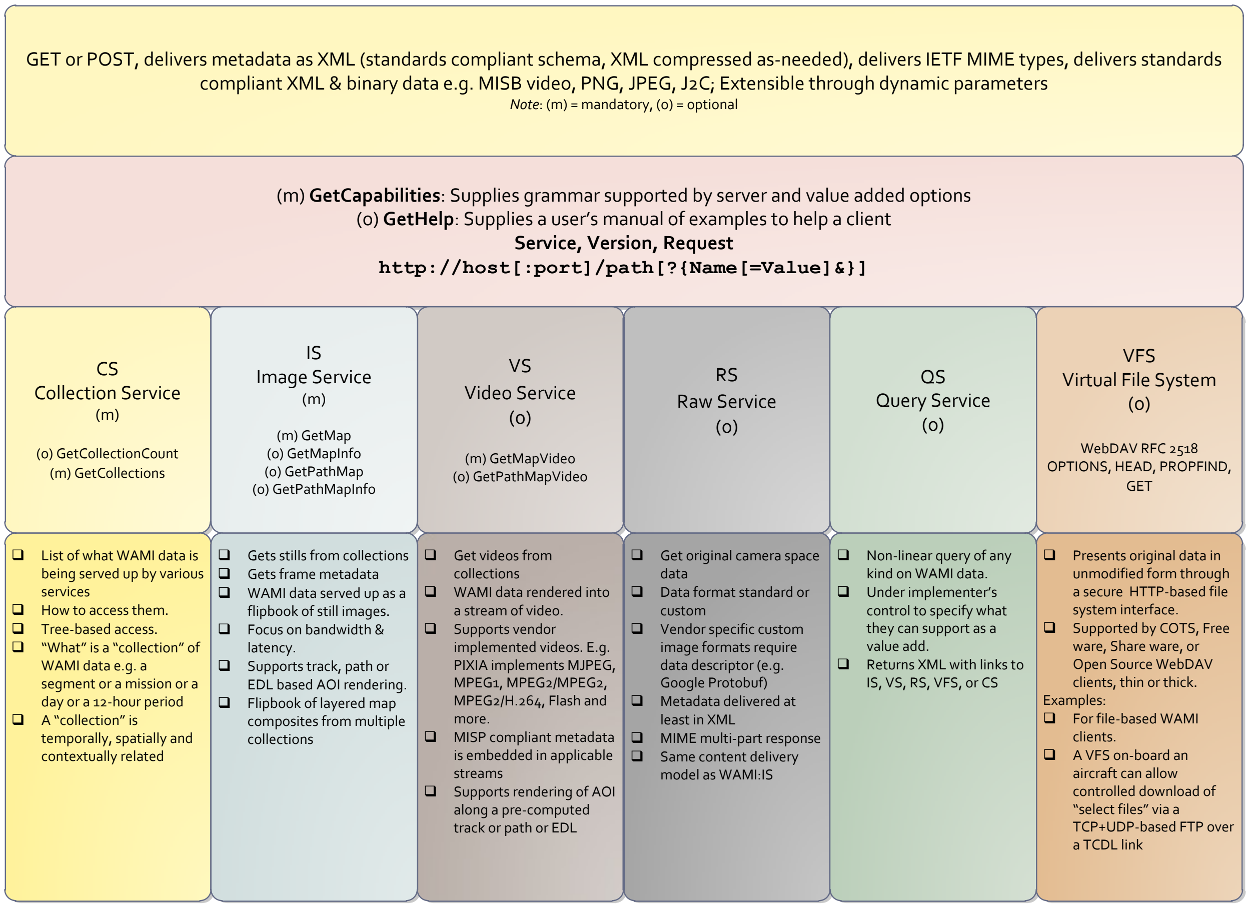
A query is a search operation performed on the WAMI data being served up. The query service or QS allows a client to know what kind of queries can be performed on the data being served up and perform those queries to filter through data being served up to find relevant information. The specification for this service is not part of this document.

8.2.6 Video Control and Streaming Service

A video control and streaming service or VCSS allows a client to request areas of interest from individual elements from a collection and receive them as video code streams in a known format. The API allows for dynamic pan/zoom, pause, go-live, etc. The specification for this service is not part of this document.

8.2.7 Virtual File System

All prior services provide data in some known format. This service provides the data as it was originally presented to the server. The server presents an HTTP based virtual file system or VFS on top of the data, making it possible for the client to transfer the original data as is. We are recommending the use of an extension to HTTP 1.1, namely WebDAV. [RFC 2518] describes WebDAV. For achieving our goals, we require implementing `OPTIONS`, `PROPFIND`, `HEAD`, and `GET`. These are read-only operations where the server will not be asked to change anything.



GET or POST, delivers metadata as XML (standards compliant schema, XML compressed as-needed), delivers IETF MIME types, delivers standards compliant XML & binary data e.g. MISB video, PNG, JPEG, J2C; Extensible through dynamic parameters

Note: (m) = mandatory, (o) = optional

(m) **GetCapabilities**: Supplies grammar supported by server and value added options

(o) **GetHelp**: Supplies a user's manual of examples to help a client

Service, Version, Request

```
http://host[:port]/path[?{Name[=Value]&}]
```

CS Collection Service (m)	IS Image Service (m)	VS Video Service (o)	RS Raw Service (o)	QS Query Service (o)	VFS Virtual File System (o)
(o) GetCollectionCount (m) GetCollections	(m) GetMap (o) GetMapInfo (o) GetPathMap (o) GetPathMapInfo	(m) GetMapVideo (o) GetPathMapVideo			WebDAV RFC 2518 OPTIONS, HEAD, PROPFIND, GET
❑ List of what WAMI data is being served up by various services ❑ How to access them. ❑ Tree-based access. ❑ "What" is a "collection" of WAMI data e.g. a segment or a mission or a day or a 12-hour period ❑ A "collection" is temporally, spatially and contextually related	❑ Gets stills from collections ❑ Gets frame metadata ❑ WAMI data served up as a flipbook of still images. ❑ Focus on bandwidth & latency. ❑ Supports track, path or EDL based AOI rendering. ❑ Flipbook of layered map composites from multiple collections	❑ Get videos from collections ❑ WAMI data rendered into a stream of video. ❑ Supports vendor implemented videos. E.g. PIXIA implements MJPEG, MPEG1, MPEG2/MPEG2, MPEG2/H.264, Flash and more. ❑ MISP compliant metadata is embedded in applicable streams ❑ Supports rendering of AOI along a pre-computed track or path or EDL	❑ Get original camera space data ❑ Data format standard or custom ❑ Vendor specific custom image formats require data descriptor (e.g. Google Protobuf) ❑ Metadata delivered at least in XML ❑ MIME multi-part response ❑ Same content delivery model as WAMI:IS	❑ Non-linear query of any kind on WAMI data. ❑ Under implementer's control to specify what they can support as a value add. ❑ Returns XML with links to IS, VS, RS, VFS, or CS	❑ Presents original data in unmodified form through a secure HTTP-based file system interface. ❑ Supported by COTS, Free ware, Share ware, or Open Source WebDAV clients, thin or thick. Examples: ❑ For file-based WAMI clients. ❑ A VFS on-board an aircraft can allow controlled download of "select files" via a TCP+UDP-based FTP over a TCDL link

Figure 1: Overview of WAMI dissemination services

9 Relationship between services

All WAMI services serve up *collections* of WAMI data organized as an inverted tree structure. In computer science a tree is a widely-used data structure that emulates a hierarchical tree structure with a set of linked nodes. For the purpose of this specification, it is a data structure that is an *ordered directed tree*, more specifically *arborescence*: a connected acyclic graph where each node has zero or more *children* nodes and at most one *parent* node. Furthermore, the children of each node have a specific order. A *node* is a structure which may contain a value, a condition or represent a separate data structure. For more, see the **Collection Service** specification, version 1.0.2.

Each WAMI service is stand-alone. Also, each WAMI service is implicitly related to other WAMI services.

1. A Collection Service serves up a collection tree with links to Image, Video, Raw and Virtual File System services along leaf nodes as well as other Collection Services along inner nodes.
2. An Image Service presents one or more frames as a sequence of map image files. It may use a VFS for indirect access to remotely located source.

3. A Video Service presents one or more frames as a video of a sequence of maps. It may use Image service or VFS requests and distribute the retrieval of its input.
4. A Raw Service provides original camera space data and metadata, unmodified. RS and IS can operate on the same dataset.
5. A Virtual File System presents the frames in their raw original form as they were incorporated into the collection.
6. A Query service traverses collection trees and executes any supported request for information from a client and presents data, metadata, and links to zero or more Collection, Image, Video, Raw and Virtual File System services.

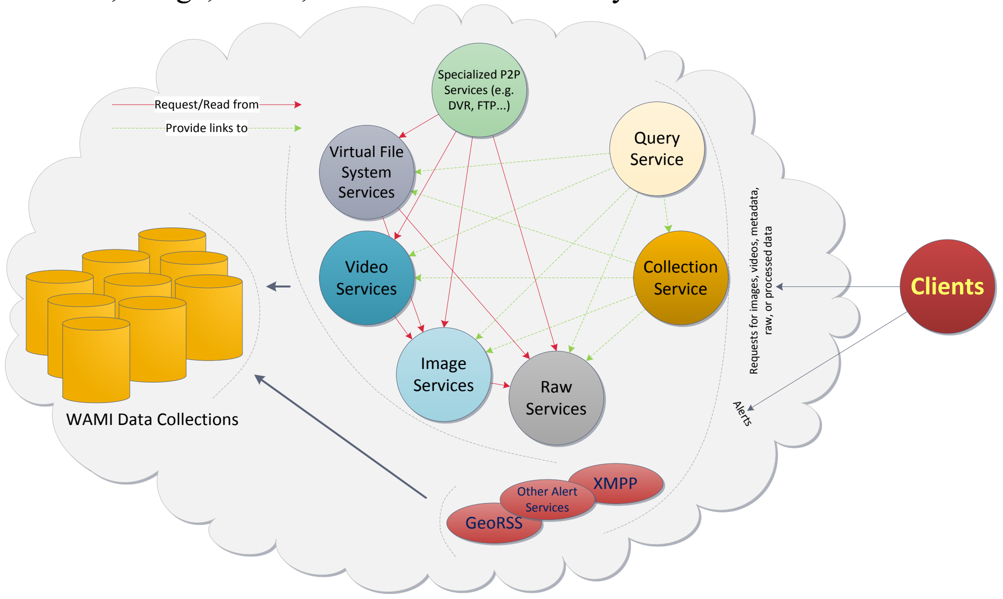


Figure 2: Relationship between various WAMI services

9.1 Higher Level Services

A higher level service would make use of any of these services to provide a functionally enhanced experience. A Video Control and Streaming Service or VCSS is an example of a "higher level service". It allows a client to establish a persistent connection with a server, start a video stream of an AOI from a specific WAMI collection, and control WAMI data that is placed in that stream. VCSS provides the client with DVR-like controls to pan around, zoom in and out, rewind, fast forward, shuttle, or change the playback speed. VCSS outputs a video stream that reflects the choices made in the controls. This service can be used to stream live data, as it is being acquired or go back in time at any point.

It can use an Image Service to get an AOI from a frame from a WAMI collection that is in process of being collected. It can then place this AOI in a "video frame buffer" that is constantly being transmitted to a client. It would use its controls to manage what the AOI is, and how frequently it is accessed.

Higher level services are vendor specific and may choose to follow the same grammar pattern that these services follow.

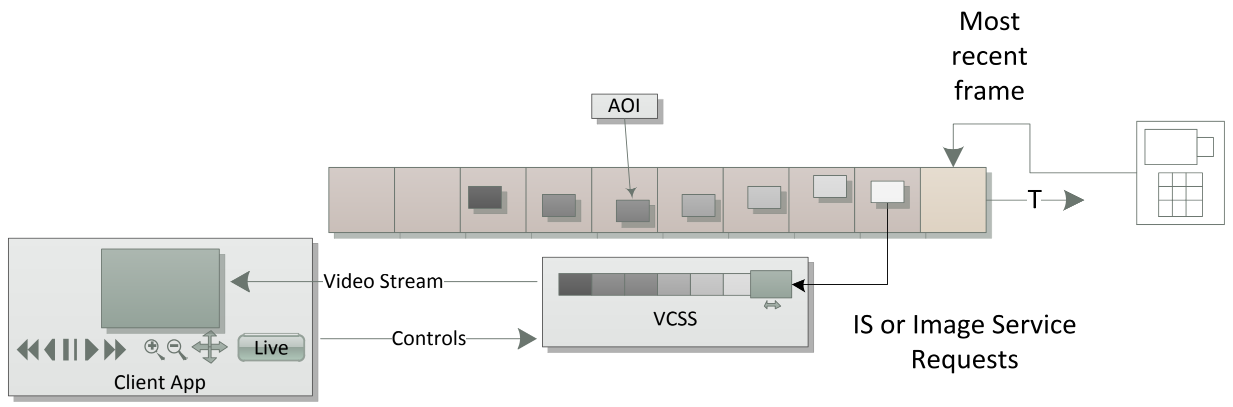


Figure 3: Video control and streaming service

10 Version

The current version number of these services is 1.0.2. This version number format is in accordance with the [OGC WSCS] section 7.3.1.

11 General Interface

The client makes an HTTP requests to a server based on [RFC 2616]. The Online Resource for each operation supported by a server is an HTTP Uniform Resource Locator (URL). See section 3.2.2 in [RFC 2616] for more on HTTP URL.

11.1 Request

HTTP supports two request methods: GET and POST. One or both of these methods may be offered by a server for each operation. The use of Online Resource URL differs in each case. GET uses Keyword Value Pair (KVP) encoded operation requests. POST can use either KVP or XML encoded operation requests.

11.1.1 HTTP request

The request is defined in accordance with [RFC 2396]. It is summarized by the following table, wherein [] and {} denotes optional components.

Table 2: Structure of an operation request using HTTP GET and POST

Component	Structure	Example	Description
URL - GET	`http://host[:port]/path[?querystring]`	`http://example.com/IS?Request=GetCapabilities`	Contains all information necessary to complete a request
URL - POST	`http://host[:port]/path`	`http://example.com/IS`	Contains ONLY routable information. It is possible to add a query-string or anchor, but they should not be considered part of the request parameters.

11.1.2 KVP Encoding

11.1.2.1 HTTP KVP query-string

The query string is defined in accordance with [RFC 2396]. It can be used either with an HTTP-GET as the query-string component of the URL, or it can be used as the POST content-body in the HTTP-POST request. It is summarized by the following table, wherein [] and {} denotes optional components. And {} denotes zero r more occurrences.

The following two tables provide the structure of a general HTTP GET request as well as the list of reserved characters in operation request strings. See [OGC WSCS] Tables 39 and 40.

Table 3: Structure of an operation request using HTTP GET

Component	Structure	Example	Description
Query string	`name=value{&name=value}`	`Service=IS&Request=GetCapabilities&Version=1.0.2`	The Key and Value pairs in accordance with [RFC2396] are properly escaped – namely no '=', '&', ';', '/', '#', '?', '@', '+', '$' symbols may be used without hex-encoding.
Query name	`part{.part}`	`Options.video.compression`	WAMI services SHALL support case insensitivity for names. Hierarchical structures may be represented by using '.' between related components.

Table 4: Reserved characters in operation request strings.

Character	Usage
?	Separator indicating start of a query string
&	Separator between parameters of a query string
;	Alternate Separator between parameters of a query string
=	Separator between the name and value of a parameter
,	Separator between individual values in list-oriented parameters or parameters that can have more than one value
%	Character used to escape path and query-string
#	Defines the beginning of an anchor tag and thus the beginning of the path and or query-string.
+	Shorthand representation for a space character in the query string
/	Path separator character, which technically unambiguous due to the presence of the '?' character, it is disallowed.
@	Separator for user-name in URL
:	Separator used for schema

11.1.2.2 KVP Encoding

The manner in which a GET request provides name/value pairs utilizes KVP encoding as shown in [OGC WSCS] Section 11.5. An abridged version of this section is provided below.

In a name/value pair, the *name* component is not case sensitive. For example, the "REQUEST" parameter could be REQUEST, Request, request or ReQuEsT.Value lists are separated by a comma. White space is not permitted as a separator. An empty list shall be represented by an empty string ("").

Boolean parameters shall have values that can be either TRUE or FALSE. They will be in upper case.Each parameter in a request should be independent of all other parameters in that request or any other request.Nested KVP encoding is permitted and follows the syntax that is specified below using EBNF notation:

```
KVPS := [ "?" | "&" ] KVPList
KVPList := KVP *( "&" KVP )
KVP := Key "=" Value
Key := UrlEncodedKey
Value := UrlEncodedKVPList | UrlEncodedValue
UrlEncodedKVPList := UrlEncodedKVP *( "%26" UrlEncodedKVP )
UrlEncodedKVP := UrlEncodedKVPList |
UrlEncodedKey "%3D" UrlEncodedValue
UrlEncodedKey := <Any URL Encoded string>
UrlEncodedValue := <Any URL Encoded string>
```

11.1.2.3 KVP Content Body

If the KVP is deterministically small, and under 2KB in length, it is permissible to utilize HTTP-GET and embed the KVP as the query-string portion of the URL. Note that there is a risk in doing so – Not all web-servers or web browsers support larger than 2,000 byte HTTP HEADERS (including all associated metadata including cookies – which are not always controllable – an example is the AJP binary proxy protocol). For most WAMI services, this is unlikely. However, Some WAMI calls facilitate list multi-valued KVP items, and thus run the risk of exceeding this 'safe' limit. When in doubt, use HTTP-POST described below. HTTP-GET is provided merely as a convenience as it is readily reproducible with commonly available web-browsers.

The WAMI spec promotes HTTP-POST with the KVP query-string as the content-body as a safe alternative to HTTP-GET style query-strings. This allows unbounded multi-valued input parameter sequences with little risk to either the client or server (as both ends can stream contents).When submitting an HTTP-POST with query-string KVP encoding, the Content-Type MUST be "application/x-www-form-urlencoded".

Note that HTTP-POST commonly supports Content-TYPE "multipart/form-encoded" but this is more verbose, cumbersome and less efficient than KVP encoding as encoded by "application/x-www-form-urlencoded". It is generally only used for file-uploads from web-browsers, for which the WAMI API does not require.

11.1.2.3.1 Example HTTP-GET

```
GET /IS?Service=IS&Request=GetCapabilities HTTP/1.1
Host: example.com
```

11.1.2.4 Example HTTP-POST

```
POST /IS HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 34

Service=IS&Request=GetCapabilities
```

11.1.3 XML Encoding

As an alternative to KVP encoding, XML encoding is a viable alternative. This requires that the HTTP method be of type 'POST' and the Content-Type be of type 'application/xml'. The WAMI schema defines the XML request documents.

Of notable difference between the KVP and XML encoding is the need for XML to use space-separated values instead of comma separated values. Also, in the WAMI, many KVP items are broken into hierarchical structures, as they are more easily validated.

The use of XML request messages facilitates alternate RPC mechanisms such as SOAP, and thus a fully validated end-to-end web-service solution. However, the simplicity of the request parameters and the proliferation of non-SOAP RESTful to clients has lead this specification to discourage SOAP implementations.

A client MUST NOT send a POST XML encoded document unless the GetCapabilities Operation lists an explicit PostContentTypeMetaData permission which contains a <wami:XML /> element.

An example XML fragment:

```
<Operation name="GetCollectionCount">
<Metadata>
<wami:PostContentTypeMetaData>
<wami:KVP/>
<wami:XML schemaElement="CS_GetCollectionCountRequest"/>
</wami:PostContentTypeMetaData>
</Metadata>
</Operation>
```

This example shows an XML ONLY Collection Service which is exclusively expecting an HTTP POST, a Content-Type of "application/xml" and the CS_GetCollectionCountRequest XML document.

Note, however, that here, the URL INCLUDES a query-string. It is legal to contain both a query-string and POST data in an HTTP request, as generally they are used by separate layers of the application. Here the query-string is part of the routing information, which will expect the specified XML document. Clients that wish to leverage POST data should be aware of such URL query-string options.

The XML encoding, however, is more geared towards SOAP oriented implementations. Meaning a given SOAP service provider will expose a WSDL which defines its own way to specify service ports, but the Document Request and Document Response bodies wrapped in the SOAP envelop are expected to implement the Request/Response XML schema's outlined in [WAMI XSD].

Each WAMI Service defines both a Request and Response XML XSD element name. But only the Response is required to be implemented by all WAMI implementations. XML-Request encoding is completely optional and must be specified in GetCapabilities.

11.2 Response

See [OGC WSCS] Section 11.7

11.2.1 HTTP Status Codes

A successful response to a valid request must include 200 HTTP status code. An example set of HTTP status codes for exception codes that are generated for the service are as shown in the table below. For a more comprehensive list, see [RFC 2616] section 10.

Table 5: Services exception code values and corresponding HTTP status codes

Services Exception Code value	HTTP Status Code	HTTP Status Message
OperationNotSupported	501	Not implemented
MissingParameterValue	400	Bad request
InvalidParameterValue	400	Bad request
VersionNegotiationFailed	400	Bad request
InvalidUpdateSequence	400	Bad request
OptionNotSupported	501	Not implemented
NoApplicableCode	3xx,4xx,5xx	Internal server error

11.2.2 Example Response with HTTP Status Code

```
    HTTP/1.1 400 Bad Request
    Date: Thu, 4 Sep 2008 06:20:15 GMT
    Content-Type: application/xml
    Content-Language: en
    Content-Length: 415

<?xml version="1.0" encoding="UTF-8"?>
<ExceptionReport
   xmlns="http://www.opengis.net/ows/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/ows/2.0
http://schemas.opengis.net/ows/2.0/owsAll.xsd"
   version="1.0.2"
   xml:lang="en-US"
>
<Exception exceptionCode="OperationNotSupported" locator="GetExtendedCapabilities">
<ExceptionText>operation name GetExtendedCapabilities not recognized</ExceptionText>
</Exception>
</ExceptionReport>
```

11.2.3 HTTP Response Body

The HTTP Response body shall be accompanied by the appropriate MIME type. See [RFC 2045] for a list of MIME types.

11.2.3.1 MIME Types

All response objects shall be accompanied by the appropriate MIME type. See [RFC 2045] for MIME types. IANA maintains MIME types. Basic structure of MIME types is a string of the form “type/subtype”. MIME allows additional parameters in a string of the form “type/subtype; param1=value1; param2=value2”. A server may include parameterized MIME types in its list of supported output formats. In addition to any parameterized variants, the server should offer the basic un-parameterized version of the format.

11.2.4 Response XML Compression

When returning a large XML, some form of compression should be supported. Client-server communication will be considerably faster with the compression. The standard HTTP way of negotiating compression using **gzip** is fully defined in [RFC 2616], sections 3.5, 14.3 and 14.11. If the client can support compression, it may include the MIME header “`Accept-Encodings: gzip`” in its operation request. If the server sees this MIME header in the request and supports this compression, it may compress its operation response using **gzip**. The fact that the response is compressed is specified by the server by including the MIME header “`Content-Encoding: gzip`”. If the client sees this MIME header in the operation response, it shall decompress the response before parsing it. Also see [OGC WSCS] Section 11.7.3.

12 Exception Handling

Upon receiving an invalid operation request, each service shall respond to the client using an Exception Report message to describe to the client application and/or its human user the reason(s) that the request is invalid. Whenever a server detects an exception condition while responding to a valid operation request, and cannot produce a normal response to that operation, the server shall also respond to the client using an Exception Report. Refer to section 8 of [OGC WSCS].

All responses to exceptions shall be reported **at least** in MIME type **`application/xml`**. If the service interface specification requires the server to report the exception in any other format, it shall do so explicitly by setting the **Exceptions** parameter to its appropriate value in the request. See the section on Request Grammar below.

12.1 Exception Report

The element **ExceptionReport** begins each Exception Report. An **ExceptionReport** shall contain two attributes, **version** and **xml:lang**. The **version** attribute is required and shall contain a version number in the x.y.z format as specified in the [OGC WSCS]. The **xml:lang** attribute is optional and shall identify the language used by all included exception text values. The language identifier is based on [RFC 4646].

An **ExceptionReport** shall contain one or more **Exception** elements. Each **Exception** element signals the detection of an independent error.

An **Exception** element shall contain two attributes, **exceptionCode** and **locator** and one element **ExceptionText**. The **exceptionCode** attribute is required and contains a code representing the exception. Possible values of **exceptionCode** codes are specified in the table below. The **locator** attribute is optional and indicates to the client where an exception was encountered in servicing the client's operation request. Possible **locator** values are specified in the table below.

An **ExceptionText** element is optional but highly recommended. The value of this element is an ordered sequence of test strings that provide a description of the error.

The XML schema document for an exception report is available at the OGC website in the document named **owsExceptionReport.xsd**.

When a server responds with an Exception Report the HTTP reply header shall indicate an error. This is accomplished by setting the appropriate status code as shown in **HTTP Status Codes**.

Table 6: Element and attributes for "ExceptionReport"

Name	Description	Type, data type and value	Multiplicity and use
version	The version of the specification to which the ExceptionReport conforms	Attribute. Character String, not empty. Format is x.y.z where x, y, z are non-negative and [0..99]. Value is set by each implementation specification and schema version	One (mandatory)
xml:lang	Language used by all included exception text values	Attribute. Character String, not empty. Based on [RFC 4646]	Zero or one (optional). Should be included.
Exception	Element that indicates an independent error	Element. Complex type, not empty. See table below on Exception	One or more (Mandatory).

Table 7: Elements and attributes for "Exception"

Name	Description	Type, data type and value	Multiplicity and use
exceptionCode	Code representing the type of exception	Attribute. Character String, not empty. Allowed values are set by the implementation specification	One (mandatory)
Locator	Indicator of location in the client's operation request where this exception was encountered	Attribute. Character String, not empty. Contents are defined by the implementation specification	Zero or one (optional). Should be included.
ExceptionText	Text describing specific exception represented by the exceptionCode	Element. Complex type, not empty. Value is exception description as defined by the server implementation.	Zero or more (optional). Should be included. Omit only when no relevant information is available.

Table 8: Standard exception codes and meanings for all services

exceptionCode Value	Meaning	The value of **locator**
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. (We recommend placing the value that was invalid and if possible, a solution to the error in **ExceptionText**)	Name of parameter with invalid value.
VersionNegotiationFailed	List of versions in **AcceptVersions** parameter value in GetCapabilities operation request did not include any version supported by this server. (We recommend placing supported version numbers in the **ExceptionText**)	None. Omit **locator** attribute.
InvalidUpdateSequence	Value of the optional **UpdateSequence** parameter in **GetCapabilities** is greater than current value of service metadata **UpdateSequence** number.	Name of option not supported.
OptionNotSupported	Request is for an option that is not supported by this server.	Name of option not supported.
NoApplicableCode	None of the other exceptionCode values specified by this service match the exception and the error was server specific. (We highly recommend a detailed **ExceptionText**).	None. Omit **locator** attribute.

12.2 Examples of Exception Reports

Simple exception report:

```
<?xml version="1.0" encoding="UTF-8"?>
<ExceptionReport
   xmlns="http://www.opengis.net/ows/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/ows/2.0
http://schemas.opengis.net/ows/2.0/owsAll.xsd"
   version="1.0.2"
   xml:lang="en-US"
>
<Exception exceptionCode="OperationNotSupported" locator="GetExtendedCapabilities">
<ExceptionText>operation name GetExtendedCapabilities not recognized</ExceptionText>
</Exception>
</ExceptionReport>
```

Multiple exceptions in one exception report:

```
<?xml version="1.0" encoding="UTF-8"?>
<ExceptionReport
   xmlns="http://www.opengis.net/ows/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/ows/2.0
http://schemas.opengis.net/ows/2.0/owsAll.xsd"
   version="1.0.2"
   xml:lang="en-US"
>
<Exception exceptionCode="MissingParameterValue" locator="version" />
<Exception exceptionCode="InvalidParameterValue" locator="collection" />
</ExceptionReport>
```

It is highly recommended that an exception report consist of details in the **ExceptionText** element, specifically when the response MIME type is XML.

13 Request Grammar

The first step to communicating with any one of the services is to negotiate with the server as to what grammar is supported from the specification and what features have been implemented by the server implementation of the service. The following topics discuss the specification that applies to all services.

The grammar formalized the structure of each request and reply. It also formalizes how exceptions are handled.

A client begins by communicating with a service, asking it what its capabilities are, i.e. what the server is servicing. Based on the reply, the client then begins accessing data in a form that is meaningful to both the client and the server.

All WAMI services follow at least the specified grammar. They shall have the requests and exception handling implemented as shown in the following specification.

13.1 Required Parameters

Each operations request to a service shall identify the service as a value to the **Service** parameter. The name of the operations request shall be specified as a value to the **Request** parameter.

There are two types of requests. The first request type provides information on all other requests. It is the **GetCapabilities** request. The second type of requests comprise of service specific requests.

The **GetCapabilities** request shall require parameters **Service** and **Request** as mandatory parameters. All other requests require **Service**, **Request** and **Version** as mandatory parameters. All other parameters, mandatory or optional, are service specific.

A few examples of the KVP encoding of a request are shown below:

```
http://example.com/path?SERVICE=CS&REQUEST=GetCapabilities
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&VERSION=1.0.2
http://example.com/path?SERVICE=CS&REQUEST=GetCapabilities&ACCEPTFORMATS=application/xml,applicat
ion/json
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&VERSION=1.0.2&FORMAT=application/xml
```

13.2 Optional Parameters

The **Format** and **Exceptions** parameters shall be implemented by both the client and the server. Unless explicitly stated as required, these are optional parameters for all requests except the **GetCapabilities** request. In case of the **GetCapabilities** request, **Exceptions** and **Format** are not part of the grammar syntax.

Both can have one possible value from a list of possible values. All possible values and the default value supported by the server are provided to the client in the Capabilities response for that service under the **AllowedValues** and **DefaultValue** elements. If a service delivers a response in text format, it shall support **XML** as the default. If the client does not set the **Format** or **Exceptions** parameter in a request, the server shall use the default value.

13.2.1 Format

This parameter specifies the target format that the request should encode the response in. Note that in some circumstances, this is the actual HTTP response Content-Type (such as application/xml, or image/jpeg). In other circumstances, the response needs to be mime-multi-part encoded with a fixed application/xml HTTP Content-Type, but the individual returned items would honor the format parameter.

Valid values are IETF MIME types.

13.2.2 Exceptions

If the response of a request was an exception, the value to this parameter sets the format in which details regarding the exception are delivered.

Table 9: Possible values for the Exceptions parameter

Value	Meaning
XML	Ignores the MIME type set in the **Format** parameter and returns exceptions in XML format. At least XML shall be supported. **Example**: A **GetMap** request in the **Image Service** request may set **Format=image/jpeg** but if **Exceptions=XML**, then in case of an exception the server implementation shall return a response content type of **application/xml**.
IMAGE	Applicable only if MIME type of the **Format** parameter is of type **image/[something].** It means if the request generates an exception, return the response as an image with the exception string embedded in the image. **Example**: If, in a **GetMap** request from an **Image Service**, **Format=image/jpeg** and **Exceptions=IMAGE**, then in case of an exception the server implementation shall return a response content type of **image/jpeg** that has the exception message string embedded in the image. The rendering of the text is governed by the server implementation or the specification. **Warning**: If MIME type of the response is text and **Exceptions**=**IMAGE**, this is not permitted. In this case, the server shall ignore the parameter and assume the default.

NONE	In case of an exception just return the appropriate response header with no body.
Other	Same as XML, except this may be some other standard format. Shall be extensively document by the server implementation.

A few examples of the KVP encoding of a request are shown below:

```
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&VERSION=1.0.2
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&VERSION=1.0.2&FORMAT=application/xm
l&EXCEPTION=XML
http://example.com/path?SERVICE=CS&REQUEST=GetHelp&VERSION=1.0.2&FORMAT=text/html&EXCEPTION=X
ML
http://example.com/path?SERVICE=IS&REQUEST=GetMap&VERSION=1.0.2&FORMAT=image/png&EXCEPTION=IM
AGE&LAYERS="a,b,c"&STYLES=&SRS=TRANSPARENT=TRUE&BBOX=...
```

14 Common Service Model

14.1 Summary

A summary of all requests common to all services is specified in the table below:

Table 10: Requests common to all WAMI services

Request	Meaning	Parameters	Optionality
GetCapabilities	Allows any client to retrieve metadata about the capabilities provided by any server that implements any WAMI specification	Service, Request, AcceptVersions, Sections, UpdateSequence, AcceptFormats, AcceptLanguages	Mandatory
GetHelp	Allows any client to retrieve help documentation about any server that implements any WAMI specification	Service, Request, Version, Topic, Format, Exceptions	Optional

15 Common Operations

15.1 GetCapabilities

The mandatory **GetCapabilities** operation allows any client to retrieve metadata about the capabilities provided by any server that implements any WAMI services specification. The normal response to the **GetCapabilities** operation is a service metadata document that is returned to the requesting client. This service metadata document primarily contains metadata about the specific server abilities (such as about the specific data and formats available from that server). This service metadata also makes a WAMI server partially self-describing, supporting late binding of clients.

15.1.1 Request Parameter Summary

The following table describes request parameters for **GetCapabilities**.

Table 11: GetCapabilities request URL parameters

Parameter name	KVP Example	Optionality and multiplicity	Definition, format and use
Service	`Service=CS`	Mandatory, one	Abbreviated service identifier text. Shall be implemented by both client and server.
Request	`Request= GetCapabilities`	Mandatory, one	Operation name text. Shall be implemented by both client and server.
AcceptVersions	`AcceptVersions= 1.0.0,1.0.2`	Optional, zero or one	Prioritized sequence of one or more specification versions that the client accepts, with preferred versions listed first. When omitted, return the latest supported version. Should be implemented by clients. Shall be implemented by servers.
Sections	`Sections= Contents`	Optional, zero or one	Comma-separated un-ordered list of zero or more names of sections of service metadata document to be returned in service metadata document. May be implemented by client and server. If not implemented, expect/provide default response.
UpdateSequence	`UpdateSequence= XXX (where XXX is a character string previously provided by the server)`	Optional, zero or one	Service metadata document version, value is “increased” whenever any change is made in complete service metadata document. May be implemented by client and server. If not implemented, expect/provide default response.
AcceptFormats	`AcceptFormats= application/xml, text/xml, application/json`	Optional, zero or one	Prioritized sequence of zero or more response formats desired by client, with preferred formats listed first. When omitted or not supported by server, return response using MIME type application/xml. May be implemented by client and server. If not implemented, expect/provide default response.
AcceptLanguages	`AcceptLanguages=en-US`	Optional, zero or one	List of languages desired by the client for all human readable text in the response, in order of preference. For every element, the first matching language available from the server shall be present in the response. When not supported by server, return human readable text in a language of the server’s choice. Shall be implemented by multi-lingual servers and clients.

15.1.2 KVP Encoding Example

- `http://example.com/path?SERVICE=CS&REQUEST=GetCapabilities`
- `http://example.com/path?SERVICE=CS&REQUEST=GetCapabilities&ACCEPTVERSIONS=1.0.0,1.0.2`
- `http://example.com/path?SERVICE=CS&REQUEST=GetCapabilities&ACCEPTFORMATS=application/xml&ACCEPTVERSIONS=1.0.2`
- `http://example.com/path?SERVICE=CS&REQUEST=GetCapabilities&ACCEPTFORMATS=application/json,application/xml&ACCEPTVERSIONS=1.0.2`

- `http://example.com/path?SERVICE=CS&REQUEST=GetCapabilities&ACCEPTFORMATS=application/xml&SECTIONS=ServiceIdentification,Contents`

15.1.3 XML Encoding Schema

See **[WAMI XSD]** element "**GetCapabilitiesRequest**"

15.1.4 Request Parameter Details

Each of the parameters has an expected behaviour. It is discussed below.

15.1.4.1 Service

This parameter allows a client to set the name of a service as its value. For WAMI services, it can be one of CS, QS, IS or VS.

15.1.4.2 Request

This parameter allows a client to set the name of the request as its value. For GetCapabilities its value shall be GetCapabilities.

15.1.4.3 AcceptVersions

As the service evolves, version numbers of the specification may change. A client and a server may support one or more versions of a service specification. They may not always match. This parameter allows the client and server to negotiate the best case common version for both. The client sends a comma separated list of version numbers to the server, in order of preference and the server sends back a response in the first client preferred version number that it supports. Version numbers are in the format x.y.z where x, y and z shall be non-negative numbers no more than 99.

Version negotiation rules are defined in the [OGC WSCS] section 7.3.2.

Upon error, **exceptionCode** of **VersionNegotiationFailed** is returned.

15.1.4.4 Sections

The optional **Sections** parameter shall contain a comma-separated un-ordered list of zero or more names of sections of service metadata document to be returned in service metadata document. Sections are names of XML elements within the service metadata XML document. If no names are listed, the service metadata returned may not contain any of the sections that could be listed (e.g. `SERVICE=CS&REQUEST=GetCapabilities&SECTIONS=&ACCEPTFORMATS=application/xml`).

Further **Sections** usage is defined in the [OGC WSCS] section 7.3.3.

If a server receives a request without the **Sections** parameter, **Sections** is assumed to have the value of **All.** That is the default behavior.

Client implementation of **Sections** is optional. If a client does not contain **Sections** in its request, default behavior is assumed. Server implementation of **Sections** is optional as well. If the server has not implemented **Sections**, the **Sections** parameter within the client's request is ignored and default behavior is assumed.

Allowed values for **Sections** are shown in the table below.

Table 12: Meaning of section names in Sections parameter

Section name	Meaning
ServiceIdentification	Return the ServiceIdentification element in service metadata document
ServiceProvider	Return the ServiceProvider element in service metadata document
OperationsMetadata	Return the OperationsMetadata element in service metadata document
Contents	Return the Contents element in service metadata document
Languages	Return the Languages element in service metadata document
All	Return the complete service metadata document with all elements

15.1.4.5 UpdateSequence

A client may choose to cache a **GetCapabilities** response of the service metadata document. This parameter value may be an integer, a time stamp in ISO 8601:2004 format, or any other number or string. A server may include an **UpdateSequence** metadata value in its service metadata document. If supported, the **UpdateSequence** value shall be increased by the server to indicate any changes in the service metadata document.

None of the WAMI services provide metadata on the content of the data being served as part of the **GetCapabilities** reply. As a result, UpdateSequence may change if the server chooses to provide additional features or parameters and is updated.

Further **UpdateSequence** usage is defined in the [OGC WSCS] section 7.3.4.

15.1.4.6 AcceptFormats

The optional **AcceptFormats** parameter may be used by a client to attempt to negotiate a **GetCapabilities** operation response format other than "application/xml". When included in an operation request, this parameter shall contain a list of the alternative MIME types that the client wants to be returned, listed in the client's preferred order. The MIME type "application/xml" is always an implicit last option, but may be explicitly included.

When a server implements the **AcceptFormats** parameter and receives a value for it, the server shall return the Capabilities document i.e. the service metadata document in the format of the first MIME type in this list that it is capable of returning. When not received or not implemented, the server shall return the Capabilities document in normal XML, using the MIME type "application/xml". All clients and servers shall implement the "application/xml" MIME type for the GetCapabilities operation. Since "application/xml" is always an implicit last option, the server always has an implemented MIME type to use to return a Capabilities document to the client.

Further **AcceptFormats**usage is defined in the [OGC WSCS] section 7.3.5.

15.1.4.7 AcceptLanguages

This parameter is meant for multi-lingual client and servers. Further **AcceptLanguages**usage is defined in the [OGC WSCS] section 7.3.6.

15.1.5 Response

A valid response to a valid GetCapabilities request returns what is known as a service metadata document. In case of an error, an exception report is returned as an XML document. The service metadata document shall contain metadata appropriate to the specific server for the specific WAMI service. For a server with tightly coupled data that it serves or uses, this service metadata document shall include metadata about that data. However, all WAMI services are loosely coupled with the data they serve and provide an explicit interface to getting that data. That service metadata document shall be encoded in XML, and shall use XML Schemas to specify the correct document contents and organization.

15.1.5.1 XML Response schema

See **[WAMI XSD]** element "**ExceptionReport"** for requests with errors.

See **[WAMI XSD]** element "**Capabilities"** for normal application/xml result content.

15.1.5.2 Exceptions

An error in a **GetCapabilities** request may generate the following **exceptionCodes**:

1. MissingParameterValue
2. InvalidParameterValue
3. VersionNegotiationFailed
4. InvalidUpdateSequence
5. NoApplicableCode

It is highly recommended that each exception report contain a list of all exceptions in the request and that each exception contain a detailed error description as a value to the **ExceptionText** element.

15.1.5.3 Example GetCapabilities Response

```
<?xml version="1.0" encoding="UTF-8"?>
<wami:Capabilities
        xmlns:wami="http://www.pixia.com/wami/v101"
        xmlns="http://www.opengis.net/ows/2.0"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.pixia.com/wami wami_1_0_1.xsd
                http://www.opengis.net/ows/2.0 owsAll.xsd
                http://www.w3.org/1999/xlink xlinks.xsd"

        version="1.0.2"
        updateSequence="1"
>
<ServiceIdentification>
<Title>Collection Service</Title>
<ServiceType>CS</ServiceType>
```

```
<ServiceTypeVersion>1.0.2</ServiceTypeVersion>
</ServiceIdentification>
<ServiceProvider>
<ProviderName>Acme Corp</ProviderName>
<ServiceContact>
<IndividualName>John Doe</IndividualName>
</ServiceContact>
</ServiceProvider>
<OperationsMetadata>
<Operation name="GetCollectionCount">
<DCP>
<HTTP>
<Get xlink:href="http://example.com/CS?"/>
</HTTP>
</DCP>
<Parameter name="Service">
<AllowedValues>
<Value>CS</Value>
</AllowedValues>
</Parameter>
<Parameter name="Request">
<AllowedValues>
<Value>GetCollectionCount</Value>
</AllowedValues>
</Parameter>
<Parameter name="Version">
<AllowedValues>
<Value>1.0.2</Value>
</AllowedValues>
</Parameter>
<Parameter name="Format">
<AllowedValues>
<Value>application/xml</Value>
<Value>application/x-json</Value>
</AllowedValues>
<DefaultValue>application/xml</DefaultValue>
</Parameter>
<Parameter name="AcceptLanguages">
<AllowedValues>
<Value>en-US</Value>
<Value>en</Value>
</AllowedValues>
<DefaultValue>en-US</DefaultValue>
</Parameter>
<Parameter name="NID">
<AnyValue/>
<Meaning>Client sets one Node ID.
                    The service returns collection count information starting from this node.
                    If not set or empty, implies root node.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Depth">
<AllowedValues>
<Value>1</Value>
<Value>All</Value>
</AllowedValues>
<DefaultValue>All</DefaultValue>
<Meaning>Specifies how deep the collection counter shall go down the tree.
                    It has two possible values: 1 or All.
                    If not set or empty, implies Depth=All.
                    Shall be implemented by both client and server.
                    1: Client received number of child nodes under root or specified node.
                    All: Client receives all the node counts below specified node.
</Meaning>
</Parameter>
```

```
</Operation>
<Operation name="GetCollections">
<DCP>
<HTTP>
<Get xlink:href="http://example.com/CS?"/>
</HTTP>
</DCP>
<Parameter name="Service">
<AllowedValues>
<Value>CS</Value>
</AllowedValues>
</Parameter>
<Parameter name="Request">
<AllowedValues>
<Value>GetCollections</Value>
</AllowedValues>
</Parameter>
<Parameter name="Version">
<AllowedValues>
<Value>1.0.2</Value>
</AllowedValues>
</Parameter>
<Parameter name="Format">
<AllowedValues>
<Value>application/xml</Value>
<Value>application/x-json</Value>
</AllowedValues>
<DefaultValue>application/xml</DefaultValue>
</Parameter>
<Parameter name="AcceptLanguages">
<AllowedValues>
<Value>en-US</Value>
<Value>en</Value>
</AllowedValues>
<DefaultValue>en-US</DefaultValue>
</Parameter>
<Parameter name="NID">
<AnyValue/>
<Meaning>Client sets one Node ID.
                    The service returns collection count information starting from this node.
                    If not set or empty, implies root node.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Metadata">
<AnyValue/>
<Meaning>
                    A comma-separated ordered list of zero or more names of sections of metadata
to be returned.
                    If not set or empty, it means do not send metadata with node information.
                    If set to All, it means send all metadata with node information.
                    At least the value of All shall be implemented.
                    A complete list shall be provided as part of the Capabilities response to the
GetCollections request.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Depth">
<AllowedValues>
<Value>0</Value>
<Value>1</Value>
<Value>All</Value>
</AllowedValues>
<DefaultValue>0</DefaultValue>
<Meaning>Specifies how deep the collection counter shall go down the tree.
                    It has three possible values: 0, 1 or All.
                    If not set or empty, implies Depth=0.
                    Shall be implemented by both client and server.
                    0: only send information about this node.
                    1: Client received number of child nodes under root or specified node.
                    All: Client receives all the node counts below specified node.
</Meaning>
</Parameter>
</Operation>
</OperationsMetadata>
<wami:Language>en-US</wami:Language>
<wami:Language>en</wami:Language>
</wami:Capabilities>
```

15.2 GetHelp

This request is not a standard OGC recommended request. However, over the years the one request that all client developer had from the authors was clear documentation associated with various aspects of a specific service. The result of these requests is the **GetHelp** request. This request enables a server implementation to provide documentation or a link to documentation via the web services specification. Any additional help such as sample source code or sample clients to implement a service may also be provided via this mechanism.

A server may choose not to implement this request. If it is implemented, a corresponding **Operation** element shall be available in the **OperationsMetadata** section.

15.2.1 Request Parameter Summary

A table of the request parameters for **GetHelp** is provided below.

Table 13: GetHelp request URL parameters

Parameter name	KVP Example	Optionality and multiplicity	Definition, format and use
Service	`Service=CS`	Mandatory, one	Abbreviated service identifier text. Shall be implemented by both client and server.
Request	`Request= GetHelp`	Mandatory, one	Operation name text. Shall be implemented by both client and server.
Version	`Version=1.0.2`	Mandatory, one	Version number of service. Shall be implemented by both client and server
Format	`Format= application/xml`	Optional, zero or one	MIME type of the format in which the server shall provide the response. A list of supported MIME types shall be made available in the Capabilities response under the Operation XML element. The default format shall also be specified in the corresponding Capabilities response.
Exceptions	`Exceptions=XML`	Optional, zero or one	Sets the format of the exception.
Topic	`Topic= Specifications`	Optional, zero or one	Name of a help topic. A list of supported help topics shall be made available in the Capabilities response. This parameter shall specify a topic that the client chooses to request help on. The server shall support sending a valid response to all values it lists as

			allowed values to this parameter. The default format shall also be specified in the corresponding Capabilities response.
AcceptLanguages	`AcceptLanguages =en-US`	Optional, zero or one	List of languages desired by the client for all human readable text in the response, in order of preference. For every element, the first matching language available from the server shall be present in the response. When not supported by server, return human readable text in a language of the server's choice. Shall be implemented by multi-lingual servers and clients.

15.2.2 KVP Encoding Example

- `http://example.com/path?SERVICE=CS&REQUEST=GetHelp&VERSION=1.0.2&FORMAT=application/xml&TOPIC=CSInterfaceSpecifications`
- `http://example.com/path?SERVICE=CS&REQUEST=GetHelp&VERSION=1.0.2&FORMAT=application/xml&TOPIC=OpenSourceClientLinks`
- `http://example.com/path?SERVICE=CS&REQUEST=GetHelp&VERSION=1.0.2&FORMAT=application/xml&TOPIC=ServerSetupGuide`
- `http://example.com/path?SERVICE=CS&REQUEST=GetHelp&VERSION=1.0.2&FORMAT=application/xml&TOPIC=SupportBlog`

15.2.3 XML Encoding schema

See **[WAMI XSD]** element "**GetHelpRequest**"

15.2.4 Request Parameter Details

Each of the parameters has an expected behavior. It is discussed below.

15.2.4.1 Service

This parameter allows a client to set the name of a service as its value. For WAMI services, it can be one of CS, QS, IS or VS.

15.2.4.2 Request

The value of this required parameter should be **GetHelp**.

15.2.4.3 Version

The value of this required parameter specifies an explicit version number of the service. The version number is set by the **version** attribute in the Capabilities response. A server implementation shall look at this version number and use the corresponding service specification version to perform syntax and semantic checks.

15.2.4.4 Format

The value of this optional parameter specifies the MIME type in which the server returns the response to a **GetHelp** request. If not specified, the default value is used. The possible values and the default value supported by the server are provided to the client in the Capabilities response. MIME types are defined and maintained by the IANA. For **GetHelp**, the value of `application/xml` is required and `text/html` is recommended but optional. If not specified, `application/xml` shall be the default value.

15.2.4.5 Exceptions

Default is XML. See Exception Handling and the grammar of Exceptions.

15.2.4.6 Topic

The value of this optional parameter specifies the name of the topic that the client wishes to get help on. A list of valid topic names is provided to the client as part of the Capabilities response. The default value is also supplied as part of the Capabilities response.

15.2.5 Response

The response from a **GetHelp** request can be in a format of the clients choosing from a list of formats that the server supports. If the response is of MIME type application/xml, then the XSD for the response is specified below.

15.2.5.1.1 XML Response schema

See **[WAMI XSD]** element "**ExceptionReport"** for requests with errors.

See **[WAMI XSD]** element "**GetHelp"** for normal application/xml result content.

15.2.5.2 Exceptions

An error in a **GetHelp** request may generate the following **exceptionCodes**:

1. MissingParameterValue
2. InvalidParameterValue
3. OptionNotSupported
4. NoApplicableCode

15.2.5.3 Example GetHelp Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Help
   xmlns="http://www.pixia.com/wami/v101"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.pixia.com/wami wami_1_0_1.xsd"
   lang="en" service="CS"request="GetMap"version="1.0.2">
<h1>Help Title</h1>
<dl>
<dt>Topic 1</dt>
<dd>Data 1</dd>
<dt>Topic 2</dt>
<dd>Data 2</dd>
</dl>
</Help>
```

16 Extensions

All WAMI services shall have the capability to provide the grammar for extensions. This section provides the frame work for a vendor to provide their own options to extend the interface.

As WAMI services evolve, additional parameters may have to be added to the interface specification. However, the process of changing the specification requires industry participation and may take time. In the meanwhile, the implementation may suffer due to the inflexibility of the specification to cater to their needs. The set of features provided by a vendor depends on the skill set of that vendor and the requirements presented to that vendor from the customer.

For example, consider the image service. The image service provides the client with the ability to request for areas of interest from a large WAMI raster in a format of choice. If the source data was a 16-bit 1-band Intensity or 3-band RGB image, then the vendor may provide the ability for the client to select a specific curve or technique for 16-to-8 bit conversion. The vendor may choose to support JPEG, PNG, GIF, GeoTIFF, NITF, HFA, and J2C as the supported formats. For JPEG, there are a multitude of sub-options such as quality settings, YUV encoding selection (420, 411, 422, or 444), bit rate, quality layers, etc. For PNG, one can request a 24-bit PNG or an 8-bit PNG with a set of supported methods to perform dithering. The same is true for GeoTIFF, NITF, HFA and J2C formats.

The implementation could be something as simple as extending parameters for the MIME types for each of the formats and providing a list of possible combinations. However, this list may get to be rather long. For example, JPEG image quality could be an integer from 1 to 100.

For a video service, the quantity, interdependency of options for each video format is staggering. The vendor can choose to expose a subset of available features through an enumerated list of well-named options and/or provide a power-user with a lot more controls and consequently, greater flexibility.

As seen, providing custom features set is completely vendor dependent.

Options will be defined as extra Parameter sections to a given Capabilities service request with the prefix “Options” in the parameter-name.

For example,to add jpeg-compression levels to ImageService GetMap, in the OperationsMetadata section for the GetMap Operationyou could have:

```
<Operation name="GetMap">
  …
<Parameter name="Options.jpeg_compression">
<AllowedValues>
<Range>
<MinimumValue>1</MinimumValue>
<MaximumValue>100</MaximumValue>
</Range>
</AllowedValues>
</Parameter>
</Operation>
```

The way in which to specify this parameter in a KVP request would be to use the period (.) between the parameter name and its parents. A fully qualified parameter name shall identify the parameter completely. Its value will follow the equal sign.

For example, consider the **Format** parameter for Video Service. The server may choose to set possible values to this parameter as the IETF approved MIME types of **video/mpeg**, **video/quicktime**, and **video/x-msvideo**.

Let us pick the **Format=video/mpeg**. Within this format, let us say, the server supports MISB compliant MPEG2 transport streams. Within this transport stream, let us say, it supports H.264 and MPEG2 encoding. For H.264 and MPEG2, it supports bit rate, GOP size, playback frame rate, and Chroma sub-sampling. How can we have a list of parameters that allow for the client to set these values without making it a part of the specification? Example features:

- `Format=video/mpeg`
- `Video.Mpeg=Mpeg2`
- `Video.Mpeg.Mpeg2.Codec=H264`
- `Video.Mpeg.Mpeg2.Codec.H264.BitRate=2500`
- `Video.Mpeg.Mpeg2.Codec.H264.GOP=10`
- `Video.Mpeg.Mpeg2.Codec.H264.FrameRate=29.97`
- `Video.Mpeg.Mpeg2.Codec.H264.ChromaSubSampling=422`

We can do this in one of two ways. Both ways require the Capabilities response to send back the syntax of each parameter as a parameter tree. The “.”character should be used to traverse the hierarchy to make a fully qualified unique parameter name. Values have data types, ranges, defaults and constraints; however they are not defined as part of the specification. They are defined as part of the Capabilities response from the service implementation and follow a specification.

Note that [OGS WSCS] section 11.5.5 specifically limits parameters to a deterministic finite set. However, this is to satisfy the goal of constructing a WSDL file with input request parameters. The WAMI spec defines BOTH a KVP parameter space and an XML request document parameter space. Any WSDL resource will leverage the XML parameter space, wherein only a single Options element contains the entire hierarchy of options.

16.1 KVP Options example

```
FORMAT=video/mpeg&OPTIONS.Video.Mpeg=Mpeg2&OPTIONS.Video.Mpeg.Mpeg2.Codec=H264&OPTIONS.Video.
Mpeg.Mpeg2.Codec.H264.BitRate=2500&OPTIONS.Video.Mpeg.Mpeg2.Codec.H264.GOP=10&OPTIONS.Video.M
peg.Mpeg2.Codec.H264.FrameRate=29.97&OPTIONS.Video.Mpeg.Mpeg2.Codec.H264.ChromaSubSampling=42
2&STYLES=&BGCOLOR=0x000000
```

16.2 XML Options example

```
<VS_GetMapVideoRequest xmlns="http://www.pixia.com/wami/v101"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.pixia.com/wami wami_1_0_1.xsd
                 http://www.opengis.net/ows/2.0 owsAll.xsd
                 http://www.w3.org/1999/xlink xlinks.xsd"

        service="VS"
        request="GetMapVideo"
        version="1.0.2"
        crs="wgs84"
        format="video/mpeg"
        styles=""
        bgcolor="0x000000"
        CID="collection1"
        dup="1"
>
<Option name="Options.Video.Mpeg">Mpeg2</Option>
<Option name="Options.Video.Mpeg.Mpeg2.Codec">H264</Option>
<Option name="Options.Video.Mpeg.Mpeg2.Codec.H264.BitRate">2500</Option>
<Option name="Options.Video.Mpeg.Mpeg2.Codec.H264.GOP">10</Option>
<Option name="Options.Video.Mpeg.Mpeg2.Codec.H264.FrameRate">29.97</Option>
<Option name="Options.Video.Mpeg.Mpeg2.Codec.H264.ChromaSubSampling">422</Option>
<ViewPort width="512" height="512"/>
<BBox crs="wgs84" minx="100" miny="100" maxx="100.1" maxy="100.1"/>
<Time>
<FrameRange>
<Start>5</Start>
<End>100</End>
</FrameRange>
</Time>

</VS_GetMapVideoRequest>
```

17 Section 2: Collection Service

The name of this service is **CS** or **Collection Service**. A **collection** is defined as a collection of WAMI data such that each element or frame of WAMI data is temporally sequential from the previous element. An **element** or **frame** of WAMI data can be a single image or a group of images that constitute a single logical picture of an instant in time. How these elements are stored on the server side is implementation and vendor specific and not associated with the specifications themselves.

The collection service or CS allows a client to discover what collections of WAMI data are being served by a server implementation. Due to indirect collection reference capability and incremental traversal, a CS can be implicitly federated to link up with a hierarchy of CS.

18 CS data model

18.1 Collection Tree

This section draws heavily from the industry as well as multiple OGC specifications of presenting the organization of a set of maps, metadata, raster data, vector data, and video. Hierarchical data is generally presented in the form of an inverted tree structure.

In computer science a tree is a widely-used data structure that emulates a hierarchical tree structure with a set of linked nodes.

For the purpose of this specification, it is a data structure that is an *ordered directed tree*, more specifically *arborescence*: a connected acyclic graph where each node has zero or more *children* nodes and at most one *parent* node. Furthermore, the children of each node have a specific order.

A node is a structure which may contain a value, a condition or represent a separate data structure. A *childnode* is hierarchically below a node and a node that has a child is the *parentnode* of that child. Nodes without children are *leaf nodes* or *terminal nodes*. Nodes with children are also known as *inner nodes*. The first node of the tree has zero parent nodes and is the *root node*. A connection between nodes is called an *edge* or a *link*. The path from the root node to any node is unique and may be traversed through the links. Each collection (as defined above), is unique from all other collections.

A general tree structure is shown in Figure 4. In this example, to the right you can also see a possible organizational structure that may get used practically.

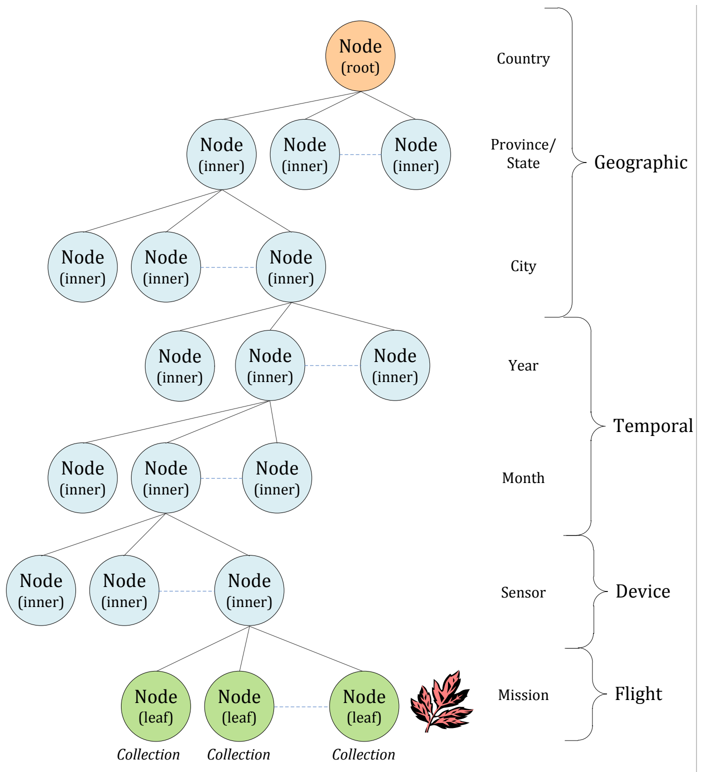


Figure 4: Example of nodes in a tree of Collections

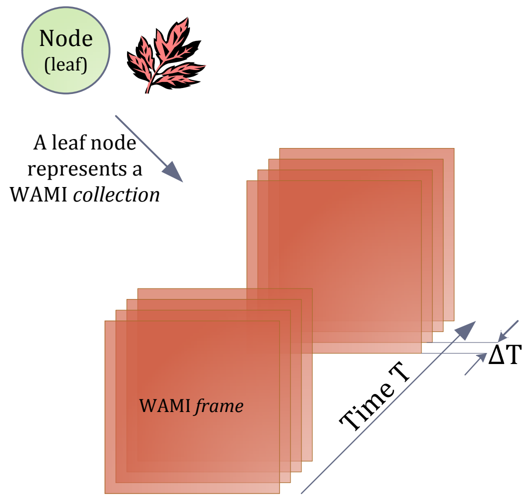


Figure 5: A leaf node represents a WAMI Collection

All nodes in the collection tree have the same general structure. Root nodes do not have a parent or links to collections. Inner nodes do not have links to collections. Leaf nodes do not have children.

Table 14: Members of a node's structure

Node Members	Meaning	Mandatory/Optional/Absent		
		Root	Inner	Leaf
ParentNID	NID of the parent node and a URL to serve it up. A root node does not have this member.	A	M	M
NID	Node Id – this is guaranteed to be unique within a collection-service. It SHOULD be globally unique (if a UUID is utilized) so that collection services can be aggregated together.	M	M	M
CID	Collection ID – this SHOULD be unique within a Collection Service. It may or may not be the same value as NID. This is the value that will be passed to IS, VS, VCSS operations as the C ID parameter. It may be a UUID.	A	A	M
Name	A human readable name of this node	M	M	M
Description	A human readable description of this node	O	O	O

Metadata	Structured metadata about this node	O	O	O
Node	The outer or sub-node which contains an NID, URL(s) to serve it up from the Collection-Service, and in the case of collection-Nodes, URLs to service up the actual collection data (i.e. IS, VS, VCSS). Absent in leaf nodes.	O	O	A
Service	One or more Image or Video Service URLs. Absent in inner and root nodes. This base URL is used to construct a **GetCapabilities**, **GetMap**, etc request.	A	A	O
updateSequence	This may only be supported by a Node or by a Collection (represented by a node), by both, or neither. It is vendor specific because it represents the ability of the WAMI collection service to detect changes of different categories. If a Node supports updateSequence, then it should increase every time an add/remove/move of a child (or descendent) node occurs. Note this does NOT reflect changes to individual leaf-collections, only the existence / placement of said collection. If a Collection (represented here as a Node) supports updateSequence, then it should increase every time the metadata for a collection changes.	A	A	O

18.2 Example XML Encoding of a Collection Tree

Use the GetCollections request to retrieve a collection tree. The following are examples of a portion of the response of a GetCollections request. In the following responses, if the parameter Metadata was set to “All” or any other vendor specific value, the element metadata would be included in the XML response as shown. Otherwise, if the parameter Metadata was not specified in the request or empty, the element metadata is not included.

Root node with no parent:

```
<CS_Collections xmlns="http://www.pixia.com/wami/v101" xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.pixia.com/wami wami_1_0_1.xsd
http://www.opengis.net/ows/2.0 owsAll.xsd                http://www.w3.org/1999/xlink
xlinks.xsd" updateSequence=”1”>
<Node NID="root" name="ROOT" updateSequence=”1”>
<Node NID="2010" name="2010" parentNID="root">
<Description>This section contains all 2010 missions</Description>
<Node NID="2010/Jan" name="Jan Missions" parentNID="2010" updateSequence=”1”>
<Node
           NID="2010/2010-MISSION1"
           CID="2010-MISSION1"
           name="MISSION 1"
           parentNID="2010/Jan"
           updateSequence=”1”>
<Description>Info about Mission 1</Description>
<Service name="IS">
```

```
<ows:DCP>
<ows:HTTP>
<ows:Get xlink:href="http://example.com/CS?NID=2010-MISSION1"/>
</ows:HTTP>
</ows:DCP>
</Service>
</Node>
<Node NID="2010/2010-MISSION2" CID="2010-MISSION2" name="MISSION 2" parentNID="2010/Jan">
<Description>Info about Mission 2</Description>
<Service name="IS">
<ows:DCP>
<ows:HTTP>
<ows:Get xlink:href="http://example.com/CS?NID=2010-MISSION1"/>
</ows:HTTP>
</ows:DCP>
</Service>
</Node>
</Node>
</Node>
</Node>
</CS_Collections>
```

Inner node with parent:

```
<CS_Collections xmlns="http://www.pixia.com/wami/v101" xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.pixia.com/wami wami_1_0_1.xsd
http://www.opengis.net/ows/2.0 owsAll.xsd                 http://www.w3.org/1999/xlink
xlinks.xsd">
<Parent NID="2010">
<Service name="CS">
<ows:DCP>
<ows:HTTP>
<ows:Get xlink:href="http://example.com/CS?NID=2010"/>
</ows:HTTP>
</ows:DCP>
</Service>
</Parent>
<Node NID="2010/2010-MISSION1" CID="2010-MISSION1" name="MISSION 1" parentNID="2010">
<Description>Info about Mission 1</Description>
<Service name="IS">
<ows:DCP>
<ows:HTTP>
<ows:Get xlink:href="http://example.com/CS?NID=2010-MISSION1"/>
</ows:HTTP>
</ows:DCP>
</Service>
</Node>
</CS_Collections>
```

Node fragment with metadata:

```
<Node NID="node7" CID="collection7" parentNID="node6">
<Metadata>
<Collection
          startFrame="0"
          endFrame="15"
          frameCount="16"
          startTime="2010-01-01T15:15:15.0Z"
          endTime="2010-01-01T11:15:15.0Z"
          timeSpan="PT1H"
          frameInterval="PT1S"
          live="false"
          />
<GeoBox nativeCRS="EPSG:4236">
<BoundingBox crs="EPSG:4326" minx="1.0" maxx="2.0" miny="1.0" maxy="2.0" resx="50" resy="50"/>
```

```
</GeoBox>
</Metadata>
</Node>
```

18.3 Client Resources, Bandwidth, and Latency

The number of fielded sensors is constantly growing. For surveillance, reconnaissance and analytics applications, a web server may be required to field a large number of collections. These collections may be organized in a complex tree structure. The networks may be high bandwidth, but with high latency. They may not be high bandwidth either. The interface shall provide a way to send as much data as possible in a single request as well as allow the client to query the data in smaller chunks. The clients may not have enough computing resources such as memory to handle a large tree.

For clients with limited resources, the basic method to follow would be to:

1. Get the number of nodes in the tree and its depth. In other words, get the number of total nodes, leaf nodes, and the depth of the tree.
2. Get the root node and traverse the tree one level at a time.
3. Get metadata on a per-collection basis.
4. Request for compressed XMLs to further save bandwidth.

For clients with reasonable resources and high latency, minimize number of requests:

1. Get the number of nodes in the tree and its depth. In other words, get the number of total nodes, leaf nodes, and the depth of the tree.
2. Get the entire tree, with metadata.
3. Request for compressed XML to reduce incurred cost of bandwidth.

19 CS service model

19.1 Summary

A summary of all the requests for a collection service is specified in the table below:

Table 15: Collection Service request summary

Request	Meaning	Parameters	Optionality
GetCapabilities	Allows the client to retrieve metadata about the capabilities of the CS server implementation	Service, Request, Sections, AcceptVersions, AcceptFormats, AcceptLanguages, UpdateSequence,	Mandatory
GetHelp	Allows the client to retrieve help on topics provided by the server	Service, Request, Version, Topic, Format	Optional
GetCollectionCount	Retrieves the number of collections being served by the server	Service, Request, Version, Format, AcceptLanguages, NID, Depth, Bbox, CRS, Time	Optional

GetCollections	Retrieves the set of collections being served by the server. Optionally, metadata about each collection can also be retrieved. Basic filtering capabilities permit control over quantity of data transmitted.	Service, Request, Version, Format, NID, Metadata, Depth, Bbox, CRS, Time	Mandatory

19.2 Request UML (XML/KVP)

The following shows the root XML elements representing all possible **CS** requests. All UML attributes represent XML and KVP request parameters, their data-type, and multiplicity.

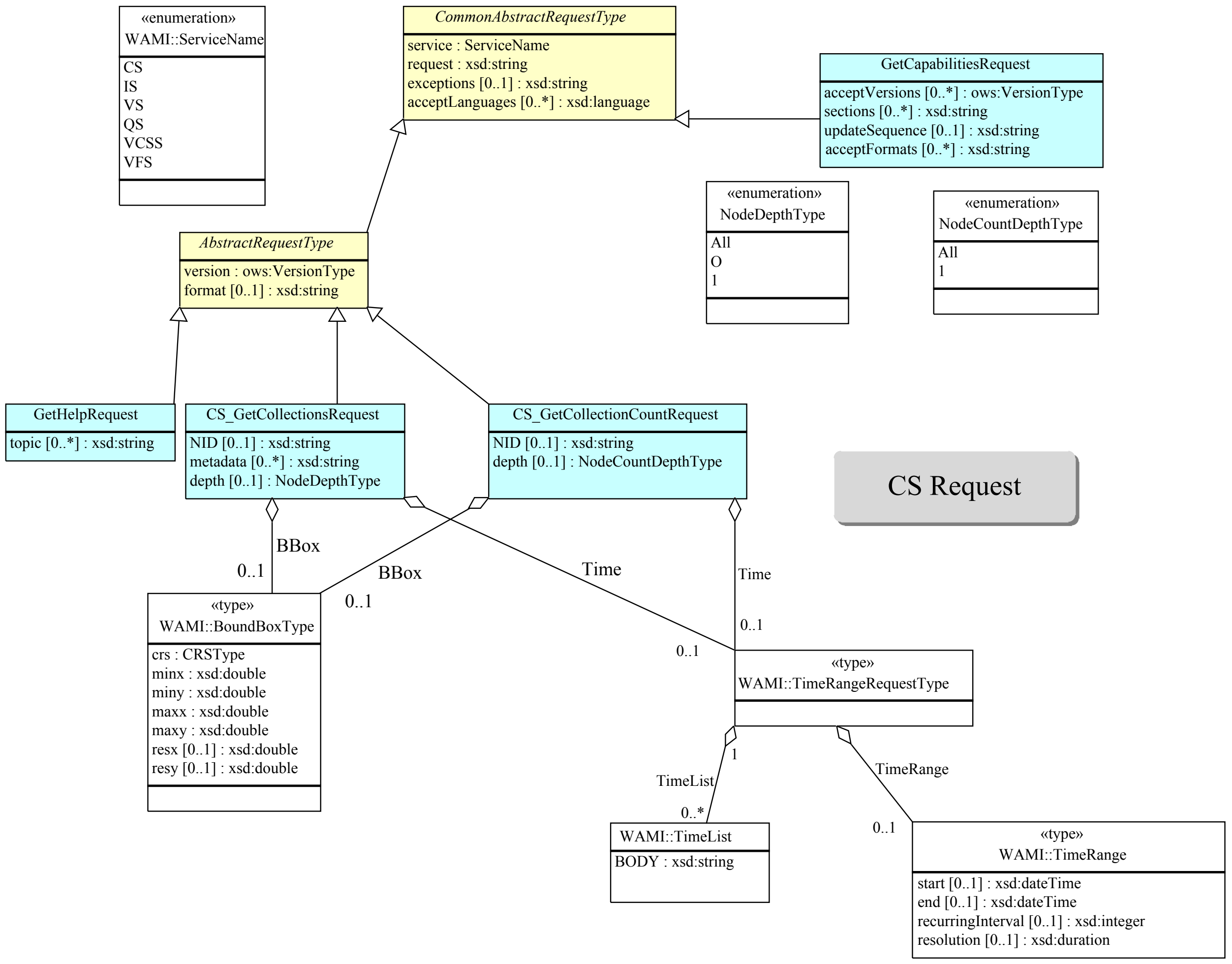

«enumeration»
WAMI::ServiceName
CS
IS
VS
QS
VCSS
VFS
CommonAbstractRequestType
service : ServiceName
request : xsd:string
exceptions [0..1] : xsd:string
acceptLanguages [0..*] : xsd:language
GetCapabilitiesRequest
acceptVersions [0..*] : ows:VersionType
sections [0..*] : xsd:string
updateSequence [0..1] : xsd:string
acceptFormats [0..*] : xsd:string
«enumeration»
NodeDepthType
All
O
1
«enumeration»
NodeCountDepthType
All
1
AbstractRequestType
version : ows:VersionType
format [0..1] : xsd:string
GetHelpRequest
topic [0..*] : xsd:string
CS_GetCollectionsRequest
NID [0..1] : xsd:string
metadata [0..*] : xsd:string
depth [0..1] : NodeDepthType
CS_GetCollectionCountRequest
NID [0..1] : xsd:string
depth [0..1] : NodeCountDepthType
CS Request
BBox
0..1
BBox
0..1
Time
Time
0..1
0..1
«type»
WAMI::BoundBoxType
crs : CRSType
minx : xsd:double
miny : xsd:double
maxx : xsd:double
maxy : xsd:double
resx [0..1] : xsd:double
resy [0..1] : xsd:double
«type»
WAMI::TimeRangeRequestType
1
TimeList
0..*
TimeRange
0..1
WAMI::TimeList
BODY : xsd:string
«type»
WAMI::TimeRange
start [0..1] : xsd:dateTime
end [0..1] : xsd:dateTime
recurringInterval [0..1] : xsd:integer
resolution [0..1] : xsd:duration

Figure 6: CS Request UML

19.3 CollectionMetadata UML (XML)

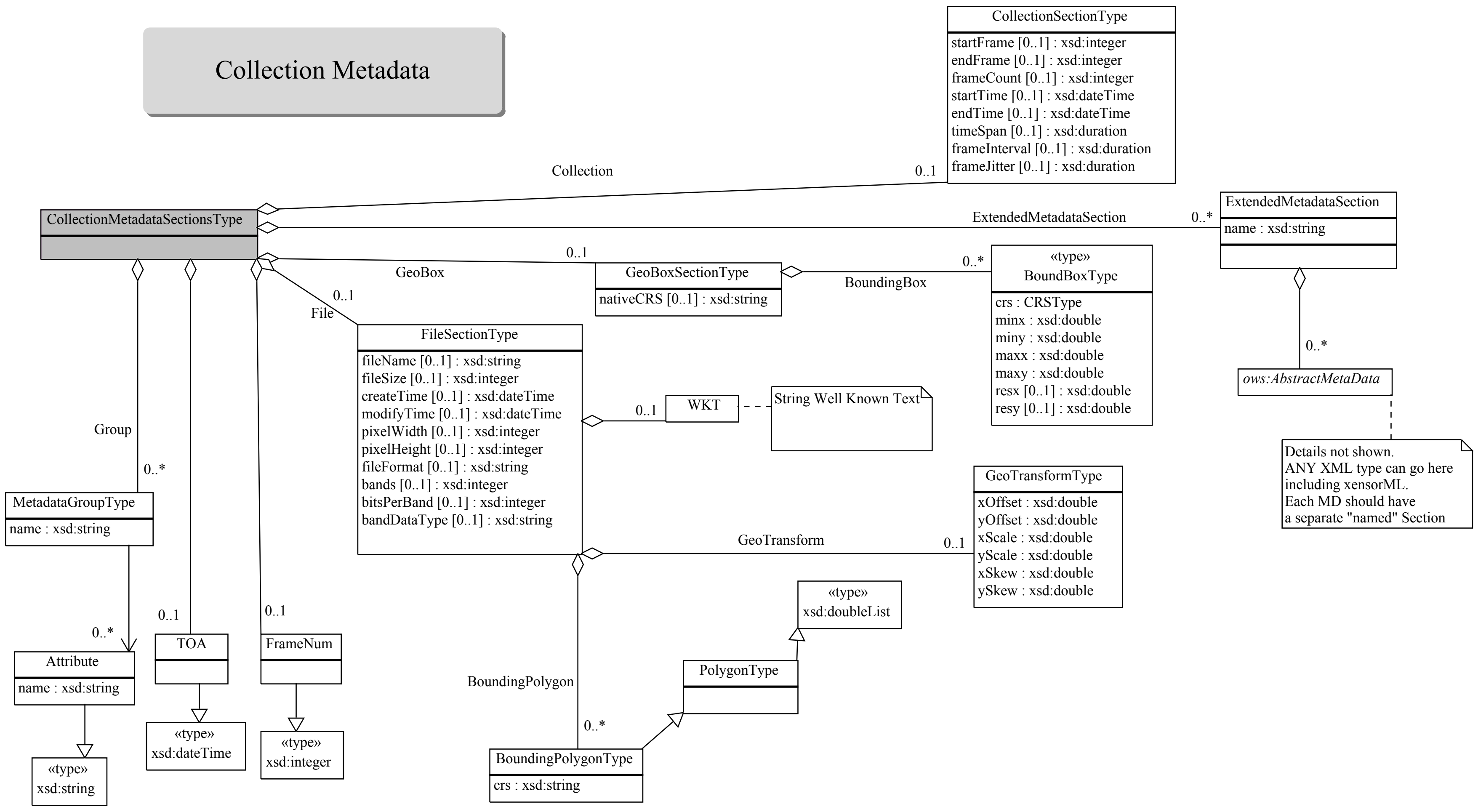


Figure 7: CollectionMetadata UML

19.4 Response UML (XML)

The following shows the root XML elements representing **CS** specific response bodies. Not shown are **GetCapabilities** Service Documents or **GetHelp** response bodies.

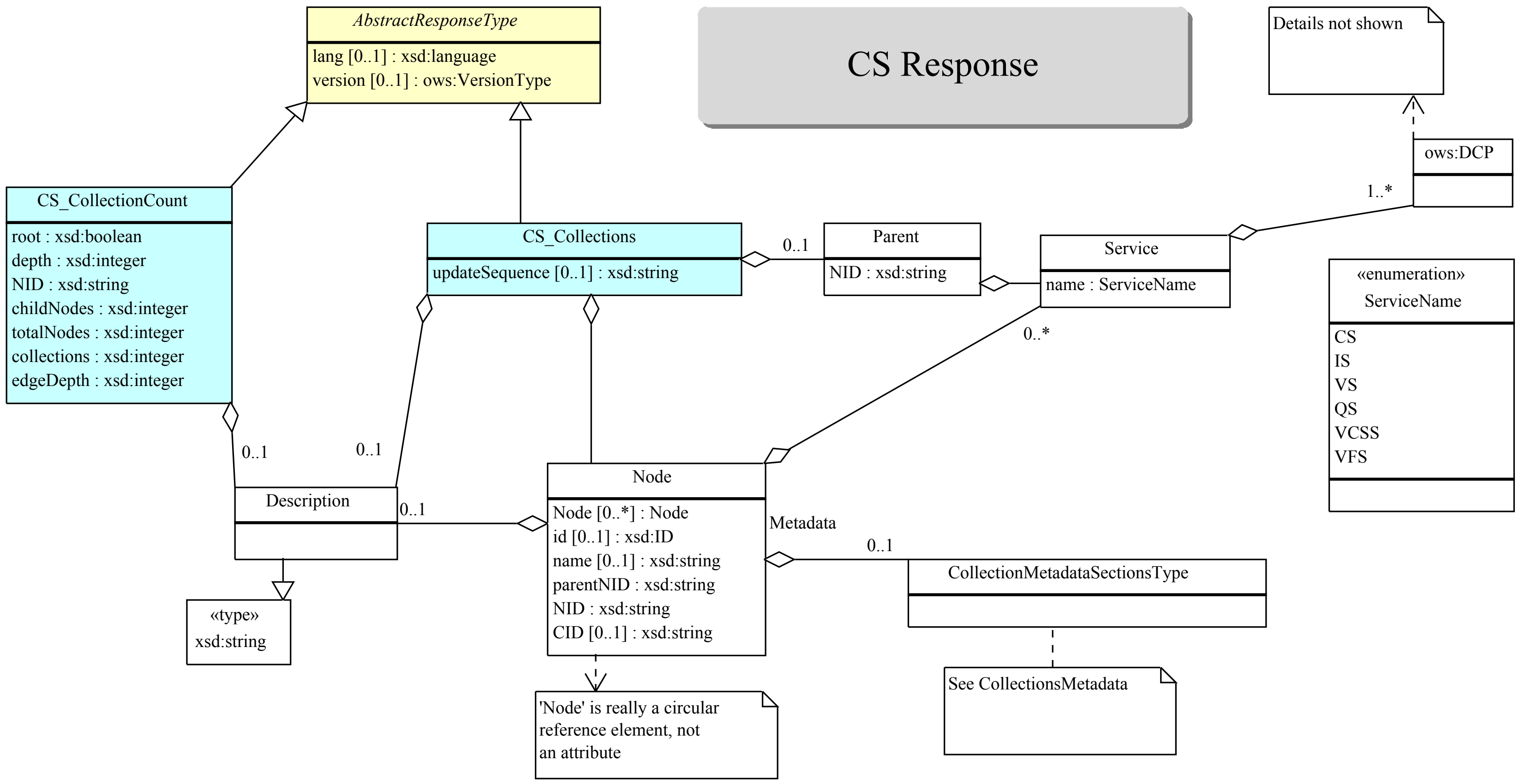


Figure 8: CS Response UML

20 CS operations

20.1 GetCapabilities

See specifications for this required request in **[WAMI OVERVIEW]**.

20.2 GetHelp

See specifications for this required request in **[WAMI OVERVIEW]**.

20.3 GetCollectionCount

This request returns the number of collections being served. It allows the clients to understand what kind of resource allocation they may have to do in order for them to prepare for processing other requests. The response provides three types values:

1. The number of collections being served by this service, i.e. the total number of leaf nodes in the tree.
2. The total number of nodes in the tree, including the root node, inner nodes and leaf nodes.
3. The depth of the tree. The depth of the tree is the total number of edges or links from the root node to the deepest leaf in the tree.

The above pieces of information can be provided starting from the root node, or starting from any node. A node can be selected explicitly by its Node Identifier. A set of nodes can be implicitly selected using a spatiotemporal range. The request shall also provide information about the number of children below one node.

20.3.1 Request Parameter Summary

A table of the request parameters for **GetCollectionCount** is provided below.

Table 16: GetCollectionCount request URL parameters

Parameter name	KVP Example	Optionality and multiplicity	Definition, format and use
Service	`Service=CS`	Mandatory, one	The value for this parameter shall be CS.
Request	`Request= GetCollectionCount`	Mandatory, one	The value for this parameter shall be **GetCollectionCount**.
Version	`Version=1.0.2`	Mandatory, one	Version number of service. Shall be implemented by both client and server
Format	`Format=application/ xml`	Optional, zero or one	Please refer to **[WAMI OVERVIEW]**

AcceptLanguages	`AcceptLanguages=en-US`	Optional, zero or one	Please refer to **[WAMI OVERVIEW]**
NID	`NID=Unique1234`	Optional, zero or one	Client sets one Node ID. The service returns collection count information starting from this node. If not set or empty, implies root node. Shall be implemented by both client and server.
Depth	`Depth=All`	Optional, zero or one	Specifies how deep the collection counter shall go down the tree. It has two possible values: **1** or **All**. If not set or empty, implies Depth=All. The parameter shall be implemented by both client and server. **1**: Client receives number of child nodes under root or specified node. **All**: Client receives all the node counts below specified node.
Bbox	`Bbox=minx,miny,maxx,maxy`	Optional, zero or one	Bbox specifies a bounding box in the reference system set by CRS. The request filters selected nodes in the tree through this bounding box. The parameter shall be implemented by both client and server. This parameter is optional. The default value is “unbounded” to include all collections.If Bbox is set, CRS shall be set. Setting Bbox without CRS, or setting CRS without Bbox indicates error.
CRS	`CRS=epsg:4326`	Optional, zero or one	See Bbox.
Time	`Time=T1/T2,T3/T4,T5`	Optional, zero or one	Specifies one or more time values or time ranges. The request filters selected nodes in the tree through the time values and ranges. The parameter shall be implemented by both client and server. This parameter is optional. The default value is all time. Time is set in ISO 8601:2004 format (not as frame numbers).

20.3.2 KVP encoding example

```
http://example.com/path?SERVICE=CS&REQUEST=GetCollectionCount&VERSION=1.0.2&FORMAT=applicatio
n/xml
http://example.com/path?SERVICE=CS&REQUEST=GetCollectionCount&VERSION=1.0.2&FORMAT=applicatio
n/xml&NID=Unique1234&DEPTH=1
http://example.com/path?SERVICE=CS&REQUEST=GetCollectionCount&VERSION=1.0.2&FORMAT=applicatio
n/xml&NID=Unique1234&DEPTH=All
```

20.3.3 XML Encoding schema

See **[WAMI XSD]** element “**CS_GetCollectionCountRequest**”

20.3.4 Request Parameter Details

20.3.4.1 Service

The value for this parameter shall be **CS**.

20.3.4.2 Request

The value for this parameter shall be **GetCollectionCount**.

20.3.4.3 Version

Please refer to **[WAMI OVERVIEW].**

20.3.4.4 Format

Please refer to **[WAMI OVERVIEW].**

20.3.4.5 AcceptLanguages

Please refer to **[WAMI OVERVIEW].**

20.3.4.6 NID

This parameter is optional. The value of this parameter is one node ID of any node within the tree. If NID is not set, or empty, it implies the root node of the collection tree. If NID is set to a valid value, the service returns collection count information starting from the specified node.

20.3.4.7 Depth

This parameter is optional. The value of this parameter can either be `1` or `All`. If not set, or empty, it implies `Depth=All`. If `Depth=1`, it means, the services returns collection count information about the node set in NID only. The request shall return number of child nodes. Total node count follows the response specification as shown below. Edge depth is 1.

20.3.4.8 CRS

This parameter indicates the coordinate reference system of the values of parameter **BBox**. Detailed specification is as defined in **[OGC OWSCS]**. Also see OGC's WMS and WCS specifications. The Capabilities response of a GetCollectionCount request shall provide a list of CRSs that are supported by the service implementation.

This parameter is optional. If **Bbox** is set, **CRS** shall be set. Setting **Bbox** without **CRS**, or setting **CRS** without **Bbox** indicates error.

20.3.4.8.1 Example Capabilities Response for GetCollectionCount's CRS Parameter

```
<Parameter name="CRS">
<AllowedValues>
<Value>EPSG:4326</Value>
<Value>EPSG:4269</Value>
<Value>EPSG:32641</Value>
<Value>EPSG:32642</Value>
<Value>EPSG:32643</Value>
<Value>EPSG:32644</Value>
<Value>EPSG:32645</Value>
</AllowedValues>
<Meaning>Coordinate reference system for BBox</Meaning>
</Parameter>
```

Example: `CRS=EPSG:4326`

20.3.4.9 BBox

This parameter sets the bounding box of the filter AOI. This parameter indicates the bounding box in the coordinate reference system set as the value of parameter **CRS**. This bounding box may intersect with the bounds of one or more Collections. The service filters out all collections that are wholly outside the bounding box, responding with those wholly or partially within. Detailed specification is as defined in **[OGC WSCS]**. Also see OGC's WMS and WCS specifications.

This parameter is optional. If **Bbox** is set, **CRS** shall be set. Setting **Bbox** without **CRS**, or setting **CRS** without **Bbox** indicates error. If **Bbox** is not specified, it implies there are no bounding box restrictions i.e. **Bbox** is unbounded to include all Collections.

Example: `CRS=EPSG:4326&BBOX=-117.6788847,35.603284,-112.6704312,38.608133`

In this example, the response returns information on all collections whose bounding rectangles at least partially intersect with the bounding box. The bounding box is specified in EPSG:4326 or geographic WGS84.

20.3.4.10 Time

This parameter is optional. The purpose of the **Time** parameter is to let the client filter Collections wholly outside the time window and respond with Collections that intersect with the values of this parameter. The value for **Time** is a *time interval*. If **Time** is not specified, it implies unbounded to include all time. See Section **23.1Time**for details. **Time** is specified as time of acquisition value or interval in ISO 8601:2004 format. It is not specified as absolute frame numbers.

Example: `Time=2008-08-20T00:00:00Z/2009-08-20T00:00:00Z`

In this example, the response returns information on all collections that have at least one frame that was captured between 20th August 2008 and 2009.

20.3.5 Example GetCollectionCount Capabilities Fragment Response

```
<Operation name="GetCollectionCount">
<DCP>
<HTTP>
<Get xlink:href="http://example.com/CS?"/>
</HTTP>
</DCP>
<Parameter name="Service">
<AllowedValues>
<Value>CS</Value>
</AllowedValues>
</Parameter>
<Parameter name="Request">
<AllowedValues>
<Value>GetCollectionCount</Value>
</AllowedValues>
</Parameter>
<Parameter name="Version">
<AllowedValues>
<Value>1.0.2</Value>
</AllowedValues>
</Parameter>
<Parameter name="Format">
<AllowedValues>
<Value>application/xml</Value>
<Value>application/x-json</Value>
```

```
</AllowedValues>
<DefaultValue>application/xml</DefaultValue>
</Parameter>
<Parameter name="AcceptLanguages">
<AllowedValues>
<Value>en-US</Value>
<Value>en</Value>
</AllowedValues>
<DefaultValue>en-US</DefaultValue>
</Parameter>
<Parameter name="NID">
<AnyValue/>
<Meaning>Client sets one Node ID.
                    The service returns collection count information starting from this node.
                    If not set or empty, implies root node.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Depth">
<AllowedValues>
<Value>1</Value>
<Value>All</Value>
</AllowedValues>
<DefaultValue>All</DefaultValue>
<Meaning>Specifies how deep the collection counter shall go down the tree.
                    It has two possible values: 1 or All.
                    If not set or empty, implies Depth=All.
                    Shall be implemented by both client and server.
                    1: Client received number of child nodes under root or specified node.
                    All: Client receives all the node counts below specified node.
</Meaning>
</Parameter>
<Parameter name="Bbox">
<AnyValue/>
<Meaning>Bounding box to filter collections wholly outside the box.</Meaning>
</Parameter>
    <Parameter name="CRS">
    <AllowedValues>
    <Value>EPSG:4326</Value>
    <Value>EPSG:4269</Value>
    <Value>EPSG:32645</Value>
    </AllowedValues>
    <Meaning>Coordinate reference system for BBox</Meaning>
    </Parameter>
<Parameter name="Time">
<AnyValue/>
<Meaning>ISO 8601 time interval</Meaning>
</Parameter>
</Operation>
```

20.3.6 Response

The response from a GetCollectionCount request can be in a format of the clients choosing from a list of formats that the server supports. If the response is of MIME type application/xml then the XSD for the response is specified below.

20.3.6.1 XML Response schema

See **[WAMI XSD]** element “**ExceptionReport”** for requests with errors.

See **[WAMI XSD]** element “**CS_CollectionCount”** for normal application/xml result content.

20.3.6.2 Exceptions

An error in a **GetCollectionCount** request may generate the following **exceptionCodes**:

1. MissingParameterValue
2. InvalidParameterValue
3. OptionNotSupported
4. NoApplicableCode

20.3.6.3 Example GetCollectionCount Response

20.3.6.3.1 Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<CS_CollectionCount xmlns="http://www.pixia.com/wami/v101"
xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.pixia.com/wami wami_1_0_1.xsd
http://www.opengis.net/ows/2.0 owsAll.xsd                    "
  childNodes="3"
  collections="5"
  depth="1"
  edgeDepth="2"
  NID="ROOT"
  root="true"
  totalNodes="25"
  version="1.0.2">
<!-- depth could have been "All" -->
<Description>Detailed statistics about a snapshot of the tree from a given Node ID
(NID)</Description>
</CS_CollectionCount>
```

The request provides information about entries in the entire collection tree.

20.3.6.4 Response Attributes

The response XSD specifies the syntax and constraints for each attribute. Below is the meaning of each attribute and an explanation of the interpretation of its values.

1. The attribute **version** specifies the version of the response.
2. The attribute **root** can be **True** or **False**. If **root=True,** it means the collection count information has been provided starting from the root node. If **root=False,** it means the specified node is either an inner node or leaf node.
3. The attribute **depth** can be **1** or **All** and specifies the value of the **Depth** parameter that was set in the request that spawned this response.
4. The attribute **NID** is the node ID of the collection node being queried. Its value is either the value of the **NID** parameter in the request (if it was set explicitly) or the node ID of the root node.
5. The attribute **childNodes** holds the number of immediate children below this node. The value of **childNodes** shall be an integer ≥ 0. If **childNodes=0**, it means the given node is a leaf node and points to a valid collection. If **childNode** has a value ≥ 1, it means the given node is a root node or inner node.

6. The attribute **totalNodes** holds the total number of nodes below the node specified in NID, *including* the node specified in NID. The value of **totalNodes** shall be an integer > 0. If **Depth=1** in the request, then **totalNodes** shall be equal to **childNodes+1**.
7. The attribute **collections** holds the total number of collections available *below* this node. The value of **collections** is an integer ≥ 0. If **collections=0** and **childNodes=0**, it means the node specified in NID points to a collection. If **collections=0, childNodes>0** in the response and **Depth=1** in the request, it means the node specified in NID has child nodes that may have additional nodes under them that may have collections under them. If **collections=0, childNodes>0** in the response and **Depth=All** in the request, it means there are no collections under the node specified in NID.
8. The attribute **edgeDepth** holds the total number of edges or links from the node specified in NID to the deepest leaf in the sub-tree. The value of **edgeDepth** shall be an integer ≥ 0. A leaf node i.e. a node that points to a collection has **edgeDepth=0**. If **Depth=1** in the request, then **edgeDepth** can be **1** or **0** in the response.

20.4 GetCollections

This request provides the client to retrieve a host of information regarding a collection of WAMI data being served by a server. The data is presented to the client in the form of a tree. A collection has been defined above. The structure and purpose of a collection tree have been discussed above. The client can retrieve from a server serving a collection tree:

1. One or more nodes in the WAMI collection tree
2. A sub-tree of one or more nodes, starting from any node in the WAMI collection tree
3. All, selected or no metadata about each node

A sub-tree of nodes can be isolated by:

1. Node identifier
2. A range time intervals
3. A coverage area of interest

In order for a client to get a collection start from a specific node, the nodes needs to be identified by a node identifier (NID) that is unique to the tree being served. If the node ID is not specified in the service request, then the root node is assumed. To retrieve all nodes below a specific node or the root node, the parameter **Depth** must be set to **All**, explicitly.

The client can optionally choose to receive metadata about a node, or not. If the client chooses to receive metadata about a node, it can receive all metadata or some metadata based on a server-defined scheme.

20.4.1 Warning

When requesting a response to this service, if you are expecting a large XML or do not know whether it could be a large XML or the values in **GetCollectionCount** seem to be significant, requesting a compressed XML is recommended.

20.4.2 Request Parameter Summary

A table of the request parameters for **GetCollections** is provided below.

Table 17: GetCollections request URL parameters

Parameter name	KVP Example	Optionality and multiplicity	Definition, format and use
Service	`Service=CS`	Mandatory, one	The value for this parameter shall be CS.
Request	`Request= GetCollections`	Mandatory, one	The value for this parameter shall be **GetCollections**.
Version	`Version=1.0.2`	Mandatory, one	Version number of service. Shall be implemented by both client and server
Format	`Format=application/xml`	Optional, zero or one	Please refer to **[WAMI OVERVIEW]**
AcceptLanguages	`AcceptLanguages=en-US`	Optional, zero or one	Please refer to **[WAMI OVERVIEW]**
NID	`NID=”unique-1234”`	Optional, zero or one	Specifies the node identifier to retrieve. It shall be a value string that is unique to the tree. If this parameter is not specified or left empty, it means the root node. Shall be implemented by both client and server.
Depth	`Depth=1`	Optional, zero or one	This parameter shall have only three possible values: **0**, **1** and **All**. The default value shall be **Depth=0**. Shall be implemented by both client and server. □ **0**: only send information about this node. □ **1:** send information about this node and its immediate children. □ **All**: send information about this node and all nodes under it.
Metadata	`Metadata=All`	Optional, zero or one	A comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty, it means do not send metadata with node information. If set to **All**, it means send all metadata with node information. At least the value of **All** shall be implemented. A complete list shall be provided as part of the Capabilities response to the **GetCollections** request. Shall be implemented by both client and server.
Bbox	`Bbox=minx,miny,maxx,maxy`	Optional, zero or one	Same as **GetCollectionCount**
CRS	`CRS=epsg:4326`	Optional, zero or one	Same as **GetCollectionCount**
Time	`Time=T1/T2,T3/T4,T5`	Optional, zero or one	Same as **GetCollectionCount**

20.4.3 KVP encoding example

All collection tree nodes, no metadata:

```
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&VERSION=1.0.2&FORMAT=application/xm
l&DEPTH=All
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&VERSION=1.0.2&METADATA=&FORMAT=appl
ication/xml&DEPTH=All
```

All collection tree nodes, all metadata:

```
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&METADATA=All
&VERSION=1.0.2&FORMAT=application/xml&DEPTH=All
```

All collection tree nodes, some metadata elements as made available by server:

```
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&METADATA=Platform,Geospatial"&VERSI
ON=1.0.2&FORMAT=application/xml&DEPTH=All
```

Only the root node, no metadata:

```
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&VERSION=1.0.2&FORMAT=application/xm
l&DEPTH=0
```

Only a specific node and all its metadata:

```
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&VERSION=1.0.2&FORMAT=application/xm
l&DEPTH=0&NID=UniqueID-1234&METADATA=All
```

Only a specific node and all its immediate children and no metadata:

```
http://example.com/path?SERVICE=CS&REQUEST=GetCollections&VERSION=1.0.2&FORMAT=application/xm
l&DEPTH=1& METADATA=&NID=UniqueID-1234
```

20.4.4 XML Encoding schema

See **[WAMI XSD]** element "**CS_GetCollectionsRequest**"

20.4.5 Request Parameter Details

All parameters except **AcceptLanguages** shall be implemented by both client and server.

20.4.5.1 Service

The value for this parameter shall be **CS**.

20.4.5.2 Request

The value for this parameter shall be **GetCollectionCount**.

20.4.5.3 Version

Please refer to **[WAMI OVERVIEW].**

20.4.5.4 Format

Please refer to **[WAMI OVERVIEW].**

20.4.5.5 AcceptLanguages

Please refer to **[WAMI OVERVIEW].**

20.4.5.6 NID

The value of this optional parameter is an identifier string. If not specified in the request or set to empty (…NID=&...), it implies the root node ID. This parameter is otherwise used to identify a specific node in a tree of nodes being served by a Collection Service. Each node has an ID that identifies that node and is unique to the tree of collections being served.

20.4.5.7 Depth

This optional parameter shall have one of three possible values**.** They are **0**, **1** or **All**. If this parameter is not specified in the request or set to empty (…Depth=&), it implies Depth=0. When requesting for information in a specific node, the client may choose to retrieve information about a specific node and:

1. Just the node only, and nothing else. In this case, Depth=0.
2. All its immediate children only. In this case, Depth=1.
3. All nodes under it. In this case Depth=All

20.4.5.7.1 Warning

When **Depth=1** or **Depth=All** and the number of collections returned could be large, it is highly recommended that the response be requested in a compressed format.

20.4.5.8 Metadata

When a client requests for information on a node, the client may choose to optionally ask for metadata associated with the node. Contents of metadata are up to the vendor implementation. The vendor may choose to have zero metadata elements. In that case, if the client requests for metadata, the metadata element in the response shall be empty.

The value to this parameter shall be a comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty (…Metadata=&...), it means do not send metadata associated with the node. If set to **All**, it means send all metadata associated with the node. At least the value of **All** shall be implemented. A complete list shall be provided as part of the Capabilities response to the **GetCollections** request. Besides **All**, implementing additional parameters is completely up to the server implementation. This parameter shall be implemented by both client and server.

If a collection has a lot of metadata per node, the vendor can choose to let the client select one or more sections of the metadata. For example, a leaf node may have metadata on the sensor, the platform holding the sensor, methods used to normalize the data, raw camera model, geospatial information such as the bounding box, spatial reference system, classification, and more. If the server has organized this metadata into sections, each section may be requested independently or in any combination. The **GetCollections** capabilities response provides an example.

20.4.5.8.1 A list of schema defined metadata sections for CS Metadata

- Collection – Aggregate info about the collection
- GeoBox – A 2D bounding representation of the collection imagery
- File – Information about the collection as if it were a file (last-modified, etc.)

- Vendor Defined – Each vendor may specify a list of valid Metadata section names which may be represented as one of the following two XML Element types:
    - Group – Arbitrary grouped name/value pairs (useful when a simple string is sufficient)
    - ExtendedMetadataSection – Named OWS extensible Metadata elements (for use with GML, SensorML, etc.)

20.4.5.9 CRS

Same as GetCollectionCount.

20.4.5.10 BBox

Same as GetCollectionCount.

20.4.5.11 Time

Same as GetCollectionCount.

20.4.6 Example GetCollections Capabilities Fragment Response

```
<Operation name="GetCollections">
<DCP>
<HTTP>
<Get xlink:href="http://example.com/CS?"/>
</HTTP>
</DCP>
<Parameter name="Service">
<AllowedValues>
<Value>CS</Value>
</AllowedValues>
</Parameter>
<Parameter name="Request">
<AllowedValues>
<Value>GetCollections</Value>
</AllowedValues>
</Parameter>
<Parameter name="Version">
<AllowedValues>
<Value>1.0.2</Value>
</AllowedValues>
</Parameter>
<Parameter name="Format">
<AllowedValues>
<Value>application/xml</Value>
<Value>application/x-json</Value>
</AllowedValues>
<DefaultValue>application/xml</DefaultValue>
</Parameter>
<Parameter name="AcceptLanguages">
<AllowedValues>
<Value>en-US</Value>
<Value>en</Value>
</AllowedValues>
<DefaultValue>en-US</DefaultValue>
</Parameter>
<Parameter name="NID">
<AnyValue/>
<Meaning>Client sets one Node ID.
                    The service returns collection count information starting from this node.
                    If not set or empty, implies root node.
                    Shall be implemented by both client and server.
```

```
</Meaning>
</Parameter>
<Parameter name="Metadata">
<AnyValue/>
<Meaning>
                    A comma-separated ordered list of zero or more names of sections of metadata
to be returned.
                    If not set or empty, it means do not send metadata with node information.
                    If set to All, it means send all metadata with node information.
                    At least the value of All shall be implemented.
                    A complete list shall be provided as part of the Capabilities response to the
GetCollections request.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Depth">
<AllowedValues>
<Value>0</Value>
<Value>1</Value>
<Value>All</Value>
</AllowedValues>
<DefaultValue>0</DefaultValue>
<Meaning>Specifies how deep the collection counter shall go down the tree.
                    It has three possible values: 0, 1 or All.
                    If not set or empty, implies Depth=0.
                    Shall be implemented by both client and server.
                    0: only send information about this node.
                    1: Client received number of child nodes under root or specified node.
                    All: Client receives all the node counts below specified node.
</Meaning>
</Parameter>
<Parameter name="Bbox">
<AnyValue/>
<Meaning>Bounding box to filter collections wholly outside the box.</Meaning>
</Parameter>
    <Parameter name="CRS">
    <AllowedValues>
    <Value>EPSG:4326</Value>
    <Value>EPSG:4269</Value>
    <Value>EPSG:32645</Value>
    </AllowedValues>
    <Meaning>Coordinate reference system for BBox</Meaning>
    </Parameter>
<Parameter name="Time">
<AnyValue/>
<Meaning>ISO 8601 time interval</Meaning>
</Parameter>
</Operation>
```

20.4.7 Response

The response from a **GetCollections**request can be in a format of the clients choosing from a list of formats that the server supports. If the response is of MIME type `application/xml` then the XSD for the response is specified below.

20.4.7.1 XML Response schema

See **[WAMI XSD]** element “**ExceptionReport”** for requests with errors.

See **[WAMI XSD]** element “**CS_Collections”** for normal application/xml result content.

20.4.7.2 Exceptions

An error in a **GetCollections** request may generate the following **exceptionCodes**:

1. MissingParameterValue
2. InvalidParameterValue
3. OptionNotSupported
4. NoApplicableCode

20.4.7.3 Example GetCollection Response

A sample request:

```
http://example.com/path?Service=CS&Request=GetCollections&Version=1.0.2&NID=node6&Metadata=Co
llection,GeoBox&Depth=1&Format=application/xml
```

A sample response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CS_Collections xmlns="http://www.pixia.com/wami/v101" xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.pixia.com/wami wami_1_0_1.xsd
http://www.opengis.net/ows/2.0 owsAll.xsd                   http://www.w3.org/1999/xlink
xlinks.xsd">
<!-- depth could have been "All" -->
<Description>Detailed statistics about a snapshot of the tree from a given Node ID
(NID)</Description>
<Parent NID="node5">
<Service name="CS">
<ows:DCP>
<ows:HTTP>
<ows:Get xlink:href="http://example.com/CS?Request=GetCollections&amp;NID=node5"/>
</ows:HTTP>
</ows:DCP>
</Service>
</Parent>
<Node NID="node6" parentNID="node5">
<Description>Node 6 description</Description>
<Node NID="node7" CID="collection7" parentNID="node6">
<Metadata>
<Collection
          startFrame="0"
          endFrame="15"
          frameCount="16"
          startTime="2010-01-01T15:15:15.0Z"
          endTime="2010-01-01T11:15:15.0Z"
          timeSpan="PT1H"
          frameInterval="PT1S"
          />
<GeoBox nativeCRS="EPSG:4236">
<BoundingBox crs="EPSG:4326" minx="1.0" maxx="2.0" miny="1.0" maxy="2.0" resx="50" resy="50"/>
</GeoBox>
</Metadata>
</Node>
</Node>
</CS_Collections>
```

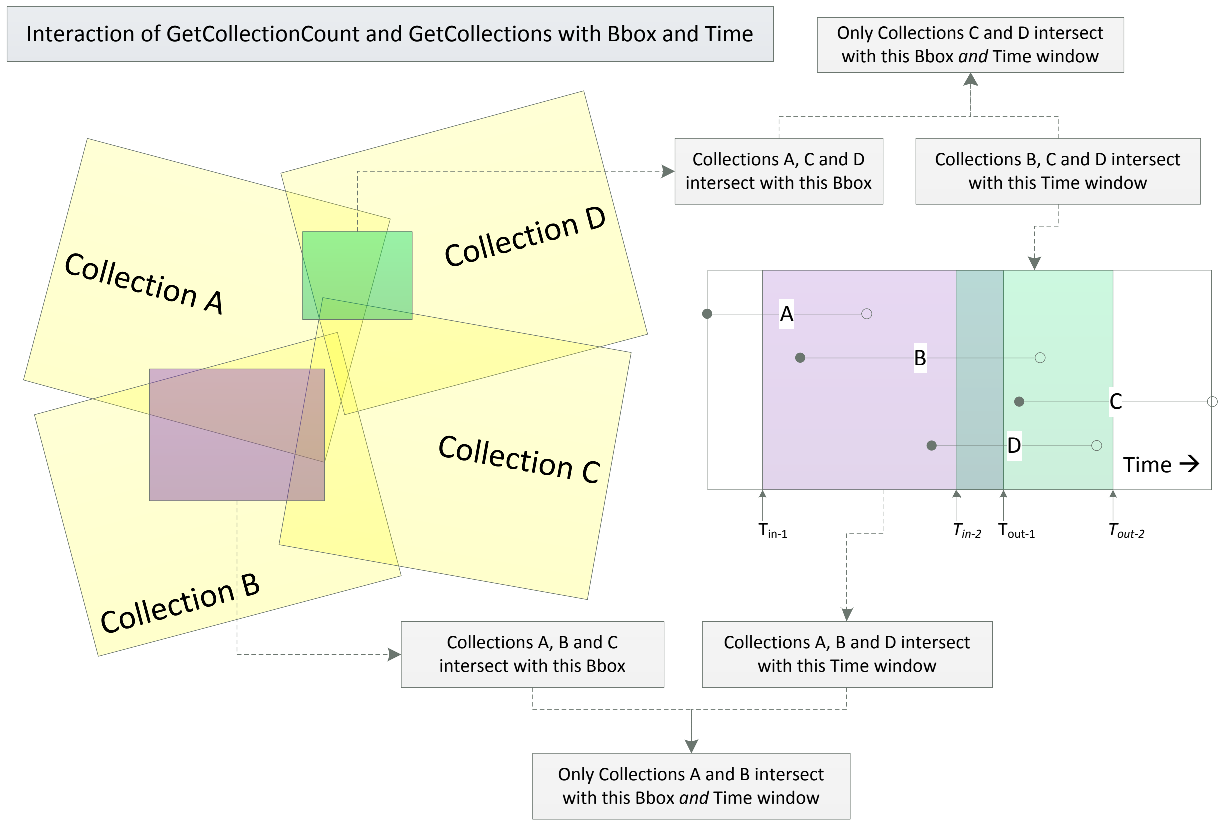

Interaction of GetCollectionCount and GetCollections with Bbox and Time
Only Collections C and D intersect with this Bbox *and* Time window
Collections A, C and D intersect with this Bbox
Collections B, C and D intersect with this Time window
Collection A
Collection D
Collection C
Collection B
A
B
C
D
Time →
T in-1
T in-2
T out-1
T out-2
Collections A, B and C intersect with this Bbox
Collections A, B and D intersect with this Time window
Only Collections A and B intersect with this Bbox *and* Time window

Figure 9: Filter GetCollectionCount and GetCollections by Bbox and Time

20.5 Filtering on Bbox, Time, NID, Depth and Metadata

Time and **Bbox** act as filters in tandem. If **Time** and/or **Bbox** are set in a client request, all Collections wholly outside the bounding box or temporal range are excluded from the response. The rest are included. Quantity and count of data is further controlled by **NID** and **Depth**. The amount of metadata included in the response is controlled by **Metadata**. This applies to both GetCollectionCount and GetCollections.

21 What a Collection Service shall not do

Collection Service is not a search service. It shall not perform arbitrary search operations on collection trees. Metadata based tree filtering and similar non-linear search query operations on a set of collections shall be implemented within a Query Service.

A CS has CRS, Bbox, and Time parameters allowing it to be consistent with and IS, VS requests. By virtue of having these requests, a client can filter or discover data spatiotemporally. Further, it can limit the quantity of data travelling on the network by starting from a specific node and going down tree levels incrementally. This filtering enables data discovery. Search is, as stated above, in the realm of a Query Service.

22 Section 3: Image Service

The name of this service is **IS** or **Image Service**. An Image Service serves a client a requested area of interest (AOI) from a collection of WAMI data and delivers it as an image file of known and supported format. The Image Service also provides metadata about the same AOI. Metadata is reported as formatted text.

To find how collections are being served by a service, or to get information about a collection, use a Collection Service or CS. See **Collection Service Specifications version 1.0.2** for more.

The WAMI specifications document draws heavily from OGC® to define the grammar. Specifically it has references from [OGC WSCS].

23 IS data model

23.1 Time

Various WAMI services require a frame/interval specification. This is generically applied via a 'Time' KVP parameter.

Time interval is set either in absolute collection frame numbers or as an interval of time of acquisition of data. When **Time** is set in terms of an interval of time of acquisition, the format is as specified in the ISO 8601:2004 specification. The representation of time as stated in ISO 8601:2004 is complete. A subset of time interval specification in ISO 8601:2004 is applicable to WAMI specifications as described below.

Time interval values are inclusive. This means the first frame number/time and the last frame number/time are included in the request's reply message.

23.1.1 General interval syntax

The general syntax of values for **Time** is *similar* to **Annex C** and **D** of **OGC** document number **06-042** namely **OGC Web Map Server Implementation Specification version 1.3.0**.

WAMI services treat position and temporal quantities as being equally important. We have extended the possible combinations of values for **Time** as taken from ISO 8601:2004. In this document, the syntax for listing one or more extent values is as per the table below. All intervals are specified in this format. A summary is provided below.

Table 18: Syntax for listing one or more time interval values

KVP Syntax	Meaning	Interpretation
value	A single time value of as frame number or instant of time of acquisition	Get one map at this time value
$value_1$,$value_2$,$value_3$…	A list of explicit multiple values	Get multiple maps, each at the specified time value
min/max/resolution	An interval defined by its lower and upper bounds and its resolution. Lower and upper	Get multiple maps, starting from *min*, up to and including *max*, in

	bounds are time values.	steps of *resolution*.
$min_1/max_1/res_1$, $min_2/max_2/res_2$…	A list of multiple intervals.	Repeat the above process for multiple time intervals.
recurring_interval/ value/ resolution	An interval defined by a recurring time interval designator, interval start time value and its resolution.	Get a total of recurring*_interval* maps, starting from the time *value*, and increment time value by resolution for each consecutive map.
$int_1/value_1/res_1$, $int_2/value_2/res_2$…	A list of multiple intervals.	Repeat the above process for multiple time intervals
recurring_interval/ min/max	An interval defined by a recurring time interval designator, and by its lower and upper bounds. Lower and upper bounds are time values.	Get a total of *recurring_interval* maps, starting from time value *min*, up to and including time value *max*, dispersed evenly from *min* to *max*.
$int_1/min_1/max_1$, $int_2/min_2/max_2$…	A list of multiple intervals.	

If *resolution* is 0 or it is not specified, it means that the data has infinitely fine resolution for the purpose of making requests to the server.

23.1.2 Interpretation of Time Interval

For the purpose of the following discussion, let us designate letters with suffixes for each of the time interval descriptors shown in *Table 18: Syntax for listing one or more time interval values*.

1. V represents *value*.
2. S/E/dT represents *min/max/resolution*.
3. R/V/dT represents *interval/value/resolution*.
4. R/S/E represents *interval/min/max*.

Additionally,

- V_1, V_2, V_3… represent $value_1, value_2, value_3$…
- $S_1/E_1/dT_1, S_2/E_2/dT_2$… represents $min_1/max_1/res_1, min_2/max_2/res_2$…
- $R_1/V_1/dT_1, R_2/V_2/dT_2$ represents $int_1/value_1/res_1, int_2/value_2/res_{2.}$
- $R_1/S_1/E_1, R_2/S_2/E_2$ represents $int_1/min_1/max_1, int_2/min_2/max_2$…
- dT' is the resolution at which the data was captured.
- $R > 0$
- $dT \geq 0$
    - *Special case*: dT can be < 0 only in case of R/V/dT
- If dT is not specified or set to 0, it implies, dT = dT'.

To let the client retrieve multiple maps, starting from a specific instant in time, until a specific instant in time, S, E and dT are necessary and sufficient.

Let us apply the requirement that S, E and dT are necessary and sufficient to retrieve one or more maps to the four possible methods of setting the value of the parameter **Time** mentioned above.

1. V resolves to S/E/dT based on the following rules:
   a. V = S; E and dT are not set
2. S/E/dT is resolved based on the following rules:
   a. S/E/dT resolves to S/E/dT
   b. S/E resolves to S/E/dT'
   c. If S = E; then S/E/dT resolves to V, where V = S
3. R/V/dT resolves to S/E/dT based on the following rules:
   a. If R = 1; R/V/dT resolves to V
   b. If R > 1 and dT > 0; S = V, E = V + (R – 1) x dT
   c. If R > 1 and dT < 0; E = V, S = V + (R – 1) x |dT| and dT = |dT|, applicableif dT is< 0.
   d. R/V resolves to R/V/dT'
   e. V/dT resolves to V
4. R/S/E resolves to S/E/dT based on the following rules:
   a. If R = 1, R/S/E resolves to V, where V = S
   b. If S = E, then R/S/E resolves to V, where V = S
   c. If R > 1, dT = (|E – S|)/(R – 1)
   d. R/S resolves to R/V
   e. S/E resolves to S/E/dT'

The figures below represents cases where you can go forward or backward in time to request maps. S and E are inclusive. The figures also show how to resolve the situation when S and E are known, but if one starts at S and keep going in steps of dT, what happens if one passes E.

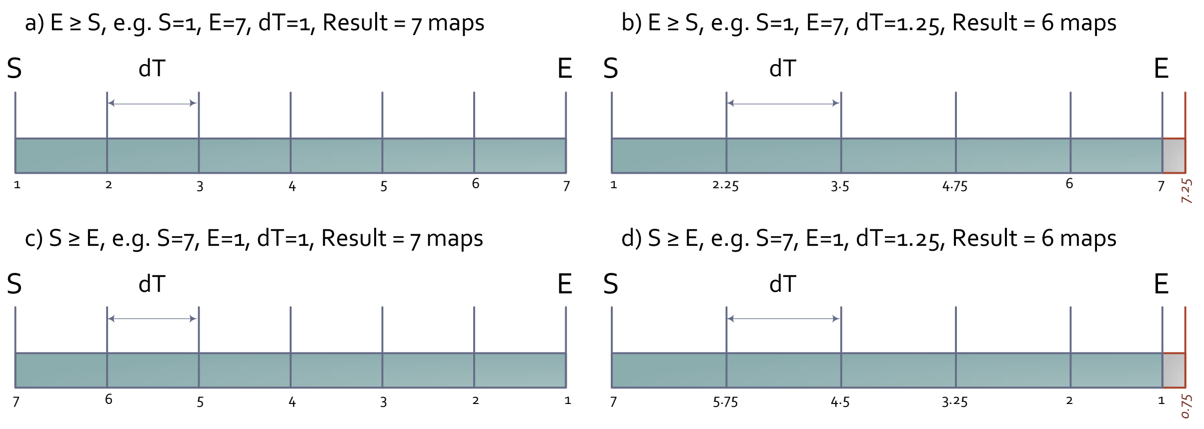

a) E ≥ S, e.g. S=1, E=7, dT=1, Result = 7 maps
S
dT
E
1 2 3 4 5 6 7
b) E ≥ S, e.g. S=1, E=7, dT=1.25, Result = 6 maps
S
dT
E
1 2.25 3.5 4.75 6 7 7.25
c) S ≥ E, e.g. S=7, E=1, dT=1, Result = 7 maps
S
dT
E
7 6 5 4 3 2 1
d) S ≥ E, e.g. S=7, E=1, dT=1.25, Result = 6 maps
S
dT
E
7 5.75 4.5 3.25 2 1 0.75

Figure 10: Time interval interpretation

For an XML POST request, the XSD is also provided **[WAMI XSD]** element "**FrameOrTimeRangeRequestType**". Details of the XML schema are available in **Annex C** of the **OGC** document number **06-042.** The **Annex D** of the **OGC** document number **06-042** provides time format details**.**

23.1.3 Time interval in terms of frames

A collection service shall provide frame range as part of a collection's metadata. The first frame in the collection shall always be numerically smaller than the last frame in the collection. Frame numbers shall be positive integers greater than or equal to zero. Each subsequent frame number shall always be one more than the previous frame number.

If the time interval is specified in terms of absolute frame numbers, it does not follow the ISO 8601:2004 standard.

Let us assume that FRAME is designated to be a frame number, the first or start frame is designated START, the last or end frame is designated as END, the frame step size or frame increment is designated as STEP and the recurring interval is designated as N. FRAME, START and END are positive integers greater than or equal to 0. N is a positive integer greater than 1.

If the time interval is specified in terms of absolute frame numbers then the syntax can be one of:

Table 19: Syntax of specifying Time in terms of absolute frame numbers

Syntax	Interpretation
F[FRAME]	Get one map from frame number FRAME.
F[START]/F[END]/FS[STEP]	Get multiple maps starting from frame number START, up to and including frame number END, incrementing the frame number for each map starting from START in steps of STEP frames. To go backward, set END < START. STEP > 0.
R[N]/F[START]/FS[STEP]	Get a total of N frames starting from frame START, and increasing each subsequent frame by STEP. STEP is an integer. STEP < 0 to go backward. STEP > 0 to go forward. STEP ≠ 0.
R[N]/F[START]/F[END]	Get a total of N frames, starting from frame START and ending at frame END. To go backwards, set END < START. N > 0.

The letter F is the frame number designator. It is followed by START or END frame number. Example: F23

The letters FS are the frame step designator. It is followed by STEP. If not specified, the default if FS1. Example: FS1 or FS-1

The letter R is the recurring interval designator. It is followed by N. Example: R10.

F[START]/F[END] resolves to F[START]/F[END]/FS1

R[N]/F[START] resolves to R[N]/F[START]/FS1

R[N]/F[START]/F[END] resolves to F[START]/F[END]/FS[STEP] where STEP = (END – START) / (R – 1)

In case of R[N]/F[START]/F[END] , if N is such that STEP computes to a fraction, then intermediate frame numbers will resolve to a fractional value. The resulting fractional frame numbers shall be rounded off to the nearest integer frame numbers. The server implementation shall determine how to round it off. For example, if N=6, START=1, END=3 then STEP resolves to (3 – 1) / (6 – 1) or 0.4 and five frame numbers are 1, 1.4, 1.8, 2.2, 2.6, and 3. The server can choose to resolve these frame numbers by *rounding* to the nearest integer, or rounding to the *floor* integer, or rounding to the *ceiling* integer. A *floor* shall result in 1, 1, 2, 2, 3 and 3, a *round* shall result in 1, 1, 1, 2, 2 and 3 and a *ceiling* shall result in 1, 2, 2, 3, 3, and 3.

It is recommended that the server implementations use rounding to the nearest integer.

A client shall query a Collection Service to gain knowledge of the first and last frame numbers of a collection. Some server implementations may choose to start the first frame of all its collections at frame number at 0, others at 1. Some server implementations may have a different frame numbering sequence that may not have frame numbers starting from 0 or 1.

Table 20: Examples of KVP encoding Time as frame numbers

Example KVP Requests	Meaning
`Time=F100`	One map of frame number 11 from a collection
`Time=F100/F2629`	From frame 100 to 2629; expect a total of 2530 maps
`Time=F100/F2629/FS2`	From frame 100 to 2629; in increments of 2; expect a total of 1265 maps
`Time=F200/F100/FS2`	From frame 200 to 100 (backward); in increments of 2; expect a total of 51 maps
`Time=R10/F200/FS2`	From frame 200; in increments of 2; expect a total of 10 maps from frames 200, 202, 204, 206, 208, 210, 212, 214, 216, and 218.
`Time=R10/F200`	Expect 10 maps, from frame 200 up to and including 209.
`Time=R10/F200/FS-2`	From frame 200; in decrements of 2; expect a total of 10 maps from frames number 200, 198, 196, 194, 192, 190, 188, 186, 184, and 182.
`Time=F1,F11,F21,F31,F41`	Expect a total of 5 maps comprising of frame 1, 11, 21, 31 and 41 in that order.
`Time=F1/F11,F21/F31,F34/F44`	Expect a total of 33 frames from 1 to 11, followed by 21 to 31, followed by 34 to 44.
`Time=F1/F11/FS2,F21/F31/FS3,F34/F44/FS2`	Frames 1 to 11 in steps of 2 or frames 1, 3, 5, 7, 9 and 11; frames 21 to 31 in steps of 3 or frames 21, 24, 27, and 30; frames 34 to 44 in steps of 2 or frames 34, 36, 38, 40, 42, and 44; expect a total of 16 maps.

23.1.4 When to set Time as Absolute Frame Numbers

A service implementation may number all collections starting from the same reference frame number such as 0 or 1. If a client requests for multiple collections, we have to assume that for each collection, the exact same frame interval is requested.

Using absolute frame numbers is efficient when dealing with getting a map from one collection at a time. It is also efficient if multiple collections cover adjoining spatial area and have the same time of acquisition for each frame amongst those collections.

Using absolute frame numbers negates the need to finding frame numbers within a collection that is closest to a specific time of acquisition.

A service implementation may number all collections starting from different frame numbers. In that case we recommend not using absolute frame numbers. Instead, use time of acquisition.

To maintain a temporal context, i.e. to get maps for particular time intervals within a set of collections over a certain spatial region, we recommend setting the time interval in terms of time of acquisition.

23.1.5 Time interval in terms of time of acquisition

The basic syntax of for *value*, *min* and *max* in the table 3 above when represented as time is: ***ccyy-mm-dd-Thh:mm:ss.ssssZ***

Where ***cc*** is the century, ***yy*** is the years in the century, ***mm*** is the month number, ***dd*** is the day of the month, ***hh***, mm and ***ss.ssss*** are hours, minutes and seconds. And as seen, seconds has a decimal value to provide sub-second resolution. ***Z*** indicates Zulu time or Coordinated Universal Time (UTC).At least ***ccyy-mm-dd*** must be specified. If time is not set, it is assumed to be **T00:00:00.0Z**.

An **ISO 8601 Recurring Time Interval** is used to indicate the *recurring interval* within the available range. ISO 8601 format for representing the *recurring interval*: Designator R, an integer value N. All representations start with the designator [R], followed, without spaces, by the number of recurrences, if present, followed, without spaces, by a solidus [/], followed, without spaces, by the expression of a time interval. The expression of a time interval is further specified in detail in the specification.

An **ISO 8601 Period** is used to indicate the time *resolution* within the available range. The ISO 8601 format for representing a period of time is used to represent the resolution: Designator P (for Period), number of years Y, months M, days D, time designator T, number of hours H, minutes M, seconds S. Unneeded elements may be omitted. Values must be positive numbers[1].

Table 21: Examples of KVP encoding of Time as ISO 8601:2004

Example KVP Requests	Meaning
`Time=2008-08-20T15:04:26.772Z`	Retrieve one map with the specified time of acquisition (TOA)
`Time=2008-08-20T15:04:26.772Z/2008-08-20T15:29:35.9733Z`	Retrieve all maps between the specified interval(s).
`Time=2008-08-20T15:04:26.772Z/2009-08-20T15:29:35.9733Z/P1M`	Retrieve all maps between the specified interval(s), with every map from the specified 2008 date until the specified 2009 date at a 1 month interval from the prior map.
`Time=2008-08-20T15:04:26.772Z/2009-08-20T15:29:35.9733Z/P3D`	Retrieve all maps between the specified interval(s), with every map from the specified 2008 date until the specified 2009 date at a 3 day interval from the prior map.

[1] *Special case*: There is only one situation, R/V/dT where time interval dT may be a negative value.

[2] Key AOI Entry: A Key AOI entry is analogous to a Key Frame in Key Frame Animation.

[3] A path is defined as a single track of key frames in one collection and a track evaluation method. The AOI (area of interest) is

`Time=2008-08-20T15:04:26.772Z/2009-08-20T15:29:35.9733Z/P1M15D`	Retrieve all maps between the specified interval(s), with every map from the specified 2008 date until the specified 2009 date at a 1 month and 15 day interval from the prior map.
`Time=2008-08-20T15:04:26.772Z/2009-08-20T15:29:35.9733Z/P1M, 2010-08-20T15:04:26.7702Z/2011-08-20T15:29:35.9733Z/P1D`	Retrieve all maps between the 1st interval(s), with every map from the specified 2008 date until the specified 2009 date at a 1 month interval from the prior map. Then retrieve all maps between the 2nd interval(s), with every map from the specified 2010 date until the specified 2011 date at a 1 day interval from the prior map.
`Time=2008-08-20T15:04:26.772Z/2008-08-20T15:29:35.9733Z/PT1.5S`	Retrieve all maps between the specified range, with every map from the specified 2008 date until the specified 2009 date at a 1.5 second interval from the prior map.
`Time=R100/2008-08-20T15:04:26.772Z`	Retrieve 100 consecutive maps starting from the specified 2008 date and time, moving forward in time, at an interval that is the same as the capture rate of each consecutive frame in that collection.
`Time=R100/2008-08-20T15:04:26.772Z/PT1.5S`	Retrieve 100 consecutive maps starting from the specified 2008 date and time, moving forward in time, with each subsequent map at increments of 1.5 seconds.
`Time=R60/2008-08-20T15:04:01Z/2008-08-20T15:04:90Z`	Retrieve 60 consecutive maps starting from the specified date and time of 15:04:01 until 15:04:90, moving forward in time. Based on the request for 60 consecutive maps, the period computes to about PT1.509S.

23.1.6 Mapping Requested to Actual Time of Acquisition

When time of acquisition is set using an ISO 8601 time value, the issue of exactness comes to play. Each collection comprises of a finite number of frames. Each frame has a precise time of acquisition. When performing a request, the client would most likely provide a time of acquisition interval that does not exactly fall on a specific frame number’s time of acquisition.

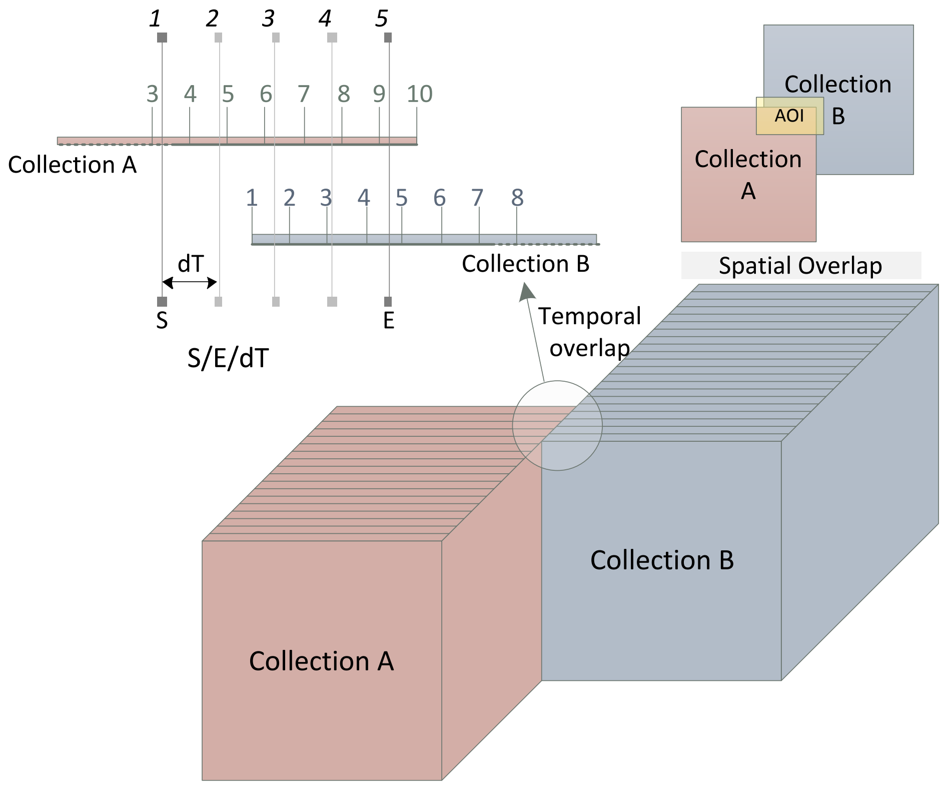


Figure 11: Non-aligned request time atop spatio-temporally overlapping collections

In Figure 11above, we see a request with an AOI bounding box that spatially spans over two collections – A and B. Both of these collections have a temporal overlap as well. However, based on their time of acquisition, it is obvious that the time of acquisition for frames of collection B is not exactly aligned with that of A. The request for multiple frames starts and ends at a time that is not exactly on top of either a collection A frame or a collection B frame. The time interval is such that none of the intermediate frames either fall exactly on any frame within A or on any frame within B.

The server shall decide to implement how it shall resolve non-aligned temporal values. Whatever the selected method, it is necessary to keep it simple and consistent.

Consider the same the example in Figure 11.

The server implementation may choose to select the frame numbers that have a time of acquisition that is closest to but less than or equal to the client's requested time. Frames from collection A would be 3, 4, 6, 7, and 9. Frames from collection B would be 1, 3 and 4. Output frame 1 maps to A3, frame 2 maps to A4, frame 3 maps to B1 + A6, frame 4 maps to B3 + A7, and frame 5 maps to B4 + A9.

The server implementation may choose to select the frame number with the nearest time of acquisition that matches client's requested time. Frames from collection A would be 3, 5, 6, 8, and 9. Frames from collection B would be 2, 3 and 5. Output frame 1 maps to A3, frame 2 maps to A4, frame 3 maps to B2 + A6, frame 4 maps to B3 + A8, and frame 5 maps to B5 + A9.

The table below shows the two possible resolutions for the example inFigure 11.

Table 22: Two possible choices for resolving non-aligned time in a request.

Output Frame	Output Time	Option 1		Option 2	
		Collection A	Collection B	Collection A	Collection B
1	S	A3	-	A3	-
2	S + dT	A4	-	A5	-
3	S + (2 x dT)	A6	B1	A6	B2
4	S + (3 x dT)	A7	B3	A8	B3
5	E	A9	B4	A9	B5

23.2 Client Resources, Bandwidth and Latency

The number of fielded sensors is constantly growing. For surveillance, reconnaissance and analytics applications, a web server may be required to field a large number of collections. Each collection may have thousands to hundreds of thousands of frames. To request for N-number of AOIs from WAMI frames, you would require N requests. The networks may be high bandwidth, but with high latency. They may not be high bandwidth either.

The interface shall provide a way to receive as much data as possible in a single request as well as allow the client to receive data in smaller chunks in parallel. The clients may not have enough computing resources such as memory to handle a large request and therefore may also need requests that ask for small areas of interest as maps.

For low or high bandwidth networks and reasonably low latency, and for clients with limited resources, the basic method to follow could be to:

1. Divide the problem into requests for maps of reasonable sizes – in terms of width and height.
2. Send those requests out in parallel
3. Exploit the received data

For high bandwidth, high latency connections and for thick clients:

1. Know in advance how many frames of AOIs & AOIs you wish to request.
2. If the AOI is the same, use the GetMap request with a ‘Disposition’ parameter of either ‘ordered’ or ‘unordered’.
3. If the AOI may change across frames, generate a track and use the GetPathMap request with valid ‘Disposition’ parameter.
4. If you desire to display metadata along with the imagery, then include the request for metadata in GetMap, GetPathMap and specify a valid ‘Disposition’ parameter.

5. Send a single GetMap/GetPathMap request and receive a stream of still images and (optionally) metadata.

23.3 Path

One path comprises of one or more tracks. One track comprises of two or more key AOIs that fall within one collection's spatial and temporal bounds. For each track, the server generates intermediate AOIs based on these key AOIs. Number of intermediate AOIs can be client controlled or server automated. The server renders a path, i.e. one or more tracks, as a sequence of images of fixed width and height and delivers them as a stream of images. As soon as the first image is available to the server, it is sent, as soon as the second is available, it is sent, and so on until all images are sent.

Each path requires the following inputs:

1. **Width and height**: Dimensions of the output images. All output images from a rendered path shall be of the same size.
2. **Output format**: The format in which the output shall be rendered. In our case it is a sequence of map images in a known file format MIME type.
3. **CRS**: Coordinate reference system of the input bounding boxes of key AOI points as well as for the output sequence of map images.
4. **Options**: Vendor supported special options for formatting the output sequence of map images.
5. **Metadata**: Optional inclusion of per frame metadata, the server provides a list of types of metadata that it supports. If **Metadata** is included, the **Output format** should be set accordingly.
6. **Tracks**: An ordered-list of tracks.

23.3.1 Track

The reason for having at least two Key AOI values is simple. Use **GetMap** for everything else. Each track consists of the following inputs:

1. **Collection ID**: Comma separated ordered list of one or more collection IDs of the collections from which to extract a sequence of frames that follow this track. Spatial and temporal overlap rules apply to a track in the same manner as they do in a **GetMap** request.
2. **Method**: The interpolation method to generate an intermediate AOI corner points from a sequence of Key AOIs. For example, the server may implement linear interpolation, Ease-in/Ease-out curve based interpolation, B-spline based interpolation, or a Catmull-Rom spline based interpolation on the bounding box corner points from one instant in time to the next.

3. A list of **Key AOI entries**: A list of two or more Key AOI entries[2] (this is analogous to specifying key frames in key frame animation). For each Key AOI entry, the required elements are:
   a. **Bounding Box**: The bounding box of the AOI (BBOX=minx, miny, maxx, maxy). The bounding box values are set in the coordinate reference system set in CRS.
   b. **Time of acquisition**: The time of acquisition (TOA) of a specific frame within the collection. This value is specified as a single time value or frame number (Time=<ISO8601 Time Value>; Time=F<number>). The specified bounding box must at least partially intersect the bounds of the WAMI frame at that specific instant in time. If the TOA is set as an ISO8601 time value and the captured frame does not fall exactly at that that time, the server maps it to the closest valid captured frame.
   c. **ΔT**: The time increment. This value is specified as frames or seconds between this key AOI and the next key AOI. The last key AOI shall have a ΔT of zero. If a client sets a key AOI ΔT to be zero and this key AOI entry is not the last key AOI entry, it means the client is requesting the server to determine the number of intermediate AOIs to generate. ΔT is assumed to be in frames if Time of acquisition is set in frames. ΔT is assumed to be in seconds if Time of acquisition is set as an ISO8601 time value.
   d. Server-side automation of intermediate frames is useful for sensors whose acquisition frame rate is not constant. For instance, assume that this collection was acquired at the rate of about 1.8 frames per second:
   e. Consider that a request comprises of two key AOIs.
   f. First key AOI is at time T1.
   g. ΔT is set to 0.
   h. The next key AOI is at time T1 + 10 seconds.
   i. This means a total of 18 frames were captured from T1 to T1 + 10 seconds.
   j. The server returns a sequence of 18 maps images for this section of the track.
   k. **Options**: Additional information required to define each bounding box vertex. This is vendor specific and may be used for supporting description of parameters required for curve-based rendering of intermediate AOI maps. For instance, a spline curve based interpolation method may need additional "knobs" when treating each bounding box vertex as a control point.

23.3.2 Rendering a Track

Rendering of frames is based on the Key Frame Animation technique that has been the basis of moving characters since the advent of traditional and computer animation.

[2] Key AOI Entry: A Key AOI entry is analogous to a Key Frame in Key Frame Animation.

A path has multiple key areas of interest bounding boxes where each AOI bounding box is a set of four points. As we move forward (or backward) in time, the server is interpolating each point between consecutive key AOIs using a pre-defined interpolation method. The result is an intermediate AOI at a given instant in time between the two key AOIs. From a server's perspective, an intermediate frame is one **GetMap** request asking for one AOI.

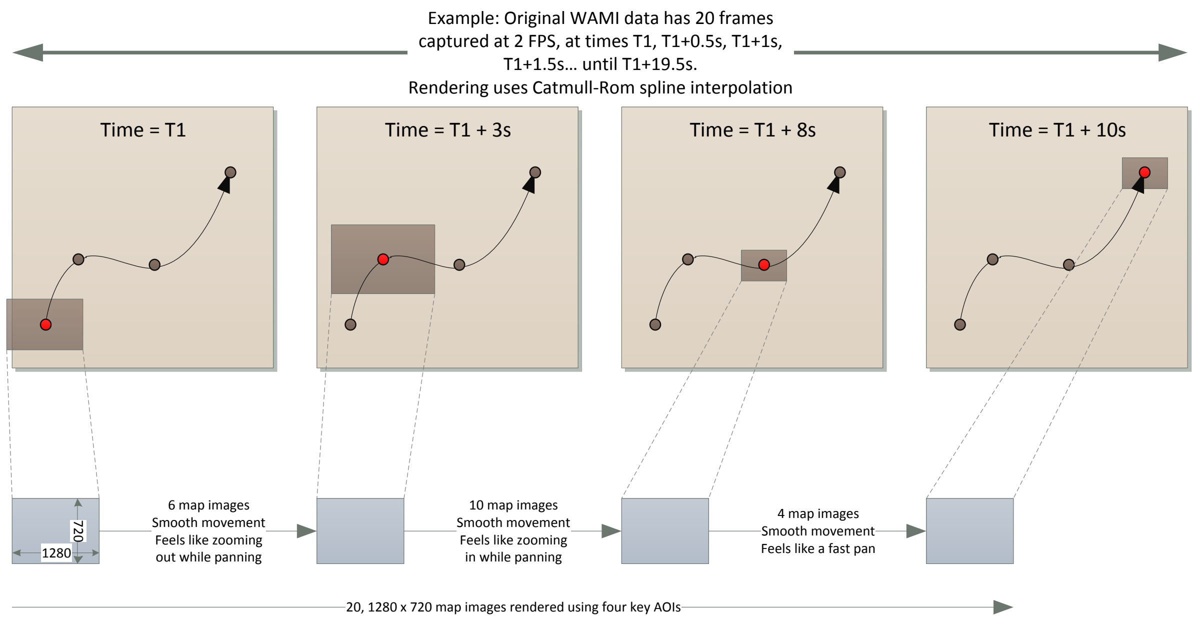

Example: Original WAMI data has 20 frames captured at 2 FPS, at times T1, T1+0.5s, T1+1s, T1+1.5s… until T1+19.5s.
Rendering uses Catmull-Rom spline interpolation
Time = T1
Time = T1 + 3s
Time = T1 + 8s
Time = T1 + 10s
720
1280
6 map images Smooth movement Feels like zooming out while panning
10 map images Smooth movement Feels like zooming in while panning
4 map images Smooth movement Feels like a fast pan
20, 1280 x 720 map images rendered using four key AOIs

Figure 12: Example path based AOI rendering of WAMI data

23.4 GetMap in WMS and Image Service

As you can see, a **GetMap** request in an Image Service (IS) is almost identical to a **GetMap** request in OGC WMS. There are a few differences but they are fundamental.

1. In Image Service, the **CID** parameter replaces the **Layers** parameter. The server implementation may choose to implement the fact that the system only handles one collection per request (CID=a) or multiple collections in a single request specified as a comma separated ordered list of Collection IDs (CID=a,b,c).
2. The **Time** parameter implementation is required in IS but it is optional in WMS. It shall have the idea of specifying time value and time ranges as set by OGC as well as in terms of absolute frame numbers. Both methods are intuitive, provided some preliminary information is available from a Collection Service metadata response for leaf nodes from a collection tree. If the server implements multiple frame requests, it must support a response that enables having multiple separate data elements in a single response message. For instance:
   a. If absolute frame numbers of collections are known, then requesting for frame numbers becomes more intuitive than time ranges based on time of acquisition.

b. If the time of acquisition of each frame is known and reasonably constant, even if the request asks for a frame at an approximate time of acquisition or frame within a temporal range for an given bounding box AOI for a set of collections.
c. The specification could resolve any ambiguity in dereferencing a time value into an absolute frame number within a collection. For example, the specification may state: If for a given collection, the requested time of acquisition T matches frame F_A at time of acquisition T_A and F_B at time of acquisition T_B because $|T-T_A| = |T-T_B|$ then select F_A, if $F_A < F_B$.
d. While this determination is completely arbitrary, it removes any ambiguity.
e. It could be made more complex by selecting F_A or F_B based on the context of the value of T. That is, for time ranges, select F_A, if $F_A < F_B$ for start time of acquisition ambiguity resolution, and select F_B, if $F_A < F_B$ for end time of acquisition ambiguity resolution.
f. It is however recommended that time of acquisition to frame number ambiguity resolution should be kept as simple as possible.

3. A parameter named **Metadata** is introduced to include metadata in the response as part of a multipart/Related response content-type. The Disposition needs to be one of ‘ordered’ or ‘unordered’ for Metadata to be returned as part of a GetMap multipart request. It is generally useful for high latency environments. The implementation of this parameter is optional. The **Metadata** parameter, if supported, shall be constrained to the allowed values of the GetCapabilities response.

24 IS service model

24.1 Summary

A summary of all the requests for an image service is specified in the table below:

Table 23: Image Service request summary

Request	Meaning	Parameters	Optionality
GetCapabilities	Allows the client to retrieve metadata about the capabilities of the IS server implementation	Service, Request, Sections, AcceptVersions, AcceptFormats, AcceptLanguages, UpdateSequence	Mandatory
GetHelp	Allows the client to retrieve help on topics provided by the server	Service, Request, Version, Topic, Format	Optional
GetMap	Retrieves an area of interest from one or more frames from one or more collections of WAMI data as one or more images or images + metadata, in supported standard formats.	Service, Request, Version, Format, Accept Languages, Disposition, CID, CRS, BBox, Width, Height, Styles, Transparent, Bgcolor, Time, Metadata, Options	Mandatory
GetMapInfo	The request is similar to the GetMap request except that instead of images + metadata it retrieves metadata only.	Service, Request, Version, Format, Accept Languages, CID, CRS, BBox, Width, Height,	Optional

		Time, Metadata, Options	
GetPathMap	Retrieves areas of interest (AOIs) from one or more collections of WAMI data as one or more images or images + metadata, in supported standard formats. The information about the collection, time periods and bounding AOIs is supplied as a **path**[3].	Service, Request, Version, Format, Accept Languages , Disposition, CRS , Width, Height, Styles, Transparent, Bgcolor , Metadata, Options, Path	Optional
GetPathMapInfo	The request is similar to the GetPathMap request except that instead of images + metadata it retrieves metadata only.	Service, Request, Version, Format, Accept Languages , CRS , Width, Height, Metadata, Options, Path	Optional

24.2 Request UML (XML/KVP)

Figure 13shows in detail the root XML elements representing all possible **IS** requests. All UML attributes represent XML and KVP request parameters, their data-type, and multiplicity.

There are complex element relationships which differ in the XML and KVP request encoding. See the specific request Parameter section to see KVP encoding for the following fields:

- ViewPort
- Time
- BBox
- Option

The 'Path' element relationship is retained as URL encoded XML for a 'Path' KVP parameter.

24.3 Collection Metadata UML (XML)

See **[WAMI CS]**, version 1.0.2 document

[3] A path is defined as a single track of key frames in one collection and a track evaluation method. The AOI (area of interest) is referenced using a known coordinate reference system.

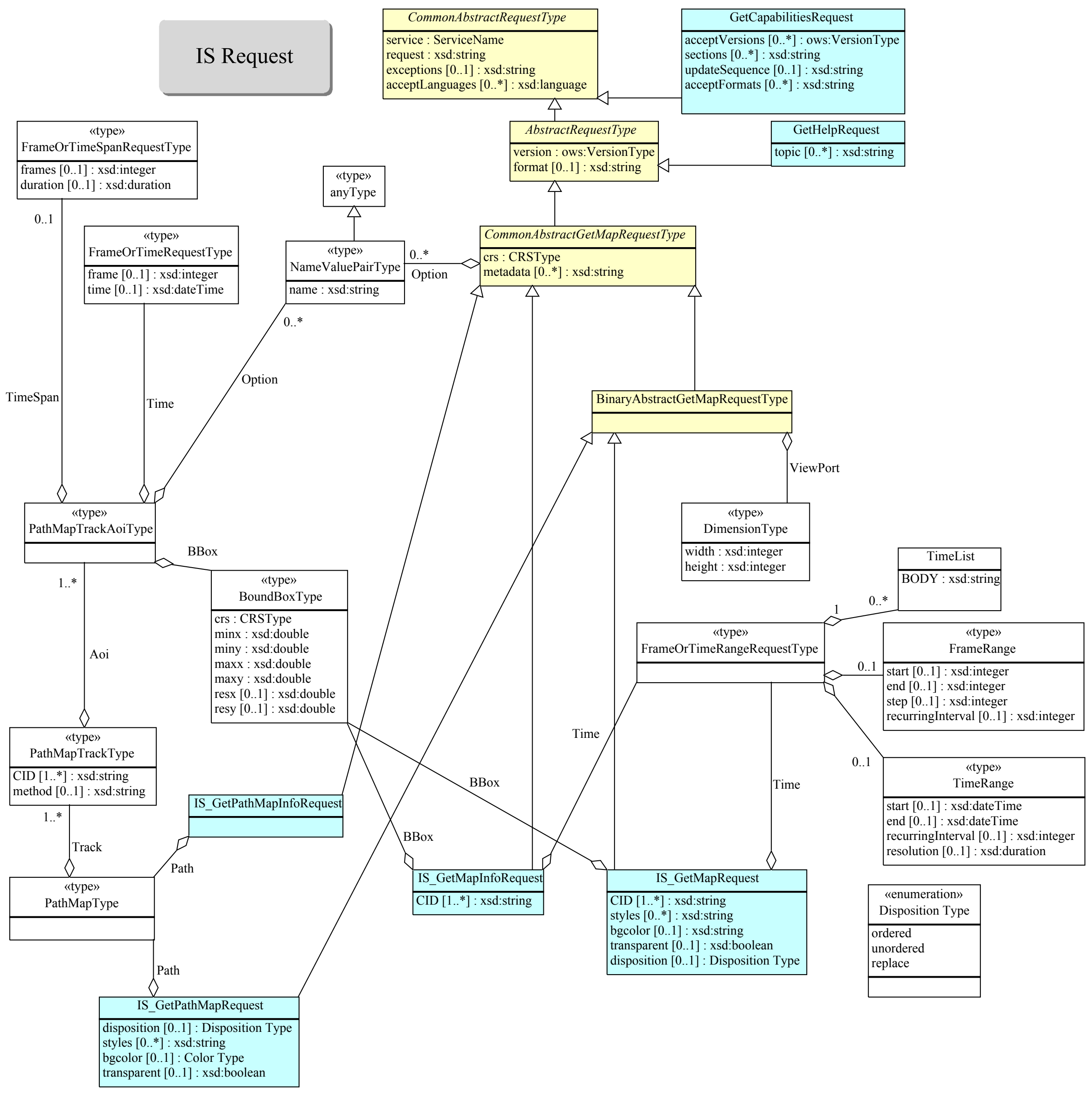

IS Request
CommonAbstractRequestType
service : ServiceName
request : xsd:string
exceptions [0..1] : xsd:string
acceptLanguages [0..*] : xsd:language
GetCapabilitiesRequest
acceptVersions [0..*] : ows:VersionType
sections [0..*] : xsd:string
updateSequence [0..1] : xsd:string
acceptFormats [0..*] : xsd:string
AbstractRequestType
version : ows:VersionType
format [0..1] : xsd:string
GetHelpRequest
topic [0..*] : xsd:string
«type» FrameOrTimeSpanRequestType
frames [0..1] : xsd:integer
duration [0..1] : xsd:duration
«type» anyType
«type» FrameOrTimeRequestType
frame [0..1] : xsd:integer
time [0..1] : xsd:dateTime
«type» NameValuePairType
name : xsd:string
CommonAbstractGetMapRequestType
crs : CRSType
metadata [0..*] : xsd:string
0..1
0..*
Option
0..*
Option
TimeSpan
Time
BinaryAbstractGetMapRequestType
ViewPort
«type» PathMapTrackAoiType
BBox
«type» DimensionType
width : xsd:integer
height : xsd:integer
TimeList
BODY : xsd:string
1..*
«type» BoundBoxType
crs : CRSType
minx : xsd:double
miny : xsd:double
maxx : xsd:double
maxy : xsd:double
resx [0..1] : xsd:double
resy [0..1] : xsd:double
1
0..*
«type» FrameOrTimeRangeRequestType
«type» FrameRange
start [0..1] : xsd:integer
end [0..1] : xsd:integer
step [0..1] : xsd:integer
recurringInterval [0..1] : xsd:integer
0..1
Aoi
Time
«type» PathMapTrackType
CID [1..*] : xsd:string
method [0..1] : xsd:string
BBox
0..1
«type» TimeRange
start [0..1] : xsd:dateTime
end [0..1] : xsd:dateTime
recurringInterval [0..1] : xsd:integer
resolution [0..1] : xsd:duration
Time
1..*
IS_GetPathMapInfoRequest
Track
BBox
Path
«type» PathMapType
IS_GetMapInfoRequest
CID [1..*] : xsd:string
IS_GetMapRequest
CID [1..*] : xsd:string
styles [0..*] : xsd:string
bgcolor [0..1] : xsd:string
transparent [0..1] : xsd:boolean
disposition [0..1] : Disposition Type
«enumeration» Disposition Type
ordered
unordered
replace
Path
IS_GetPathMapRequest
disposition [0..1] : Disposition Type
styles [0..*] : xsd:string
bgcolor [0..1] : Color Type
transparent [0..1] : xsd:boolean

Figure 13: IS Request UML

24.4 Response UML (XML)

Figure 14shows the root XML elements representing **IS** specific response bodies. Not shown are **GetCapabilities** Service Documents and**GetHelp** response bodies.

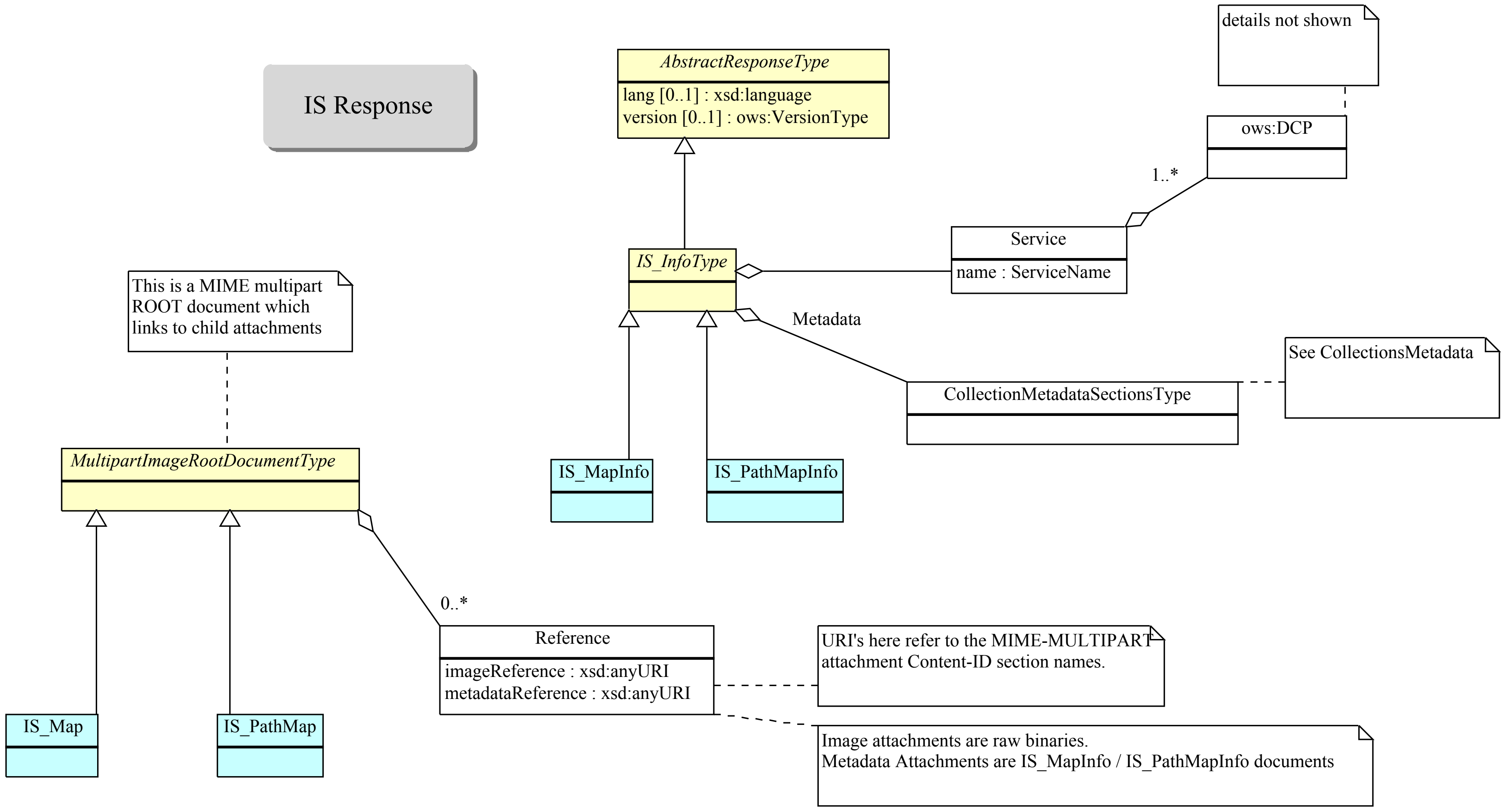

IS Response
AbstractResponseType
lang [0..1] : xsd:language
version [0..1] : ows:VersionType
details not shown
ows:DCP
1..*
Service
name : ServiceName
IS_InfoType
Metadata
CollectionMetadataSectionsType
See CollectionsMetadata
IS_MapInfo
IS_PathMapInfo
This is a MIME multipart ROOT document which links to child attachments
MultipartImageRootDocumentType
0..*
Reference
imageReference : xsd:anyURI
metadataReference : xsd:anyURI
URI's here refer to the MIME-MULTIPART attachment Content-ID section names.
Image attachments are raw binaries.
Metadata Attachments are IS_MapInfo / IS_PathMapInfo documents
IS_Map
IS_PathMap

Figure 14: IS Response UML

25 IS operations

25.1 GetCapabilities

See **[WAMI OVERVIEW]**, version 1.0.2 document

25.2 GetHelp

See **[WAMI OVERVIEW]**, version 1.0.2 document

25.3 GetMap

This request returns one or more images of desired width and height as well as optional metadata from areas of interest within a sequence of one or more WAMI frames potentially composited from one or more collections. It is similar to the WMS GetMap request.

A **GetMap** request is capable of retrieving:

1. One image encoded to the type specified in the 'Format' input parameter. The HTTP response's content-type will match the requested format.
2. An ordered sequence of Images encoded to the type specified in the 'Format' input parameter. The 'Time' input parameter should represent a range that spans 0 or more images. Additionally the 'Disposition' input parameter needs to be set to either 'ordered', 'unordered', 'replace'. It is considered an error to request a sequence without an explicit 'Disposition=ordered', 'Disposition=unordered' or 'Disposition= replace', so as to prevent clients from accidentally requesting multipart data when they can not handle that HTTP response content type.
   a. For example, "Format=image/jpg" will produce either a single image, or an error. "Format=image/jpeg&Disposition=ordered" will produce an HTTP response with "Content-Type: multipart/related" which contains exactly 1 xml response root document and 0 or more related image/jpeg encoded attachments. The root XML document contains an ordered lists of URIs; one for each attached image. Each reference URI will be a fully formed URL that can be used to fetch the image by itself. However, there will be a corresponding multi-part attachment with a Content-ID that matches the URI.
   b. It is assumed that the response-body is streamed lazily as content is rendered. Clients may either batch download the entire response, or more preferably, the client may 'poll' the content-stream for a matching Content-ID attachment which will either be transmitted or lost due to premature connection-closure. If the client specifies 'Disposition=ordered', then the server MUST perform appropriate buffering/reordering such that all attachments match the ordering specified in the root xml document (which in turn must conform to the ordering specified in the request time range). If, howver, a client is able to buffer intermediate requests (possibly by spooling to local disk), the preferred 'Disposition=unordered' can be specified which allows the server to respond in any order (with priority given to the most outstanding frame). Note, that through TCP window-flow-control, it is possible for a thick client to stream an entire long-lived playback with a single GetMap request, rendering each frame based on some playback rate - the next frame will inherently always be ready on the TCP stream so long as neither the client, server, nor intermediate routing points close the underlying HTTP connection.

   c. Lastly, "Format=image/jpeg&Disposition=replace" will produce a "Content-Type: multipart/x-mixed-replace" which consists exclusively of an ordered sequence of images, with no corresponding xml root document. This facilitates a handful of thin-clients that support the multipart/x-mixed-replace content-type, or even thick-clients that don't need to bother with intermediate XML documents.
   d. All non-empty Disposition types are considered Optional, both for client and server.
3. One or more maps as a stream of ordered images corresponding to each map as well as associated metadata corresponding to each map request.
   a. This mode is activated by specifying a non-black, valid 'Metadata' parameter to the 'GetMap' request. This also requires a 'Disposition' parameter with a value of either 'ordered' or 'unordered' ('replace' is not valid with Metadata based requests, and the server should generate an error).
   b. The allowed values for the 'Metadata' parameter are constrained by the 'GetCapabilities' request.
   c. Similar to the multi-image, non meta-data request, the response format shall be of type "Content-Type: multipart/related", which shall be followed by a root XML document which contains an ordered list of URI pairs. One URI to reference an Image, and one URI for the associated Metadata (as would be returned by a 'GetMapInfo' request).
   d. The relative order of Metadata and Image attachments is not specified, BUT when 'Disposition=ordered', Metadata MUST be returned for one frame before the Image attachment of the next frame. When 'Disposition=unordered', then there are no such constraints to the relative positions of associated Image and Metadata attachments, though priority should be given to the most outstanding Metadata document who's associated Image has been transmitted.
   e. A **GetMap** request shall optionally implementation this feature. This is because support for metadata dissemination in an Image Service and in **GetMap** is optional.

25.3.1 Request Parameter Summary

A table of the request parameters for **GetMap** is provided below.

Table 24: GetMap request parameter summary

Parameter name	KVP Example	Optionality and multiplicity	Definition, format and use
Service	`Service=IS`	Mandatory, one	The value shall be IS
Request	`Request=GetMap`	Mandatory, one	The value shall be **GetMap**
Version	`Version=1.0.2`	Mandatory, one	Version number of service. Shall be implemented by both client and server.
Accept Languages	`AcceptLanguages=en-US`	Optional, zero or one	Please refer to **[WAMI OVERVIEW]**
Exceptions	`Exceptions=XML`	Optional, zero or one	Sets the format of the exception.
Format	`Format= image/jpeg`	Mandatory, one	Image output encoding. Note that if multiple images are selected, or if metadata is requested, then the Disposition parameter must also be specified (to avoid accidental unexpected result contents). In doing so, the response Content-Type shall be one of multipart/related or multipart/x-mixed-replace; depending on the value of Disposition.
Disposition	`Disposition=ordered`	Optional, zero or one	This field is required if Metadata is requested or if multiple images are requested (via specifying a Time Range). Valid values are "ordered", "unordered", "replace". Servers that support multipart responses MUST implement "ordered". The "unordered" and "replace" values are optional both for client and server.
CID	`CID=UniqID-1234`	Mandatory, one	A comma-separated ordered list of collection identifiers. Each identifier is unique to a collection. A multi-valued

			CID parameter implies compositing of collections. Shall be implemented by both client and server.
CRS	`CRS=EPSG:4326`	Mandatory, one	The coordinate reference system of the requested map. Shall be implemented by both client and server.
BBox	`BBOX=10.0,10.0,10.1,10.1`	Mandatory, one	The bounding box of the map in the specified CRS. General model: `Bbox=minx,miny,maxx,maxy`. Shall be implemented by both client and server.
Time	`Time=F234/F345/FS1`	Mandatory, one or more	ISO 8601 time-range (in UTC time) or absolute frame-number range and optional step value. Shall be implemented by both client and server. For implementation purposes, if the first character is 'F', then it is a frame or frame-range based request. Otherwise it must comply with ISO 8601 time or time-range. For the frame-range the format is 'F' \d+ ('/F' \d+ ('/FS' \d+)?)?
Width	`Width=512`	Mandatory, one	Width in pixels of the output map pictures. Shall be implemented by both client and server.
Height	`Height=512`	Mandatory, one	Height in pixels of the output map pictures. Shall be implemented by both client and server.
Styles	`Styles=`	Mandatory, one	Comma-separated list of one rendering style per requested collection. An empty list implies default styles. Shall be implemented by both client and server.
Transparent	`Transparent=TRUE`	Optional, zero or one	Background transparency of the maps (Default: Transparent=FALSE). Shall be implemented by both client and server.
Bgcolor	`Bgcolor=0x000000`	Optional, zero or one	Hexadecimal red-green-blue color value for the background color. (Default: Bgcolor=0x000000). Shall be implemented by both client and server.
Metadata	`Metadata=Basic,Sensor,Platform`	Optional, zero or one	A comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty, it means do not send metadata with image data. Shall be optionally implemented by both client and server. When specified, the Disposition parameter needs to be set to allow multipart documents.
Options.CUSTOM	`Options.jpeg_quality= 75&Options.jpeg_yuv=422`	Optional, zero or more	Vendor specific options for this request. Extends the request with one or more vendor defined parameter names. The format of all parameter names will be of the form Options.CUSTOM_NAME = CUSTOM_VALUE. Where each option has its own uniquely specified CUSTOM_NAME (e.g. it is not allowed to specify an Options.CUSTOM_NAME twice, it should instead leverage value encoding such as with a CSV). Shall be optionally implemented by both client and server.

25.3.2 KVP encoding examples

Note – the following URL's are NOT properly URL-escaped – so as to reduce visual artifacts.

From a specific collection, get a WGS84 map, of a specific bounding box from frame number 123 as a single 24-bit PNG, with transparency enabled.

```
http://example.com/path?SERVICE=IS&REQUEST=GetMap&VERSION=1.0.2&FORMAT=image/png;mode=24bit&C
ID=SensorA2345&TIME=F123&BBOX=65.46185,32.64621,65.47011,32.65183&WIDTH=1920&HEIGHT=1080&STYL
ES=&TRANSPARENT=TRUE

Content-Type: image/png
Content-Length: xxx
```

```
[binary-data]
```

From a specific collection, get six WGS84 maps, of a specific bounding box from frame number 120 to frame number 130 in steps of 2, as an XML message comprising of six, 8-bit PNG files, with transparency enabled. Maps from frames 120, 122, 124, 126, 128, 130 will be returned as part of a mime multipart/related document containing a header response xml document that links to subsequent image part attachments.

```
    http://example.com/path?SERVICE=IS&REQUEST=GetMap&VERSION=1.0.2&FORMAT=image/png&CID=SensorA2
    345&TIME=F120/F130/FS2&DISPOSITION=ordered&BBOX=65.46185,32.64621,65.47011,32.65183&WIDTH=192
    0&HEIGHT=1080&STYLES=&TRANSPARENT=TRUE

    Content-Type: multipart/related; boundary="RANDOM12345"; start="root"
    Transfer-Encoding: chunked

    --RANDOM12345
    Content-Type: application/xml; charset=UTF-8
    Content-Id: root

    <?xml version="1.0" encoding="UTF-8"?>
    <wami:IS_Map xmlns:ows="http://www.opengis.net/ows/2.0"
     xmlns:wami="http://www.pixia.com/wami" xmlns:xlink="http://www.w3.org/1999/xlink">
     <wami:ReferenceimageReference="collection_id-image20" metadataReference="collection_id-
metadata20" />
     <wami:ReferenceimageReference="collection_id-image21" metadataReference="collection_id-
metadata21" />
     <wami:ReferenceimageReference="collection_id-image22" metadataReference="collection_id-
metadata22" />
     <wami:Reference imageReference="collection_id-image23" metadataReference="collection_id-
metadata23" />
     <wami:ReferenceimageReference="collection_id-image24" metadataReference="collection_id-
metadata24" />
     <wami:ReferenceimageReference="collection_id-image25" metadataReference="collection_id-
metadata25" />
    </wami:IS_Map>

    --RANDOM12345
    Content-ID: collection_id-metadata20
    Content-type: application/xml; charset=UTF-8
    Content-Length: 1234

    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <wami:IS_MapInfo xmlns:ns2="http://www.opengis.net/ows/2.0"
    xmlns:wami="http://www.pixia.com/wami" xmlns:ns3="http://www.w3.org/1999/xlink">
    <wami:Metadata>
    <wami:GeoBox nativeCRS="EPSG:4326">
    <wami:BoundingBox crs="EPSG:4326" minx="65.71978706512849" miny="31.61895307442826"
    maxx="65.75042093487153" maxy="31.641699925571743" resx="1.8697430263087123E-6"
    resy="1.8511434849840733E-6"/>
    <wami:BoundingBox crs="EPSG:4326" minx="65.71978706512849" miny="31.61895307442826"
    maxx="65.75042093487153" maxy="31.641699925571743" resx="1.8697430263087123E-6"
    resy="1.8511434849840733E-6"/>
    </wami:GeoBox><wami:TOA>2011-01-
    19T03:19:55.0Z</wami:TOA><wami:FrameNum>20</wami:FrameNum><wami:File
    fileName="20110119031955-01002876-VIS.ntf" fileSize="6166906" pixelWidth="16384"
    pixelHeight="12288" bands="1" bitsPerBand="8" bandDataType="u"><wami:WKT>GEOGCS["WGS
    84",DATUM["WGS_1984",SPHEROID["WGS
    84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwi
    ch",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],A
    UTHORITY["EPSG","4326"]]</wami:WKT></wami:File><wami:Group name="Nitf"><wami:Attribute
    name="RAW"></wami:Attribute></wami:Group></wami:Metadata></wami:IS_MapInfo>


    --RANDOM12345
    Content-ID: collection_id-image20
    Content-type: application/xml; charset=UTF-8
```

```
Content-Type: image/png
Content-Length: 100000

[binary-data]

--RANDOM12345
…
```

From a specific collection, get six WGS84 maps and associated metadata, of a specific bounding box from frame number 120 to frame number 130 in steps of 2, as an XML message comprising of six, 24-bit PNG files, interleaved with metadata. Maps from frames 120, 122, 124, 126, 128, 130 will be streamed as an XML reply, comprising of the 24-bit PNG images for a frame followed by the XML for that frame and so on, for the six requested frames. The response will be mime multipart/related encoded with a header response xml document that links to subsequent image-metadata xml documents that then subsequently link to the PNG image attachments themselves.

```
http://example.com/path?SERVICE=IS&REQUEST=GetMap&VERSION=1.0.2&FORMAT=image/png;mode=24bit;o
rder=interleaved&CID=SensorA2345&TIME=F120/F130/FS2&BBOX=65.46185,32.64621,65.47011,32.65183&
STYLES=&WIDTH=1920&HEIGHT=1080&METADATA=All
Content-Type: multipart/related; boundary="RANDOM12345"; start="root"
Transfer-Encoding: chunked

--RANDOM12345
Content-Type: application/xml; charset=UTF-8
Content-ID: root

<?xml version="1.0" encoding="UTF-8"?>
<IS_Map xmlns="http://www.opengis.net/wami/v101" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/wami wami_1_0_1.xsd       ">
<Reference imageReference="image1" metadataReference="metadata1" />
<Reference imageReference="image2" metadataReference="metadata2" />
<Reference imageReference="image3" metadataReference="metadata3" />
</IS_Map>
    …

--RANDOM12345
Content-Id: metadata1
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<IS_MapInfo xmlns="http://www.opengis.net/wami/v101"
xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wami wami_1_0_1.xsd
http://www.opengis.net/ows/2.0 owsAll.xsd                  http://www.w3.org/1999/xlink
xlinks.xsd" version="1.0.2">
<Service name="IS">
<ows:DCP>
<ows:HTTP>
<ows:Get xlink:href="http://example.com/IS?"/>
</ows:HTTP>
</ows:DCP>
</Service>
<Metadata>
<GeoBox nativeCRS="EPSG:4326">
<BoundingBox crs="EPSG:4326" minx="1.0" maxx="2.0" miny="1.0" maxy="2.0" resx="50"
resy="50"/>
</GeoBox>
<TOA>2010-01-01T15:15:15.0Z</TOA>
<FrameNum>5</FrameNum>
<File
 fileName="MISSION_001_2010-01-01_FRAME5.jp2"
 fileSize="1000000"
 createTime="2010-01-01T15:15:15.0Z"
 modifyTime="2010-01-01T15:15:15.0Z"
 pixelWidth="10000"
 pixelHeight="10000"
 fileFormat="image/nitf"
 bands="1"
 bitsPerBand="8"
        bandDataType="unsigned"
>
<GeoTransform xOffset="100" yOffset="100" xScale=".123" yScale=".123" xSkew="0" ySkew="0"/>
<BoundingPolygon crs="EPSG:4326">1.0 1.0 1.0 2.0 2.0 2.0 2.0 1.0 1.0 1.0</BoundingPolygon>
</File>
<Group name="geotiff">
<Attribute name="GCS">4267/NAD27</Attribute>
<Attribute name="Datum">6267/North American Datum 1927</Attribute>
<Attribute name="CornerCoordinates.upperLeft">440720.000,3751320.000</Attribute>
<Attribute name="CornerCoordinates.lowerLeft">440720.000,3720600.000</Attribute>
<Attribute name="CornerCoordinates.upperRight">471440.000,3751320.000</Attribute>
<Attribute name="CornerCoordinates.lowerRight">471440.000,3720600.000</Attribute>
<Attribute name="CornerCoordinates.center">456080.000,3735960.000</Attribute>
</Group>
</Metadata>
</IS_MapInfo>
…
```

From a specific collection, get six WGS84 maps and associated basic metadata, of a specific bounding box from frame number 120 to frame number 130 in steps of 2, as an XML message comprising of six, 8-bit PNG files, with transparency enabled and metadata for each frame. Maps from frames 120, 122, 124, 126, 128, 130 will be returned as consecutive 8-bit PNG files followed by metadata for all six frames as a single XML.

```
http://example.com/path?SERVICE=IS&REQUEST=GetMap&VERSION=1.0.2&FORMAT=image/png&CID=SensorA2
345&TIME=F120%2F130%2FS2&BBOX=65.46185,32.64621,65.47011,32.65183&WIDTH=1920&HEIGHT=1080&STYL
ES=&TRANSPARENT=TRUE&METADATA=Basic
```

From a specific collection, get a WGS84 map, of a specific bounding box from a frame closest to the specified time of acquisition as a single 24-bit PNG, with transparency enabled.

```
http://example.com/path?SERVICE=IS&REQUEST=GetMap&VERSION=1.0.2&FORMAT=image/png%3Bmode=24bit
&CID=SensorA2345&TIME=2008-08-
20T15:04:26.7702Z&BBOX=65.46185,32.64621,65.47011,32.65183&STYLES=&WIDTH=1920&HEIGHT=1080
```

From a specific collection, get one or more WGS84 maps, of a specific bounding box from the frame closest to the start time to the frames closest to the end time of acquisition, in steps of specified seconds. Return a XML message as a sequence of 8-bit PNGs, with transparency enabled as well as metadata in XML.

```
http://example.com/path?SERVICE=IS&REQUEST=GetMap&VERSION=1.0.2&FORMAT=image/png&CID=SensorA2
345&TIME=2008-08-20T15:04:26.7702Z/2008-08-
20T15:29:35.9733Z/PT1.5S&BBOX=65.46185,32.64621,65.47011,32.65183&WIDTH=1920&HEIGHT=1080&STYL
ES=&TRANSPARENT=TRUE&METADATA=All
```

25.3.3 XML Encoding schema

See **[WAMI XSD]** element “**IS_GetMapRequest**”

25.3.4 RequestParameter Details

25.3.4.1 Service

The value for this parameter shall be **IS**

25.3.4.2 Request

The value for this parameter shall be **GetMap**

25.3.4.3 Version

Please refer to **[WAMI OVERVIEW].**

25.3.4.4 AcceptLanguages

Please refer to **[WAMI OVERVIEW].**

25.3.4.5 Format

This is a required parameter. Its value shall specify the format in which the requested data shall be received. The **Format** parameter sets the MIME type of the image map. The **Format** parameter shall only support the MIME type **`image`**.

A **GetMap** request is capable of delivering a response consisting of either one image, more than one image, one image + metadata for that image, or more than one image + metadata for each image. The table below provides the expected behavior of the server for each of these combinations.

It is the responsibility of the server to send back a list of supported formats in its Capabilities response. If the client requests for maps with an incorrect format, the server shall return an **InvalidParameterValue** exception. A detailed response from the server is highly recommended in event of such an error.

Table 25: Single or Multi-part response for multiple map or map+metadata requests.

Image & metadata	Response	Response on Exception
One image, no metadata	Same as the MIME type specified in the **Format** parameter	**Exceptions=IMAGE** embeds error message in the returned response image in the format set in **Format**. An appropriate HTTP status code is in the response header. **Exceptions=XML** returns the exception message in a textual response with the HTTP header's **Content-Type: application/xml**. An appropriate HTTP status code is in the response header. **Exceptions=None** returns an empty response body. An appropriate HTTP status code is in the response header.
One image, with metadata	Response depends on the value of the MIME type in **Format**. If the MIME type in **Format** is capable of	Response in case of an exception is identical to a normal response except the image part of the response.

	holding metadata, then response MIME type is same as **Format** MIME type. If the MIME type in **Format** is not capable of holding metadata, then response MIME type shall be **multipart/related**. The first part shall be of MIME type **application/xml**. It shall contain an XML that informs the client of the contents of the next part. The second part shall be the map itself of MIME type **image** (same as the MIME typeof **Format)**.	The first part shall be of MIME type **application/xml**. It shall contain an XML that informs the client of the contents of the next part. It shall also inform the client that the next part comprises of an exception. The second part shall either be empty, if **Exceptions=None**; or it shall be of MIME type **application/xml** containing the exception message, if **Exceptions=XML**; or it shall be of MIME type that is identical to that of **Format**, if **Exceptions=IMAGE** and the image shall contain the exceptions message embedded in it.
Multiple images, no metadata	Response shall be of MIME type **multipart/related**. The first part shall be a table-of-contents of what follows, with links to the Content IDs of subsequent parts. If the server responds with N maps, there will be N message part pairs that follow. The first part of each pair shall be of MIME type **application/xml**. It shall contain an XML that informs the client of the contents of the next part. The second part of each pair shall be the map itself of MIME type **image** (same as the MIME typeof **Format)**.	Response shall be of MIME type **multipart/related**. The first part shall be a table-of-contents of what follows, with links to the Content IDs of subsequent parts. If the server responds with N maps, there will be N message part pairs that follow. The first part of each pair shall be of MIME type **application/xml**. It shall contain an XML that informs the client of the contents of the next part. It shall also inform the client that the next part comprises of an exception. The second part of each pair shall either be empty, if **Exceptions=None**; or it shall be of MIME type **application/xml** containing the exception message, if **Exceptions=XML**; or it shall be of MIME type that is identical to that of **Format**, if **Exceptions=IMAGE** and the image shall contain the exceptions message embedded in it.
Multiple images with metadata	Same as *multiple images, no metadata,* except that the first part of each pair shall contain, as part of its XML structure, the metadata element with the desired metadata, if metadata is available.	Same as *multiple images, no metadata,* except that the first part of each pair shall contain, as part of its XML structure, the metadata element with the desired metadata, if metadata is available.

The schema definition of the layout of a single and multi-part GetMap response is available later in this document.

25.3.4.5.1 Example Capabilities Response for GetMap's Format Parameter

```
<Parameter name="Format">
<AllowedValues>
<!-- For raw pixels; useful for image to video conversion -->
<Value>image/x-yuv;mode=422</Value>
<Value>image/x-yuv;mode=410</Value>
<Value>image/x-yuv;mode=420</Value>
<Value>image/x-yuv</Value>
<Value>image/x-rawrgb</Value>
<Value>image/x-rawlum</Value>
<!-- For well-known images -->
<Value>image/jpeg</Value>
<Value>image/gif</Value>
<Value>image/jp2</Value>
<Value>image/tiff</Value>
<Value>image/png</Value>
<Value>image/png;mode=24bit</Value>
</AllowedValues>
<Meaning>Set the format of a GetMap response</Meaning>
</Parameter>
```

25.3.4.6 Disposition

This parameter is required if the Metadata parameter is specified or a Time parameter specifies a range of images (and would therefore return multiple images in a multipart format). The server may choose to not implement mime-multipart responses and thus not list any valid values in its Capabilities response. If, however, multipart is supported, it MUST implement at LEAST 'ordered'. When specified, this requires that all images returned from the GetMap call are sequential in time (honoring any sequence-skip deltas). Additionally, if Metadata is requested, the metadata must occur adjacent to the image.

The response format will have a content-type of "multipart/related" and the first attachment will be a root XML document which contains links to both the GetMap image and GetMapInfo for the corresponding image. There will be subsequent attachments in the multipart response that will have Content-ID labels that match the URIs. One for the Image, and if Metadata is requested, one for the Metadata – who's XML document matches that of a GetMapInfo response.

25.3.4.6.1 Example Capabilities Response for GetMap's Disposition Parameter

```
<Parameter name="Disposition">
<!-- These categories are specified by the server implementation -->
<AllowedValues>
<Value>ordered</Value>
<Value>unordered</Value>
<Value>replace</Value>
</AllowedValues>
</Parameter>
```

25.3.4.7 CID

This is a required parameter. The value of this parameter is a comma separated ordered list of collection identifiers, also known as Collection IDs. Each Collection ID identifies a collection that is unique to a collection tree. A Collection Service provides a list of all collections it is currently serving. For each collection, the Collection Service also provides all the services that provide data for that collection.

Each Collection ID identifies the unique ID of the Collection node that contains the desired collection of frames. A Collection Service provides a list of all collections it is currently serving. It also informs which of the collections has an Image Service available for serving up the maps as a flip book.

Example: `CID=CHMKV4S20110203,CHRSM5S20110203`

25.3.4.7.1 Spatial and Temporal Overlap in a GetMap Request

Let us consider the possibility that a client requests for more than one map from more than one collection for a specific time window. In the figure below, a client issues a request for a 16:9 full HD 1920x1080 AOI as shown. The request is issued for collection IDs A, B, C, D and E. The time window ranges from T_{in} to T_{out}. Spatially, all five collections overlap towards the centre. Temporally, all five collections overlap independently at some point in time.

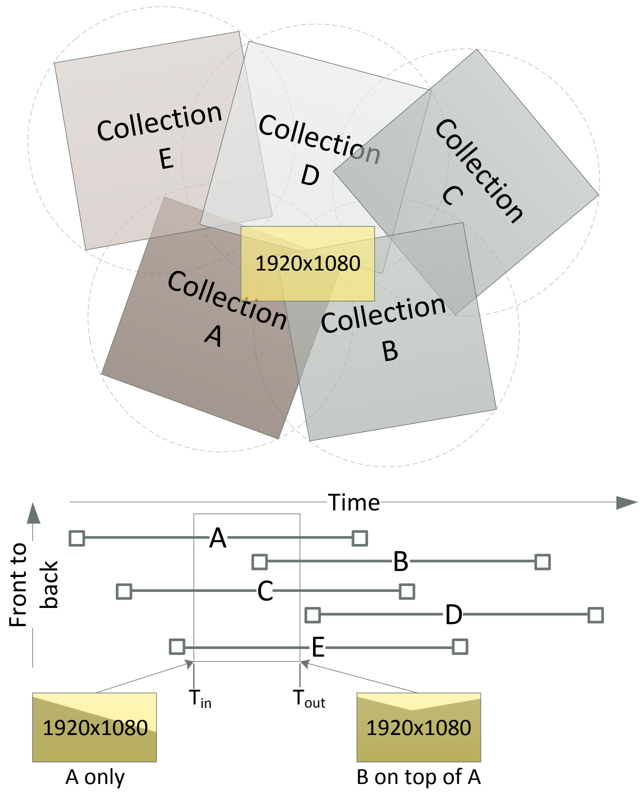


Figure 15: Spatial and temporal overlap in a GetMap request

Spatially, collections A, B and D are overlapping. Temporally, for the given time window, collection D does not intersect the time window. Collection A intersects the time window completely. Collection B starts a little past midway into the time window.

The output maps shall be generated keeping in mind temporal and spatial overlap.

In this example, if the request includes, `CID=A,B,C,D,E&TIME=`T_{in}`/`T_{out}, then the output map images will contain pixels from collection A, from T_{in} until the start of collection B. From the start of collection B until T_{out}, the output map images will contain pixels from intersecting AOIs from collections A and B. Pixels from collection B will get rendered on top of those from collection A.

25.3.4.8 CRS

This parameter indicates the coordinate reference system of the values of parameter **BBox**. It also indicates the coordinate reference system that the maps are expected in. Detailed specification is as defined in **[OGC OWSCS]**. Also see OGC's WMS and WCS specifications. If the source data is not in the specified coordinate reference system, then the image may be re-projected or processed to match the output CRS.

The Capabilities response of a GetMap request shall provide a list of CRSs that are supported by the service implementation.

25.3.4.8.1 Example Capabilities Response for GetMap's CRS Parameter

```
<Parameter name="CRS">
<AllowedValues>
<Value>EPSG:4326</Value>
<Value>EPSG:4269</Value>
<Value>EPSG:32641</Value>
<Value>EPSG:32642</Value>
<Value>EPSG:32643</Value>
<Value>EPSG:32644</Value>
<Value>EPSG:32645</Value>
</AllowedValues>
<Meaning>Coordinate reference system for BBox and map</Meaning>
</Parameter>
```

Example:

```
    CID=CHMKV4S02032011&CRS=EPSG:4326
```

25.3.4.9 BBox

This parameter sets the bounding box of the requested map. This parameter indicates the bounding box in the coordinate reference system set as the value of parameter **CRS**. This bounding box is fit within the output raster pixel area specified as values to parameters **Width** and **Height**. Detailed specification is as defined in **[OGC WSCS]**. Also see OGC's WMS and WCS specifications. If the source data is not in the specified coordinate reference system, then the image may be re-projected or processed to match the output CRS and then mapped to the specified **BBox**.

Example: `CID=CHMKV4S02032011&CRS=EPSG:4326&BBOX=-117.6788847,35.603284,-117.6704312,35.608133`

25.3.4.10 Time

The purpose of the **Time** parameter is to let the client request one or more maps starting from one start point in time until another end point in time. The **Time** parameter also lets a client choose number of maps to retrieve between the start and the end point.

The value for **Time** is a *time interval*.

See Section 23.1 for details.

25.3.4.11 Width

This required parameter specifies the width of the output map in pixels. This number shall be a positive integer greater than zero. A server may choose not to allow a width greater than some known value to avoid bandwidth and server-side resource saturation.

A Collection Service provides the pixel width and height of frames within that collection. The server fits the AOI specified in **Bbox** within a raster of the specified pixel width and height. The reason for the collection service to provide pixel width and height of the frames is to allow for a client to know what metrics the frame possesses. It also allows for the client to select a map with the same pixel size at which the frame was captured. **GetMapInfo**, if implemented, retrieves metadata about a specific frame.

Example: `WIDTH=1920&HEIGHT=1080`

25.3.4.12 Height

This required parameter specifies the height of the output map in pixels. It follows the same constraints as **Width**.

25.3.4.13 Styles

See **OGC** document number **06-042** namely **OGCWeb Map Server Implementation Specification version 1.3.0**.

25.3.4.14 Bgcolor

See **OGC** document number **06-042** namely **OGCWeb Map Server Implementation Specification version 1.3.0**.

25.3.4.15 Transparent

See **OGC** document number **06-042** namely **OGCWeb Map Server Implementation Specification version 1.3.0**. It can be TRUE or FALSE. Default value is FALSE.

25.3.4.16 Metadata

This parameter implementation is optional. A server may choose not to implement this option. In that case, if specified by a client, a server may choose to ignore this option.

For networks that may have high latency, sending a large number of requests may be detrimental to performance. In that case, if a client wants metadata, instead of making one **GetMap** request, in tandem with one **GetMapInfo** request, the client may choose to add **Metadata=[Value]** to a **GetMap** request.

If **Metadata** has a valid value then the **Format** must have a corresponding valid value. The capabilities response for a **GetMap** request for **Format** returns a list of possible formats supported by the service. If **Metadata** has been implemented, the **Format** parameter shall have one or more `application/[something]` MIME types as possible values.

Contents of metadata are up to the vendor implementation. The vendor may choose to have zero metadata elements. In that case, if the client requests for metadata, the metadata element in the response shall be empty.

The value to this parameter shall be a comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty (…Metadata=&...), it means do not send metadata associated with the node. If set to **All**, it means send all metadata associated with the node. At least the value of **All** shall be implemented. A complete list shall be provided as part of the Capabilities response to the **GetMap** request. Besides **All**, implementing additional parameters is completely up to the server implementation. This parameter shall be implemented by both client and server.

If a collection has a lot of metadata per node, the vendor can choose to let the client select one or more sections of the metadata. For example, a leaf node may have metadata on the sensor, the platform holding the sensor, methods used to normalize the data, raw camera model, geospatial information such as the bounding box, spatial reference system, classification, and more. If the server has organized this metadata into sections, each section may be requested independently or in any combination. The **GetMap** capabilities response provides an example.

When requesting metadata with a map image, the response shall always be a multi-part message. See prior section on asking for metadata as part of the GetMap request in a specific format.

25.3.4.16.1 A list of schema defined metadata sections for CS Metadata

1. Collection – Aggregate info about the collection – will generally be redundant between two frames of the same collection.
2. GeoBox – A 2D bounding representation of the Frame imagery
3. TOA – Time of Acquisition. This is a separate section to allow compact queries of all TOAs for a frame range.
4. FrameNum – This is useful to acquire discrete frame numbers for a TOA or TOA range based query.
5. File – Optional Information about File or possibly files that make up the Frame.
6. Vendor Defined – Each vendor may specify a list of valid Metadata section names which may be represented as one of the following two XML Element types:
   a. Group – Arbitrary grouped name/value pairs (useful when a simple string is sufficient)

b. ExtendedMetadataSection – Named OWS extensible Metadata elements (for use with GML, SensorML, etc)

25.3.4.16.2 Example Capabilities Response for GetMap's Metadata Parameter

```
<Parameter name="Metadata">
<AllowedValues>
<Value>All</Value>
<Value>Collection</Value>
<Value>GeoBox</Value>
<Value>TOA</Value>
<Value>FrameNum</Value>
<Value>File</Value>
<Value>SensorInfo</Value>
</AllowedValues>
</Parameter>
```

25.3.4.17 Options

This parameter prefix provides additional server specific sub-options to existing parameters. For example, if the service supports `Format=image/jpeg`, it may choose to provide control over various JPEG parameters such as image quality, JPEG encoding methods, a YUV profile, etc. It should do so by providing unique occurrences of Options. OPTION_NAME=OPTION_VALUE. For example Options.jpeg_YUV=420

25.3.5 Example GetMap Capabilities Response

```
<Operation name="GetMap">
<DCP>
<HTTP>
<Get xlink:href="http://example.com/IS?"/>
</HTTP>
</DCP>
<Parameter name="Service">
<AllowedValues>
<Value>IS</Value>
</AllowedValues>
</Parameter>
<Parameter name="Request">
<AllowedValues>
<Value>GetMap</Value>
</AllowedValues>
</Parameter>
<Parameter name="Version">
<AllowedValues>
<Value>1.0.2</Value>
</AllowedValues>
</Parameter>
<Parameter name="Format">
<AllowedValues>
<Value>application/xml</Value>
<Value>application/x-json</Value>
</AllowedValues>
<DefaultValue>application/xml</DefaultValue>
</Parameter>
<Parameter name="AcceptLanguages">
<AllowedValues>
<Value>en-US</Value>
<Value>en</Value>
</AllowedValues>
<DefaultValue>en-US</DefaultValue>
</Parameter>
```

```
<Parameter name="ID">
<AnyValue/>
<Meaning>A comma-separated ordered list of collection identifiers.
                    Each identifier is unique to a collection.
                    A multi-valued ID parameter implies compositing of collections.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Disposition">
<AllowedValues>
<Value>ordered</Value>
<Value>unordered</Value>
<Value>replace</Value>
</AllowedValues>
<DefaultValue>ordered</DefaultValue>
<Meaning>
            This field is required if Metadata is requested or if multiple images are requested
            (via specifying a Time Range).
            Valid values are “ordered”, “unordered”, “replace”.
            Servers that support multipart responses MUST implement “ordered”.
            The “unordered” and “replace” values are optional both for client and server.
</Meaning>
</Parameter>
     <Parameter name="CRS">
     <AllowedValues>
     <Value>EPSG:4326</Value>
     <Value>EPSG:4269</Value>
     <Value>EPSG:32641</Value>
     <Value>EPSG:32642</Value>
     <Value>EPSG:32643</Value>
     <Value>EPSG:32644</Value>
     <Value>EPSG:32645</Value>
     </AllowedValues>
     <Meaning>Coordinate reference system for BBox </Meaning>
     </Parameter>
<Parameter name="BBox">
<AnyValue/>
<Meaning>
                    The bounding box of the map in the specified CRS.
                    General model: Bbox=minx,miny, maxx,maxy.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Time">
<AnyValue/>
<Meaning>
                    ISO 8601 time-range (in UTC time) or absolute frame-number range and optional
step value.
                    Shall be implemented by both client and server.
                    For implementation purposes, if the first character is ‘F’, then it is a
frame or frame-range based request.
                    Otherwise it must comply with ISO 8601 time or time-range.
                    For the frame-range the format is ‘F’ \d+ ( ‘/F’ \d+ ( ‘/FS’ \d+ )? )?
</Meaning>
</Parameter>
<Parameter name="Width">
<AnyValue/>
<Meaning>
                    Width in pixels of the output map pictures.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Heigth">
<AnyValue/>
<Meaning>
                    Height in pixels of the output map pictures.
                    Shall be implemented by both client and server.
```

```
</Meaning>
</Parameter>
<Parameter name="Styles">
<AnyValue/>
<Meaning>
                    Comma-separated list of one rendering style per requested collection.
                    An empty list implies default styles.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Transparent">
<AllowedValues>
<Value>TRUE</Value>
<Value>FALSE</Value>
</AllowedValues>
<DefaultValue>FALSE</DefaultValue>
<Meaning>
                    Background transparency of the maps (Default: Transparent=FALSE).
                    Shall be implemented by both client and server.
</Meaning>
<DataType>Boolean</DataType>
</Parameter>
<Parameter name="BgColor">
<AnyValue/>
<DefaultValue>000000</DefaultValue>
<Meaning>
                    Hexadecimal red-green-blue color value for the background color.
                    (Default: Bgcolor=0x000000). Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Metadata">
<AllowedValues>
<Value>All</Value>
<Value>Basic</Value>
<Value>Geospatial</Value>
</AllowedValues>
<Meaning>
                    A comma-separated ordered list of zero or more names of sections of metadata
to be returned.
                    If not set or empty, it means do not send metadata with image data.
                    Shall be optionally implemented by both client and server.
                    When specified, the Disposition parameter needs to be set to allow multipart
documents.
</Meaning>
</Parameter>
</Operation>
```

25.3.6 Response

Depending on what and how much was requested, the response MIME type of a **GetMap** request shall change.

Table 26: GetMap response MIME types

Client requests for…	Requested MIME type is…	Server replies with MIME type…
One map	`image/[whatever]`	`image/[whatever]`
More than one map	`image/[whatever]`	`multipart/related` First part has table-of-contents of what follows; Content-Type: application/xml with success failure flag and a few other attributes. For each returned map, a two part response

		message; the first part of Content-Type: application/xml; the second part of Content-Type: image/[whatever] containing the map image.
One map + metadata	`image/[whatever]`	`multipart/related` For the returned map, a two part response message; the first part of Content-Type: application/xml with success failure flag, a few other attributes, and *metadata*; the second part of Content-Type: image/[whatever] containing the map image.
More than one map + metadata per map	`image/[whatever]`	`multipart/related` First part has table-of-contents of what follows; Content-Type: application/xml with success failure flag and a few other attributes. For each returned map, a two part response message; the first part of Content-Type: application/xml with success failure flag, a few other attributes, and *metadata*; the second part of Content-Type: image/[whatever] containing the map image.

25.3.6.1 XML Response schema

See **[WAMI XSD]** element "**ExceptionReport"** for requests with errors.

See **[WAMI XSD]** element "**IS_Map"** for normal application/xml result content.

25.3.6.2 Exceptions

An error in a **GetMap** request may generate the following **exceptionCodes**:

1. MissingParameterValue
2. InvalidParameterValue
3. OptionNotSupported
4. NoApplicableCode

25.3.6.3 Example GetMap Response

A sample request:

```
http://example.com/path?SERVICE=IS&REQUEST=GetMap&VERSION=1.0.2& FORMAT=application/x-png-
xml;mode=24bit&CID=SensorA2345&TIME=F120/F122/FS1&BBOX=65.46185,32.64621,65.47011,32.65183&ST
YLES=&WIDTH=1920&HEIGHT=1080&METADATA=All
```

25.4 GetMapInfo

This request returns metadata on one or more areas of interest within a sequence of one or more WAMI frames from one or more collections.

The **GetMapInfo** request is analogous to WMS **GetFeatureInfo**. However, the grammar seems almost identical to **GetMap**. This request may be used to retrieve metadata about a prior **GetMap** request. It may also be used to get metadata about a frame, independent of a prior request.

Returned information shall be of MIME type `application/xml`.

The GetMapInfo request may optionally retrieve information about a range of Images (If the Time parameter specifies a range). Only a single XML document will be returned.

This request need not be implemented by an IS service implementation vendor. However, it is highly recommended that this request be implemented.

25.4.1 Optional Parameters Constraint

The parameters **CRS**, and **Bbox** are optional. However, if one of these has a valid value, all must have a valid value. Otherwise, if any of these values are not set, the rest are ignored. If any of these values is not valid, it generates an **InvalidParameterValue** exception.

The presence of **CRS** and **Bbox** reflect the fact that the client is asking metadata about a specific bounding box within the frame. A server can choose to just send metadata about the entire frame or send frame metadata as well as additional information regarding the **CRS** and **Bbox** of what was requested. It also means that there is a distinct possibility the client did make or is planning to make a corresponding **GetMap** request.

25.4.2 Request Parameter Summary

A table of the request parameters for **GetMapInfo** is provided below.

Table 27: GetMapInfo request parameter summary

Parameter name	KVP Example	Optionality and multiplicity	Definition, format and use
Service	`Service=IS`	Mandatory, one	The value shall be IS
Request	`Request=GetMapInfo`	Mandatory, one	The value shall be **GetMapInfo**
Version	`Version=1.0.2`	Mandatory, one	Version number of service. Shall be implemented by both client and server.
Accept Languages	`AcceptLanguages=en-US`	Optional, zero or one	Please refer to **[WAMI OVERVIEW]**
Exceptions	`Exceptions=XML`	Optional, zero or one	Sets the format of the exception.
Format	`Format= application/xml`	Optional, zero or one	Output format of the metadata. Shall be implemented by both client and server. Shall at least support **application/xml**. Default value is **application/xml**.
CID	`CID=UniqID-1234`	Mandatory, one	A comma-separated ordered list of collection identifiers. Each identifier is unique to a collection. Shall be implemented by both client and server.
CRS	`CRS=EPSG:4326`	Optional, zero or one	The coordinate reference system of the requested map. Shall be implemented by both client and server. If not specified, the default value is the native CRS of the frame.
BBox	`BBOX=10.0,10.0,10.1,10.1`	Optional, zero	The bounding box of the map in the specified CRS.

		or one	General model: `Bbox=minx,miny,maxx,maxy`. Shall be implemented by both client and server. If not specified, the default value covers the entire frame.
Time	`Time= F234/F345/FS1`	Mandatory, one	Time period is specified either in ISO 8601 time format, or absolute frame numbers. Shall be implemented by both client and server.

Metadata	`Metadata=Basic,Sensor,Platform`
	
	`Metadata=All`
		Optional, zero or one	A comma-separated ordered list of zero or more names of sections of metadata to be returned. At least the value of **All** shall be implemented. If not set or empty, it means **Metadata=All**. Shall be implemented by both client and server.

Options.OPTION_NAME	`Options.jpeg_yuv=420`	Optional, zero or one	Vendor specific options for this request. Extends the request. (Default: Options=). Shall be optionally implemented by both client and server.

25.4.3 KVP encoding examples

From a specific collection, get all metadata from frame number 123 as an XML.

```
http://example.com/path?SERVICE=IS&REQUEST=GetMapInfo&VERSION=1.0.2&FORMAT=application/xml&CI
D=SensorA2345&METADATA=All&TIME=F123
http://example.com/path?SERVICE=IS&REQUEST=GetMapInfo&VERSION=1.0.2&FORMAT=application/xml&CI
D=SensorA2345&METADATA=&TIME=F123
http://example.com/path?SERVICE=IS&REQUEST=GetMapInfo&VERSION=1.0.2&FORMAT=application/xml&CI
D=SensorA2345&TIME=F123
```

From a specific collection, get all metadata about six 1920x1080 WGS84 maps, of a specific bounding box from frame number 120 to frame number 130 in steps of 2, as an XML. Metadata about maps from frames 120, 122, 124, 126, 128, 130 will be returned.

```
http://example.com/path?SERVICE=IS&REQUEST=GetMapInfo&VERSION=1.0.2&FORMAT=application/xml&CI
D=SensorA2345&TIME=F120/F130/FS2&BBOX=65.46185,32.64621,65.47011,32.65183&WIDTH=1920&HEIGHT=1
080
```

From a specific collection, get only the basic metadata about six 1920x1080 WGS84 maps, of a specific bounding box from frame number 120 to frame number 130 in steps of 2, as an XML. Metadata about maps from frames 120, 122, 124, 126, 128, 130 will be returned. This is assuming that the server implementation provides the option to specify **Metadata=Basic** in the service capability response.

```
http://example.com/path?SERVICE=IS&REQUEST=GetMapInfo&VERSION=1.0.2&FORMAT=image/png&CID=Sens
orA2345&TIME=F120/F130/FS2&BBOX=65.46185,32.64621,65.47011,32.65183&WIDTH=1920&HEIGHT=1080&ME
TADATA=Basic
```

25.4.4 XML Encoding schema

See **[WAMI XSD]** element "**IS_GetMapInfoRequest**"

Request Parameter Details

25.4.4.1 Service

The value for this parameter shall be IS

25.4.4.2 Request

The value for this parameter shall be **GetMapInfo**

25.4.4.3 Version

Please refer to **[WAMI OVERVIEW].**

25.4.4.4 AcceptLanguages

Please refer to **[WAMI OVERVIEW].**

25.4.4.5 Format

This is an optional parameter. Its value shall be one of a list of MIME types that shall specify the format in which the requested metadata shall be returned by the service implementation. At least the value of **application/xml** shall be specified. If this parameter is not specified, the value of `Format=application/xml` is the default value. It is recommended that the server implement the ability to send back a compressed XML stream.

If the client requests for metadata but specifies an incorrect format, the server shall return an **InvalidParameterValue** exception. A detailed response from the server is highly recommended in event of such an error.

25.4.4.5.1 Example Capabilities Response for GetMapInfo's Format Parameter

```
<Parameter name="Format">
<AllowedValues>
<!-- As supported by the server -->
<Value>application/xml</Value>
<Value>application/json</Value>
<Value>text/plain</Value>
<Value>text/html</Value>
</AllowedValues>
<Meaning>Set the format of a GetMapInfo response</Meaning>
</Parameter>
```

25.4.4.6 CID

Same as **GetMap**.

Example: `CID=CHMKV4S02032011`

25.4.4.7 CRS

This parameter indicates the coordinate reference system of the values of parameter **BBox**.

The returned metadata may include this information in the response. While no image processing is required, information about a possible image processing operation (had this been a **GetMap** request) may be returned. If not specified, it means the request is asking the server to send metadata on the entire frame.

Otherwise this parameter follows the same constraints as the **CRS** parameter in **GetMap**.

25.4.4.7.1 Example Capabilities Response for GetMapInfo's CRS Parameter

```
<Parameter name="CRS">
<AllowedValues>
```

```
<Value>EPSG:4326</Value>
<Value>EPSG:4269</Value>
<Value>EPSG:32645</Value>
</AllowedValues>
<Meaning>Coordinate reference system for BBox</Meaning>
</Parameter>
```

Example: `CID=CHMKV4S02032011&CRS=EPSG:4326`

25.4.4.8 BBox

This optional parameter sets the bounding box of the map whose metadata is being requested. If set to empty or not set, it means that the client is requesting for metadata on the entire frame. Otherwise this parameter follows the same constraints as the **Bbox** parameter in **GetMap**.

Example: `CID=CHMKV4S02032011&CRS=EPSG:4326&BBOX=-117.6788847,35.603284,-117.6704312,35.608133`

25.4.4.9 Time

This required parameter sets the time interval for the requested metadata. Time interval is set either in absolute collection frame numbers or as an interval of time of acquisition of data, as specified the in ISO 8601:2004 specification for time and time interval.Otherwise this parameter follows the same constraints as the **Time** parameter in **GetMap**.

25.4.4.10 Metadata

This parameter is optional.

The value to this parameter shall be a comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty (…Metadata=&...), it means **Metadata=All**. Besides **All**, implementing additional parameters is completely up to the server implementation. This parameter shall be implemented by both client and server.

Contents of metadata are up to a vendor implementation. The vendor may choose to have zero metadata elements. In that case, if the client requests for metadata, the metadata element in the response shall be empty.

25.4.4.10.1 Example Capabilities Response for GetMapInfo's Metadata Parameter

```
<Parameter name="Metadata">
<!-- These categories are specified by the server implementation -->
<AllowedValues>
<Value>All</Value>
<Value>Basic</Value>
<Value>Geospatial</Value>
<Value>Sensor</Value>
<Value>Platform</Value>
</AllowedValues>
</Parameter>
```

25.4.4.11 Options

This parameter provides additional server specific sub-options to existing parameters. For example, if the service supports `Format=text/plain`, it may choose to provide control over various formats that the server supports for different specific applications. Please refer to **[WAMI OVERVIEW]**, version 1.0.2 for details on **Options**.

25.4.5 Example GetMapInfo Capabilities Fragment Response

```
<Operation name="GetMapInfo">
<DCP>
<HTTP>
<Get xlink:href="http://example.com/IS?"/>
</HTTP>
</DCP>
<Parameter name="Service">
<AllowedValues>
<Value>IS</Value>
</AllowedValues>
</Parameter>
<Parameter name="Request">
<AllowedValues>
<Value>GetMap</Value>
</AllowedValues>
</Parameter>
<Parameter name="Version">
<AllowedValues>
<Value>1.0.2</Value>
</AllowedValues>
</Parameter>
<Parameter name="Format">
<AllowedValues>
<Value>application/xml</Value>
<Value>application/x-json</Value>
</AllowedValues>
<DefaultValue>application/xml</DefaultValue>
</Parameter>
<Parameter name="AcceptLanguages">
<AllowedValues>
<Value>en-US</Value>
<Value>en</Value>
</AllowedValues>
<DefaultValue>en-US</DefaultValue>
</Parameter>
<Parameter name="ID">
<AnyValue/>
<Meaning>A comma-separated ordered list of collection identifiers.
                    Each identifier is unique to a collection.
                    A multi-valued ID parameter implies compositing of collections.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="CRS">
        <AllowedValues>
        <Value>EPSG:4326</Value>
        <Value>EPSG:4269</Value>
        <Value>EPSG:32645</Value>
        </AllowedValues>
        <Meaning>Coordinate reference system for BBox </Meaning>
</Parameter>
<Parameter name="BBox">
<AnyValue/>
<Meaning>
                    The bounding box of the map in the specified CRS.
                    General model: Bbox=minx,miny, maxx,maxy.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Time">
<AnyValue/>
<Meaning>
                    ISO 8601 time-range (in UTC time) or absolute frame-number range and optional
step value.
```

```
                    Shall be implemented by both client and server.
                    For implementation purposes, if the first character is 'F', then it is a
frame or frame-range based request.
                    Otherwise it must comply with ISO 8601 time or time-range.
                    For the frame-range the format is 'F' \d+ ( '/F' \d+ ( '/FS' \d+ )? )?
</Meaning>
</Parameter>
<Parameter name="Metadata">
<AllowedValues>
<Value>All</Value>
<Value>Basic</Value>
<Value>Geospatial</Value>
</AllowedValues>
<Meaning>
                    A comma-separated ordered list of zero or more names of sections of metadata
to be returned.
                    If not set or empty, it means do not send metadata with image data.
                    Shall be optionally implemented by both client and server.
                    When specified, the Disposition parameter needs to be set to allow multipart
documents.
</Meaning>
</Parameter>
</Operation>
```

25.4.6 Response

The response from a **GetMapInfo** request can be in a format of the clients choosing from a list of formats that the server supports. The client uses the Format parameter to set the response format. The response shall at least support the MIME type of **application/xml.**

25.4.6.1 XML Response schema

See **[WAMI XSD]** element "**ExceptionReport"** for requests with errors.

See **[WAMI XSD]** element "**IS_MapInfo"** for normal application/xml result content.

25.4.6.2 Exceptions

An error in a **GetMapInfo** request may generate the following **exceptionCodes**:

1. MissingParameterValue
2. InvalidParameterValue
3. OptionNotSupported
4. NoApplicableCode

25.4.6.3 Example GetMapInfo Response

A sample request:

```
http://example.com/path?SERVICE=IS&REQUEST=GetMapInfo&VERSION=1.0.2&
FORMAT=application/xml&CID=SensorA2345&TIME=F120/F130/FS2&BBOX=65.46185,32.64621,65.47011,32.
65183&WIDTH=1920&HEIGHT=1080&METADATA=All
```

```
<?xml version="1.0" encoding="UTF-8"?>
<IS_MapInfo xmlns="http://www.opengis.net/wami/v101" xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wami wami_1_0_1.xsd
http://www.opengis.net/ows/2.0 owsAll.xsd                  http://www.w3.org/1999/xlink
xlinks.xsd" version="1.0.2">
<Service name="IS">
```

```
<ows:DCP>
<ows:HTTP>
<ows:Get xlink:href="http://example.com/IS?"/>
</ows:HTTP>
</ows:DCP>
</Service>
<Metadata>
<GeoBox nativeCRS="EPSG:4326">
<BoundingBox crs="EPSG:4326" minx="1.0" maxx="2.0" miny="1.0" maxy="2.0" resx="50" resy="50"/>
</GeoBox>
<TOA>2010-01-01T15:15:15.0Z</TOA>
<FrameNum>5</FrameNum>
<File
    fileName="MISSION_001_2010-01-01_FRAME5.jp2"
    fileSize="1000000"
    createTime="2010-01-01T15:15:15.0Z"
    modifyTime="2010-01-01T15:15:15.0Z"
    pixelWidth="10000"
    pixelHeight="10000"
    fileFormat="image/nitf"
    bands="1"
    bitsPerBand="8"
        bandDataType="unsigned"
>
<GeoTransform xOffset="100" yOffset="100" xScale=".123" yScale=".123" xSkew="0" ySkew="0"/>
<BoundingPolygon crs="EPSG:4326">1.0 1.0 1.0 2.0 2.0 2.0 2.0 1.0 1.0 1.0</BoundingPolygon>
</File>
<Group name="geotiff">
<Attribute name="GCS">4267/NAD27</Attribute>
<Attribute name="Datum">6267/North American Datum 1927</Attribute>
<Attribute name="CornerCoordinates.upperLeft">440720.000,3751320.000</Attribute>
<Attribute name="CornerCoordinates.lowerLeft">440720.000,3720600.000</Attribute>
<Attribute name="CornerCoordinates.upperRight">471440.000,3751320.000</Attribute>
<Attribute name="CornerCoordinates.lowerRight">471440.000,3720600.000</Attribute>
<Attribute name="CornerCoordinates.center">456080.000,3735960.000</Attribute>
</Group>
</Metadata>
</IS_MapInfo>
```

25.5 GetPathMap

This request returns a sequence of one or more map image files. For each returned image file, the request also returns metadata about the rendered frame.

It is highly recommended that the reply be streamed, which means, return 1st image as soon as the server completes generating it, the 2nd image as soon as the server completes generating it, and so on. Do not wait for all images to be generated before starting to send each image.

It is necessary that this request be implemented as a GET and a POST request. Due to the fact that data structures are going from the client to the server, a POST is recommended over a GET.

A server may choose not to implement the **GetPathMap** request. If the server has implemented **GetPathMap**, the capabilities response shall reflect so.

25.5.1 Request Parameter Summary

A table of the request parameters for **GetPathMap** is provided below.

Table 28: GetPathMap request parameter summary

Parameter	KVP Example	Optionality and	Definition, format and use

name		multiplicity	
Service	`Service=IS`	Mandatory, one	The value shall be IS
Request	`Request=GetPathMap`	Mandatory, one	The value shall be **GetPathMap**
Version	`Version=1.0.2`	Mandatory, one	Version number of service. Shall be implemented by both client and server.
Accept Languages	`AcceptLanguages=en-US`	Optional, zero or one	Please refer to **[WAMI OVERVIEW]**
Exceptions	`Exceptions=XML`	Optional, zero or one	Sets the format of the exception.
Format	`Format= image/jpeg`	Mandatory, one	Output format of the maps. Shall be implemented by both client and server.
CRS	`CRS=EPSG:4326`	Mandatory, one	The coordinate reference system of the requested maps. Shall be implemented by both client and server.
Width	`Width=1024`	Mandatory, one	Width in pixels of the output map pictures. Shall be implemented by both client and server.
Height	`Height=1024`	Mandatory, one	Height in pixels of the output map pictures. Shall be implemented by both client and server.
Styles	`Styles=`	Mandatory, one	Comma separated list of one rendering style per requested collection. Shall be implemented by both client and server.
Transparent	`Transparent= TRUE`	Optional, zero or one	Background transparency of the maps (Default: Transparent=FALSE). Shall be implemented by both client and server.
Bgcolor	`Bgcolor= 0x000000`	Optional, zero or one	Hexadecimal red-green-blue color value for the background color. (Default: Bgcolor=0x000000). Shall be implemented by both client and server.
Metadata	`Metadata=All`	Optional, zero or one	A comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty, it means do not send metadata with image data. Shall be optionally implemented by both client and server.
Options.OPTION_NAME	`Options.jpeg_quality=75&Options.jpeg_yuv=420`	Optional, zero or one	Vendor specific options for this request. Extends the request. (Default: Options=). Shall be optionally implemented by both client and server.
Path	`Path=`*`URL EncodedXML`*	Mandatory, one	Specifies the tracks to be rendered by the service. Shall be implemented by both client and server.

25.5.2 XML Encoding schema

See **[WAMI XSD]** element “**IS_GetPathMapRequest**”

25.5.3 Request Parameter Details

25.5.3.1 Service

The value for this parameter shall be IS

25.5.3.2 Request

The value for this parameter shall be **GetPathMap**

25.5.3.3 Version

Please refer to **[WAMI OVERVIEW].**

25.5.3.4 AcceptLanguages

Please refer to **[WAMI OVERVIEW].**

25.5.3.5 Format

Specifies the MIME type of the format in which the user wants results. Its constraints are identical to the specification of Format within a **GetMap** request. Namely, it's possible that a non-empty Metadata parameter will force a Multipart response.

25.5.3.6 Width

This required parameter specifies the width of the output map images in pixels. This number shall be a positive integer greater than zero. A server may choose not to allow a width greater than some known value to avoid bandwidth and server-side resource saturation. All map images generated by this request shall be of the same pixel dimensions.

25.5.3.7 Height

This required parameter specifies the height of the output map images in pixels. This number shall be a positive integer greater than zero. A server may choose not to allow a height greater than some known value to avoid bandwidth and server-side resource saturation. All map images generated by this request shall be of the same pixel dimensions.

25.5.3.8 CRS

This required parameter specifies the coordinate reference system (CRS) of the output map images in pixels. This value is also the coordinate reference of the bounding boxes for each key AOI in a track.

25.5.3.9 Styles

This parameter is identical in behavior to Styles specifications for a GetMap request.

25.5.3.10 Transparent

This parameter is identical in behavior to Transparent specifications for a GetMap request.

25.5.3.11 Bgcolor

This parameter is identical in behavior to Bgcolor specifications for a GetMap request.

25.5.3.12 Metadata

This parameter is identical in behavior to Metadata specifications for a GetMap request.

25.5.3.13 Options

This parameter provides additional server specific sub-options to existing parameters. For example, if the service supports `Format=image/jpeg`, it may choose to provide control over various JPEG parameters such as image quality, JPEG encoding methods, a YUV profile, etc. Please refer to **[WAMI OVERVIEW]**, version 1.0.2 for details on **Options**.

25.5.3.14 Path

This is the URL encoded XML document specifying a multi-track path.

See **[WAMI XSD]** element “**PathMapType**” for details.

25.5.4 Example GetPathMap Capabilities Response

```
<Operation name="GetPathMap">
<DCP>
<HTTP>
<Get xlink:href="http://example.com/IS?"/>
</HTTP>
</DCP>
<Parameter name="Service">
<AllowedValues>
<Value>IS</Value>
</AllowedValues>
</Parameter>
<Parameter name="Request">
<AllowedValues>
<Value>GetPathMap</Value>
</AllowedValues>
</Parameter>
<Parameter name="Version">
<AllowedValues>
<Value>1.0.2</Value>
</AllowedValues>
</Parameter>
<Parameter name="Format">
<AllowedValues>
<Value>application/xml</Value>
<Value>application/x-json</Value>
</AllowedValues>
<DefaultValue>application/xml</DefaultValue>
</Parameter>
<Parameter name="AcceptLanguages">
<AllowedValues>
<Value>en-US</Value>
<Value>en</Value>
</AllowedValues>
<DefaultValue>en-US</DefaultValue>
</Parameter>
<Parameter name="ID">
<AnyValue/>
<Meaning>A comma-separated ordered list of collection identifiers.
                    Each identifier is unique to a collection.
                    A multi-valued ID parameter implies compositing of collections.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Disposition">
<AllowedValues>
<Value>ordered</Value>
<Value>unordered</Value>
<Value>replace</Value>
</AllowedValues>
<DefaultValue>ordered</DefaultValue>
<Meaning>
                    This field is required if Metadata is requested or if multiple images are
requested
                    (via specifying a Time Range).
                    Valid values are “ordered”, “unordered”, “replace”.
                    Servers that support multipart responses MUST implement “ordered”.
                    The “unordered” and “replace” values are optional both for client and
server.
```

```
</Meaning>
</Parameter>
<Parameter name="CRS">
        <AllowedValues>
        <Value>EPSG:4326</Value>
        <Value>EPSG:4269</Value>
        <Value>EPSG:32645</Value>
        </AllowedValues>
        <Meaning>Coordinate reference system for BBox </Meaning>
</Parameter>
<Parameter name="BBox">
<AnyValue/>
<Meaning>
                       The bounding box of the map in the specified CRS.
                       General model: Bbox=minx,miny, maxx,maxy.
                       Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Path">
<AnyValue/>
<Meaning>
                       Specifies the tracks to be rendered by the service. Shall be implemented
by both client and server.
</Meaning>
</Parameter>
<Parameter name="Width">
<AnyValue/>
<Meaning>
                       Width in pixels of the output map pictures.
                       Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Heigth">
<AnyValue/>
<Meaning>
                       Height in pixels of the output map pictures.
                       Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Styles">
<AnyValue/>
<Meaning>
                       Comma-separated list of one rendering style per requested collection.
                       An empty list implies default styles.
                       Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Transparent">
<AllowedValues>
<Value>TRUE</Value>
<Value>FALSE</Value>
</AllowedValues>
<DefaultValue>FALSE</DefaultValue>
<Meaning>
                       Background transparency of the maps (Default: Transparent=FALSE).
                       Shall be implemented by both client and server.
</Meaning>
<DataType>Boolean</DataType>
</Parameter>
<Parameter name="BgColor">
<AnyValue/>
<DefaultValue>000000</DefaultValue>
<Meaning>
                       Hexadecimal red-green-blue color value for the background color.
                       (Default: Bgcolor=0x000000). Shall be implemented by both client and
server.
</Meaning>
</Parameter>
<Parameter name="Metadata">
<AllowedValues>
```

```
<Value>All</Value>
<Value>Basic</Value>
<Value>Geospatial</Value>
</AllowedValues>
<Meaning>
                       A comma-separated ordered list of zero or more names of sections of
metadata to be returned.
                       If not set or empty, it means do not send metadata with image data.
                       Shall be optionally implemented by both client and server.
                       When specified, the Disposition parameter needs to be set to allow
multipart documents.
</Meaning>
</Parameter>
</Operation>
```

25.5.5 Response

The response from a **GetPathMap**request bears the exact same format as the GetMap response. There are additional attributes and elements within the XML part for each map that focus on the specified path. Otherwise both are identical in structure.

25.5.5.1 XML Response schema

See **[WAMI XSD]** element "**ExceptionReport"** for requests with errors.

See **[WAMI XSD]** element "**IS_PathMap"** for normal application/xml result content.

25.5.5.2 Exceptions

An error in a **GetPathMap** request may generate the following **exceptionCodes**:

1. MissingParameterValue
2. InvalidParameterValue
3. OptionNotSupported
4. NoApplicableCode

25.6 GetPathMapInfo

This request returns metadata on a sequence of one or more maps that were rendered by the path. The request is identical in input and output to a **GetPathMap** request. However, instead of returning one or more images, it returns one metadata stream comprising of information on those images.

There are a few minor differences in **GetPathMap** and **GetPathMapInfo**.

- The **Metadata** parameter is optional.
  - An empty **Metadata** parameter i.e. Metadata= is the same as Metadata=All.
  - At least Metadata=All must be implemented.
  - The value to this parameter shall be a comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty (…Metadata=&...), it means **Metadata=All**.

  - o Besides **All**, implementing additional parameters is completely up to the server implementation.
  - o This parameter shall be implemented by both client and server.
  - o Contents of metadata are up to a vendor implementation.
  - o The vendor may choose to have zero metadata elements. In that case, if the client requests for metadata, the metadata element in the response shall be empty.
- The **Format** parameter takes a MIME type of text/[whatever]. Support for application/xml is required.
- The response to a **GetPathMapInfo** request is not multi-part.
- **Transparent** and **Bgcolor** parameters are not required.

A server may choose not to implement the **GetPathMapInfo** request. If the server has implemented **GetPathMapInfo**, the capabilities response shall reflect so.

25.6.1 Request Parameter Summary

A table of the request parameters for **GetPathMap** is provided below.

Table 29: GetPathMapInfo request parameter summary

Parameter name	KVP Example	Optionality and multiplicity	Definition, format and use
Service	`Service=IS`	Mandatory, one	The value shall be IS
Request	`Request=GetPathMapInfo`	Mandatory, one	The value shall be **GetPathMap**
Version	`Version=1.0.2`	Mandatory, one	Version number of service. Shall be implemented by both client and server.
Accept Languages	`AcceptLanguages=en-US`	Optional, zero or one	Please refer to **[WAMI OVERVIEW]**
Exceptions	`Exceptions=XML`	Optional, zero or one	Sets the format of the exception.
Format	`Format= applicaton/xml`	Mandatory, one	Output format of the maps. Shall be implemented by both client and server.
CRS	`CRS=EPSG:4326`	Mandatory, one	The coordinate reference system of the requested maps. Shall be implemented by both client and server.
Metadata	`Metadata=All`	Optional, zero or one	A comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty, it means do not send metadata with image data. Shall be optionally implemented by both client and server.
Options.OPTION_NAME	`Options.jpeg_quality=75&Options.jpeg_yuv=420`	Optional, zero or one	Vendor specific options for this request. Extends the request. (Default: Options=). Shall be optionally implemented by both client and server.
Path	`Path=`*URL EncodedXML*	Mandatory, one	Specifies the tracks to be rendered by the service. Shall be implemented by both client and server.

25.6.2 XML Encoding schema

See **[WAMI XSD]** element "**IS_GetPathMapInfoRequest**"

25.6.3 Request Parameter Details

Parameters match those of GetPathMap where required above.

25.6.4 Response

25.6.4.1 XML Response schema

See **[WAMI XSD]** element "**ExceptionReport"** for requests with errors.

See **[WAMI XSD]** element "**IS_PathMapInfo"** for normal application/xml result content.

25.6.4.2 Exceptions

An error in a **GetPathMapInfo**request may generate the following **exceptionCodes**:

1. MissingParameterValue
2. InvalidParameterValue
3. OptionNotSupported
4. NoApplicableCode

26 Supporting Band Combinations and Bit-Depths

Source data can be 1-band, for instance, luminance and IR data, 3-band, for instance, RGB data or have more than 3-bands such as multi-spectral data. While single multi-spectral WAMI sensors may not be currently available, this specification needs to keep them in mind.

Sensors may collect source data at any specific bit-depth. Some of the popular bit-depths at which image file format store pixels for a band are unsigned 8-bit (u8), unsigned or signed 16-bit (s16, u16), unsigned or signed 32-bit (s32, u32), 32-bit floating point (f32), and 64-bit floating point (f64). If data is captured in 10, 11, 12, or 14-bit space, it is generally stored within a 16-bit number for ease of processing. Many formats, such as TIFF, NITF, JPEG2000 support storing higher bit depth than 8-bits per band.

A WAMI service may have support the dissemination of images with higher band count as well as higher bit-depths per band.

As an example, consider a collection C, with 7-bands, from B1 to B7. Assume that each band is s16. A collection service can state that:

- The collection C is being served with B1, B3, and B4 at such and such collection ID.
- The collection C is being served with B1, B2, and B3 at such and such collection ID.
- The collection C is being served with B4, B5, and B6 at such and such collection ID.
- The collection C is being served with B7 at such and such collection ID.

And so on…

Bit-depths can be supported via either the **Options** or the **Format** parameters in **GetMap**. Source bit-depth information can be conveyed via the AOI's metadata. Output image bit-depth can either be part of **Options** parameter, for instance **Options.Jpeg.BitDepth=12** , or using the **Format** parameter, for instance **Format=image/jpeg; bitDepth=12** or **Format=image/tiff; bitDepth=u16** provided of course, **GetCapabilities** returned that as one of the supported **Options** and **Formats**.

The authors recommend that the default **Format=image/jpeg** or **Format=image/tiff** should be browser-friendly and generate 8-bit data.

Conversion from a higher to lower bit-depth is completely up to the vendor implementation. It is also recommended that the output image be visually acceptable because the authors believe that a down-sampled image is used for visualization and the original bit-depth is used for image processing.

Support for band-combinations, higher bit-depths, high-to-low bit-depth transformation, and techniques used in that transformation are completely up to a vendor implementation.

27 Section 4: Video Service

The name of this service is **VS** or **Video Service**. A Video Service serves a client a requested area of interest (AOI) from a collection of WAMI data and delivers it as a video stream or file of a known format. Italso provides metadata about the same AOIas part of the video stream. The video stream delivered is in a standard and known format. At least one video stream must be supported. At least one video stream must be MISB compliant to follow the MISP.

To find out collections are being served by a service, or to get information about a collection, use a Collection Service or CS. See **Collection Service Specifications version 1.0.2** for more.

28 VS data model

28.1 Client Resources, Bandwidth, and Latency

The number of fielded sensors is constantly growing. For surveillance and defense applications, a web server may be required to field a large number of collections. Each collection may have thousands to hundreds of thousands of frames. To request for N-number of AOIs from WAMI frames, you would require N requests. The networks may be high bandwidth, but with high latency. They may not be high bandwidth either.

The interface shall provide a way to receive as much data as possible in a single request as well as allow the client to receive data in smaller chunks in parallel. The clients may not have enough computing resources such as memory to handle a large request and therefore may also need requests that ask for small areas of interest as maps.

For low or high bandwidth networks and reasonably low latency, and for clients with limited resources, the basic method to follow could be to:

1. Divide requests for maps into reasonable sizes – in terms of width and height.
2. Send those requests out in parallel
3. Exploit the received data

For high bandwidth, high latency connections and for thick clients:

1. Know in advance how many frames of AOIs & AOIs you wish to request.
2. If the AOI is the same, use **GetMapVideo**
3. If the AOI may change across frames, generate a track and use **GetPathMapVideo**
4. If you desire to display metadata, include the request for metadata.
5. Send a single **GetMapVideo/GetPathMapVideo** request and receive a video stream with optionally embedded metadata.

28.2 When to use a Video Service and an Image Service

From a given set of collections of WAMI frames, for specific areas of interest over specific time windows, if you want:

1. Still images and optional metadata to play as a flip book, use an image service
2. Just the metadata about those still images, use an image service
3. Streaming video with optional embedded metadata, use a video service

28.3 Path

Description about paths, tracks and rendering them is available in Image Service specifications version 1.0.2.

The description of a **Path** in Image Service specifications focuses on generating a sequence of map images from one or more tracks in a path. A **Path** in a Video Service is identical in structure to an Image Service. The number of images generated when a set to tracks in a path are rendered is exactly the same. The output is not a set of map images. The output is a video stream.

1. **Width and height**: For VS, this specifies the pixel dimensions of the output video stream.
2. **Output format**: For VS, this is a video MIME type. Analogous in function and constraints to Format in GetMapVideo.
3. **CRS**: Same as IS and **GetMapVideo**.
4. **Options**: For VS, these are extended controls to whatever format was set as the Output format. Analogous to Options in **GetMapVideo**.
5. **Metadata**: Analogous to Metadata in **GetMapVideo**.
6. **Dup**: Analogous to Dup in **GetMapVideo**. As frames are rendered or generated for each track, the number of generated map images is identical to a corresponding **GetPathMap** request from an Image Service. However, as the frames are encoded into the video stream, the **Dup** value is used to replicate or skip frames.
7. **Tracks**: Same as IS.

29 VS service model

29.1 Summary

A summary of all the requests for a video service is specified in the table below:

Table 30: Video Service request summary

Request	Meaning	Parameters	Optionality
GetCapabilities	Allows the client to retrieve metadata about the capabilities of the IS server implementation	Service, Request, Sections, AcceptVersions, AcceptFormats, AcceptLanguages, UpdateSequence	Mandatory
GetHelp	Allows the client to retrieve help on topics provided by the server	Service, Request, Version, Topic, Format	Optional

GetMapVideo	Retrieves an area of interest from a sequence of frames from one or more collections of WAMI data as one video stream or video file in a supported standard format.	Service, Request, Version, Format, CID, CRS, BBox, Width, Height, Styles, Transparent, Bgcolor, Time, Dup, Metadata, Options	Mandatory
GetPathMapVideo	Retrieves an area of interest from a sequence of frames from one or more collections of WAMI data as one video stream or video file in a supported standard format. The information about the collection, time periods and bounding AOIs is supplied as a **path**[4].	Service, Request, Version, Format, CRS, Width, Height, Styles, Transparent, Bgcolor, Dup, Metadata, Method, Options, Path	Optional

29.2 Request UML (XML/KVP)

The following shows in teal the root XML elements representing all possible **VS** requests. All UML attributes represent XML and KVP request parameters, their data-type, and multiplicity.

There are complex element relationships which differ in the XML and KVP request encoding. See the specific request Parameter section to see KVP encoding for the following fields:

- ViewPort
- Time
- BBox
- Option

The ‘Path’ element relationship is retained as URL encoded XML for a ‘Path’ KVP parameter.

[4] A path is defined as a single track of key frames in one collection and a track evaluation method. The AOI (area of interest) is referenced using a known coordinate reference system.

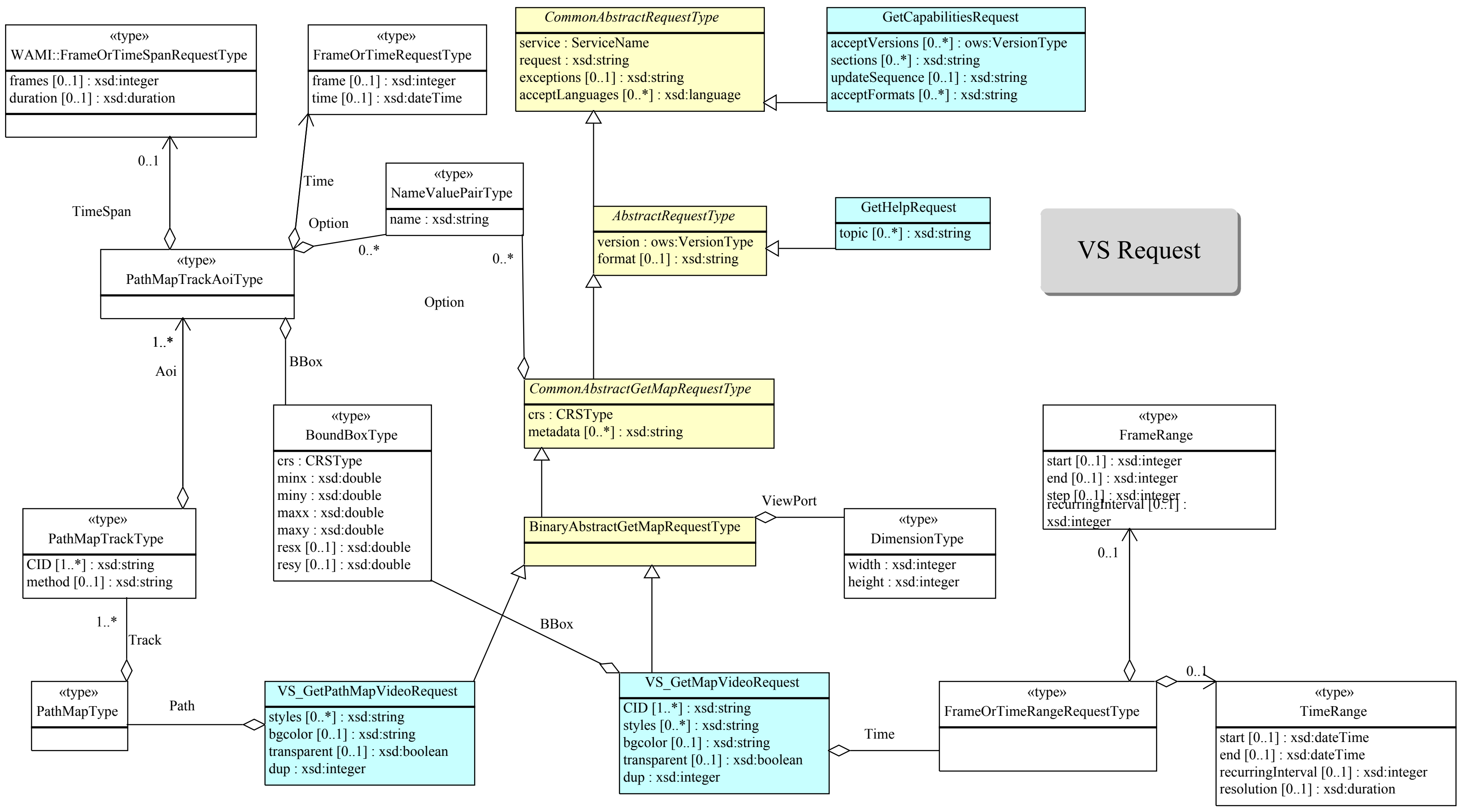

«type»
WAMI::FrameOrTimeSpanRequestType
frames [0..1] : xsd:integer
duration [0..1] : xsd:duration
«type»
FrameOrTimeRequestType
frame [0..1] : xsd:integer
time [0..1] : xsd:dateTime
CommonAbstractRequestType
service : ServiceName
request : xsd:string
exceptions [0..1] : xsd:string
acceptLanguages [0..*] : xsd:language
GetCapabilitiesRequest
acceptVersions [0..*] : ows:VersionType
sections [0..*] : xsd:string
updateSequence [0..1] : xsd:string
acceptFormats [0..*] : xsd:string
0..1
TimeSpan
Time
«type»
NameValuePairType
name : xsd:string
Option
0..*
0..*
AbstractRequestType
version : ows:VersionType
format [0..1] : xsd:string
GetHelpRequest
topic [0..*] : xsd:string
VS Request
«type»
PathMapTrackAoiType
Option
1..*
Aoi
BBox
«type»
BoundBoxType
crs : CRSType
minx : xsd:double
miny : xsd:double
maxx : xsd:double
maxy : xsd:double
resx [0..1] : xsd:double
resy [0..1] : xsd:double
CommonAbstractGetMapRequestType
crs : CRSType
metadata [0..*] : xsd:string
«type»
FrameRange
start [0..1] : xsd:integer
end [0..1] : xsd:integer
step [0..1] : xsd:integer
recurringInterval [0..1] :
xsd:integer
ViewPort
BinaryAbstractGetMapRequestType
«type»
DimensionType
width : xsd:integer
height : xsd:integer
«type»
PathMapTrackType
CID [1..*] : xsd:string
method [0..1] : xsd:string
0..1
1..*
Track
BBox
0..1
«type»
PathMapType
Path
VS_GetPathMapVideoRequest
styles [0..*] : xsd:string
bgcolor [0..1] : xsd:string
transparent [0..1] : xsd:boolean
dup : xsd:integer
VS_GetMapVideoRequest
CID [1..*] : xsd:string
styles [0..*] : xsd:string
bgcolor [0..1] : xsd:string
transparent [0..1] : xsd:boolean
dup : xsd:integer
Time
«type»
FrameOrTimeRangeRequestType
«type»
TimeRange
start [0..1] : xsd:dateTime
end [0..1] : xsd:dateTime
recurringInterval [0..1] : xsd:integer
resolution [0..1] : xsd:duration

Figure 16: VS Request UML

30 VS operations

30.1 GetCapabilities

See **[WAMI OVERVIEW]**, version 1.0.2 document

30.2 GetHelp

See **[WAMI OVERVIEW]**, version 1.0.2 document

30.3 GetMapVideo

This request returns one video stream of desired width and height as well as optional metadata from areas of interest within a sequence of one or more WAMI frames from one or more collections. It may seem familiar. It is similar to the **GetMapVideo** request in an **Image Service** as well as **WMS**.

A **GetMapVideo** request is capable of retrieving:

1. One or more than one map as a stream of video comprising of frames corresponding to each map. A **GetMapVideo** request shall implement this facility.
2. One or more than one map as a stream of video comprising of frames corresponding to each map as well as associated metadata corresponding to each map request embedded within the stream of video. A **GetMapVideo** request may implement this facility.

30.3.1 Request Parameter Summary

A table of the request parameters for **GetMapVideo** is provided below.

Table 31: GetMapVideo request parameter summary

Parameter name	KVP Example	Optionality and multiplicity	Definition, format and use
Service	`Service=VS`	Mandatory, one	The value shall be VS
Request	`Request=GetMapVideo`	Mandatory, one	The value shall be **GetMapVideo**
Version	`Version=1.0.2`	Mandatory, one	Version number of service. Shall be implemented by both client and server.
Accept Languages	`AcceptLanguages=en-US`	Optional, zero or one	Please refer to **[WAMI OVERVIEW]**
Exceptions	`Exceptions=XML`	Optional, zero or one	Sets the format of the exception.
Format	`Format=video/mpeg2`	Mandatory, one	Output format of the video stream as a valid MIME type. Shall be implemented by both client and server.
CID	`CID=UniqID-1234`	Mandatory, one	A comma-separated ordered list of collection identifiers. Each identifier is unique to a collection. Shall be implemented by both client and server.
CRS	`CRS=EPSG:4326`	Mandatory, one	The coordinate reference system of the requested map. Shall be implemented by both client and server.
BBox	`BBOX=10.0,10.0,10.1,10.1`	Mandatory, one	The bounding box of the map in the specified CRS. General model: `Bbox=minx,miny, maxx,maxy`. Shall be implemented by both client and server.
Time	`Time=F234/F345/FS1`	Mandatory, one	Range of time specified either as UTC time, UTC time range or absolute frame number or absolute frame number range. Shall be implemented by both client and server.
Dup	`Dup=1`	Optional, zero or one	Duplicate frames. It specifies the number of times each frame within the time range shall be duplicated. Shall be implemented by both client and server.
Width	`Width=1280`	Mandatory, one	Width in pixels of the output map pictures. Shall be implemented by both client and server.

Height	`Height=720`	Mandatory, one	Height in pixels of the output map pictures. Shall be implemented by both client and server.
Styles	`Styles=`	Mandatory, one	Comma-separated list of one rendering style per requested collection. An empty list implies default styles. Shall be implemented by both client and server.
Transparent	`Transparent=TRUE`	Optional, zero or one	Background transparency of the maps (Default: Transparent=FALSE). Shall be implemented by both client and server.
Bgcolor	`Bgcolor=0x000000`	Optional, zero or one	Hexadecimal red-green-blue color value for the background color. (Default: Bgcolor=0x000000). Shall be implemented by both client and server.
Metadata	`Metadata=Basic,Sensor,Platform`	Optional, zero or one	A comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty, it means do not send metadata with image data. Shall be optionally implemented by both client and server.
Options	`Options.mpeg2.codec=h264& Options.mpeg2.stream=transport& Options.mpeg2.kbps=3000& Options.mpeg2.fps=29.97`	Optional, zero or one	Vendor specific options for this request. Extends the request. (Default: Options=). Shall be optionally implemented by both client and server.

30.3.2 KVP encoding examples

From the collection named SENSORA2345, get a video of a sequence of WGS84 maps of a specific bounding box AOI. Generate 1920x1080 maps starting from frame number 120 until frame number 930, in steps of 2 frames. In other words, get the specified AOI from frames 120, 122, 124… until 930. DUP=1 indicates that repeat each frame once to get a resulting frame sequence of 120, 120, 122, 122, 124, 124… 930,930. Take this sequence of maps and generate a 1080P, MPEG2/H.264 transport video stream at 3 Mbps, 30 fps. Do not include embedded metadata.

```
http://example.com/path?SERVICE=VS&REQUEST=GetMapVideo&VERSION=1.0.2&FORMAT=video/mpeg&CID=Se
nsorA2345&TIME=F120/F930/FS2&DUP=1&BBOX=65.46185,32.64621,65.47011,32.65183&WIDTH=1920&HEIGHT
=1080&STYLES=&OPTIONS=mpeg.type%3Dmpeg2%26mpeg2.codec%3Dh264%26mpeg2.stream%3Dtransport%26mpe
g2.kbps%3D3000%26mpeg2.fps%3D30.00
```

Do the exact same thing as the first request, except the sequence plays backwards because of the value set in the TIME parameter. Do not include embedded metadata.

```
http://example.com/path?SERVICE=VS&REQUEST=GetMapVideo&VERSION=1.0.2&FORMAT=video/mpeg&CID=Se
nsorA2345&TIME=F930/F120/FS2&DUP=1&BBOX=65.46185,32.64621,65.47011,32.65183&WIDTH=1920&HEIGHT
=1080&STYLES=&OPTIONS=mpeg.type%3Dmpeg2%26mpeg2.codec%3Dh264%26mpeg2.stream%3Dtransport%26mpe
g2.kbps%3D3000%26mpeg2.fps%3D30.00
```

Do the exact same thing as the first request, and include embedded metadata that follows MISP 6.1.

```
http://example.com/path?SERVICE=VS&REQUEST=GetMapVideo&VERSION=1.0.2&FORMAT=video/mpeg&CID=Se
nsorA2345&TIME=F930/F120/FS2&DUP=1&METADATA=All&BBOX=65.46185,32.64621,65.47011,32.65183&WIDT
H=1920&HEIGHT=1080&STYLES=&OPTIONS=mpeg.type%3Dmpeg2%26mpeg2.codec%3Dh264%26mpeg2.stream%3Dtr
ansport%26mpeg2.kbps%3D3000%26mpeg2.fps%3D30.00%26mpeg2.metadata%3DMISP61
```

30.3.3 XML encoding schema

See **[WAMI XSD]** element "**VS_GetMapVideoRequest**"

30.3.4 RequestParameter Details

30.3.4.1 Service

The value for this parameter shall be **VS**

30.3.4.2 Request

The value for this parameter shall be **GetMapVideo**

30.3.4.3 Version

Please refer to **[WAMI OVERVIEW].**

30.3.4.4 AcceptLanguages

Please refer to **[WAMI OVERVIEW].**

30.3.4.5 Format

This is a required parameter. Its value shall specify the format in which the requested data shall be received. The value of this parameter is a valid MIME type.

A **GetMapVideo** request is capable of delivering a response consisting of a video code stream.

It is the responsibility of the server to send back a list of supported formats in its Capabilities response. It is the responsibility of the client to specify the correct format name corresponding to the type of request being made.

If the client specifies an incorrect format, the server shall return an **InvalidParameterValue** exception. A detailed response from the server is highly recommended in event of such an error.

30.3.4.5.1 Example Capabilities Response for GetMapVideo's Format Parameter

```
<Parameter name="Format">
<AllowedValues>
<Value>video/mpeg</Value>
<!-- Options extends this -->
<Value>video/mp4</Value>
<Value>video/webm</Value>
<Value>video/quicktime</Value>
<Value>video/SMPTE292M</Value>
<Value>video/x-ms-wmv</Value>
<Value>video/mj2</Value>
</AllowedValues>
<Meaning>Set the format of a GetMapVideo response</Meaning>
</Parameter>
```

30.3.4.6 CID

This is a required parameter. The value of this parameter is a comma separated ordered list of collection identifiers, also known as Collection IDs. Each Collection ID identifies a collection that is unique to a collection tree. A Collection Service provides a list of all collections it is currently serving. For each collection, the Collection Service also provides all the services that provide data for that collection.

Each Collection ID identifies the unique ID of the Collection node that contains the desired collection of frames. A Collection Service provides a list of all collections it is currently serving. It also informs which of the collections has an Image Service available for serving up the maps as a flip book.

Example:`CID=CHMKV4S20110203,CHRSM5S20110203`

30.3.4.6.1 Spatial and Temporal Overlap in a GetMapVideo Request

Let us consider the possibility that a client requests for more than one map from more than one collection for a specific time window. In Figure 17 a client issues a request for a 1920x1080 HD AOI for a time window from T_{in} to T_{out}. The request is issued for collection IDs A, B, C, D and E. The collections overlap spatiotemporally.

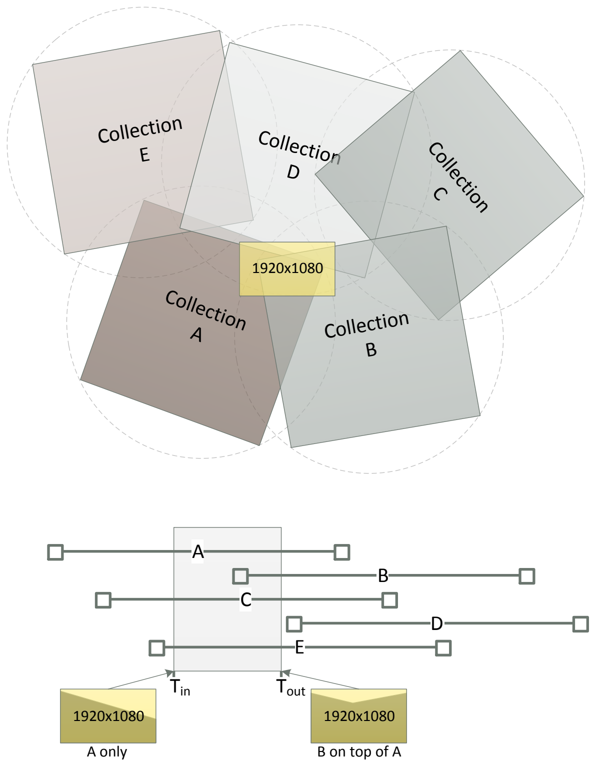


Figure 17: Spatial and temporal overlap in a GetMapVideo request

Spatially, collections A, B and D are overlapping. Temporally, for the given time window, collection D does not intersect the time window. Collection A intersects the time window completely. Collection B starts a little past midway into the time window.

The output maps shall be generated keeping in mind temporal and spatial overlap.

In this example, if the request includes, `CID=A,B,C,D,E&TIME=`T_{in}/T_{out}, then the output map images will contain pixels from collection A, from T_{in} until the start of collection B. From the start of collection B until T_{out}, the output map images will contain pixels from intersecting AOIs from collections A and B. Pixels from collection B will get rendered on top of those from collection A.

Finally, the output map images are transmitted as a video stream.

30.3.4.7 CRS

This parameter indicates the coordinate reference system of the values of parameter **BBox**. It also indicates the coordinate reference system that the maps are expected in. Detailed specification is as defined in **[OGC WSCS]**. Also see OGC's WMS and WCS specifications. If the source data is not in the specified coordinate reference system, then the image may be re-projected or processed to match the output CRS.

The Capabilities response of a **GetMapVideo** request shall provide a list of CRSs that are supported by the service implementation.

30.3.4.7.1 Example Capabilities Response for GetMapVideo's CRS Parameter

```
<Parameter name="CRS">
<AllowedValues>
<Value>EPSG:4326</Value>
<Value>EPSG:4269</Value>
<Value>EPSG:32645</Value>
</AllowedValues>
<Meaning>Coordinate reference system for BBox </Meaning>
</Parameter>
```

Example: `CID=CHMKV4S02032011&CRS=EPSG:4326`

30.3.4.8 BBox

This parameter sets the bounding box of the requested map. This parameter indicates the bounding box in the coordinate reference system set as the value of parameter **CRS**. This bounding box is fit within the output raster pixel area specified as values to parameters **Width** and **Height**. Detailed specification is as defined in [**OGC WSCS]**. Also see OGC's WMS and WCS specifications. If the source data is not in the specified coordinate reference system, then the image may be re-projected or processed to match the output CRS and then mapped to the specified **BBox**.

Example: `CID=CHMKV4S02032011&CRS=EPSG:4326&BBOX=-117.6788847,35.603284,-117.6704312,35.608133`

30.3.4.9 Time

This parameter follows the same conventions as **[WAMI IS]** GetMap

30.3.4.10 Dup

This optional parameter specifies how many times the client wants to duplicate the sequence of map frame numbers generated from the **Time** parameter. It is a positive rational number greater than or equal to zero. The absence of the `Dup` parameter, `Dup=`, or `Dup=0` mean do not duplicate, `Dup=1` means duplicate each frame once, and so on.

The video stream that is generated as a response to this request is setup to playback at a specific frame rate. The source data may have been acquired at a different frame rate.

Real-time playback implies playback at the same rate at which the data was acquired. If FPS_{acq} was the rate at which the data was acquired and the video stream playback rate is FPS_{stream}. If

$FPS_{acq} \neq FPS_{stream}$, then **Dup** can be set to FPS_{stream} / FPS_{acq}. The server can duplicate (or skip) frames to simulate real time playback.

Table 32: Use Dup to mimic real-time in 30 fps for non 30 fps WAMI capture

Rate at which original data was captured, FPS_{acq}	Desired output video stream rate, FPS_{stream}	**Dup** value to generate "real-time" playback experience
24	30	1.25
2	30	15
1.8	30	16.6667

For a value of **Dup** that has a decimal number in it, the server shall dictate how to interpret it. For instance, the server can do a field pull-down or pull-up, a fade, or skip to compute the transition. For example, for FPS_{acq} = 24; FPS_{stream} = 30, and Dup = 1.25, a server may choose to perform a 2:3 pull-down.

The accuracy of the transformation is determined by the values of **Dup**, FPS_{acq} and FPS_{stream} and the server-side method used to perform the transformation. Accuracy of the transformation is controlled by the server.

It is recommended that the client try their very best to keep **Dup** as an integer to frame interlace ambiguities.

30.3.4.11 Width

This required parameter specifies the width of the output video stream in pixels. This number shall be a positive integer greater than zero. A server may choose not to allow a width greater than some known value to avoid bandwidth and server-side resource saturation.

A Collection Service provides the pixel width and height of frames within that collection. The server fits the AOI specified in **Bbox** within a raster of the specified pixel width and height. The reason for the collection service to provide pixel width and height of the frames is to allow for a client to know what metrics the frame possesses. It also allows for the client to select a map with the same pixel size at which the frame was captured. **[WAMI IS]GetMapInfo**, if implemented, retrieves metadata about a specific frame.

Example: `WIDTH=1920&HEIGHT=1080`

30.3.4.12 Height

This required parameter specifies the height of the output video stream in pixels. It follows the same constraints as **Width**.

30.3.4.13 Styles

See **OGC** document number **06-042** namely **OGCWeb Map Server Implementation Specification version 1.3.0**.

30.3.4.14 Bgcolor

See **OGC** document number **06-042** namely **OGCWeb Map Server Implementation Specification version 1.3.0**.

30.3.4.15 Transparent

See **OGC** document number **06-042** namely **OGCWeb Map Server Implementation Specification version 1.3.0**. It can be TRUE or FALSE. Default value is FALSE.

30.3.4.16 Metadata

This parameter implementation is optional. A server may choose not to implement this option. In that case, if specified by a client, a server may choose to ignore this option.

If **Metadata** was not implemented it means the video stream shall not have any embedded metadata.

Contents of metadata are up to the vendor implementation and actual available metadata. The vendor may choose to have zero metadata elements. In that case, if the client requests for metadata, the response may not have metadata embedded in the video stream.

The value to this parameter shall be a comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty (…Metadata=&...), it means do not send metadata embedded in the video stream. If set to **All**, it means send all metadata associated with the node. At least the value of **All** shall be implemented. A complete list shall be provided as part of the Capabilities response to the **GetMapVideo** request. Besides **All**, implementing additional parameters is completely up to the server implementation. This parameter shall be implemented by both client and server.

If the metadata is packaged in a video stream, it shall follow a known standard, for example MISP 6.1.

It is highly recommended that for Government applications, Metadata be implemented. It is also highly recommended that if Metadata was implemented, that the video stream format support the MPEG2 codec and support MISP 6.1.

30.3.4.16.1 Example Capabilities Response for GetMapVideo's Metadata Parameter

```
<Parameter name="Metadata">
<!-- These categories are specified by the server implementation -->
<AllowedValues>
<Value>All</Value>
<Value>MISP61</Value>
<Value>MISP53</Value>
</AllowedValues>
</Parameter>
```

30.3.4.17 Options

This parameter provides additional server specific sub-options to existing parameters. For example, if the service supports `Format=video/mpeg`, it may choose to provide control over various MPEG parameters such as container format (e.g. MPEG2), codec (e.g.MPEG2, H.264), bit rate (e.g. 1500, 3000 Kbps), video playback rate (29.97, 30, 15), GOP size (1, 5, 20), stream type (transport or program), YUV profile (e.g. 422, 444, 420, 411). Please refer to **[WAMI OVERVIEW]**, version 1.0.2 for details on **Options**.

30.3.5 Example GetMapVideo Capabilities Response

```
<Operation name="GetMapVideo">
<DCP>
<HTTP>
<Get xlink:href="http://example.com/VS?"/>
</HTTP>
</DCP>
<Parameter name="Service">
<AllowedValues>
<Value>VS</Value>
</AllowedValues>
</Parameter>
<Parameter name="Request">
<AllowedValues>
<Value>GetMapVideo</Value>
</AllowedValues>
</Parameter>
<Parameter name="Version">
<AllowedValues>
<Value>1.0.2</Value>
</AllowedValues>
</Parameter>
<Parameter name="Format">
<AllowedValues>
<Value>application/xml</Value>
<Value>application/x-json</Value>
</AllowedValues>
<DefaultValue>application/xml</DefaultValue>
</Parameter>
<Parameter name="AcceptLanguages">
<AllowedValues>
<Value>en-US</Value>
<Value>en</Value>
</AllowedValues>
<DefaultValue>en-US</DefaultValue>
</Parameter>
<Parameter name="ID">
<AnyValue/>
<Meaning>A comma-separated ordered list of collection identifiers.
                    Each identifier is unique to a collection.
                    A multi-valued ID parameter implies compositing of collections.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Disposition">
<AllowedValues>
<Value>ordered</Value>
<Value>unordered</Value>
<Value>replace</Value>
</AllowedValues>
<DefaultValue>ordered</DefaultValue>
<Meaning>
                    This field is required if Metadata is requested or if multiple images are 
requested
                    (via specifying a Time Range).
                    Valid values are “ordered”, “unordered”, “replace”.
                    Servers that support multipart responses MUST implement “ordered”.
                    The “unordered” and “replace” values are optional both for client and server.
</Meaning>
</Parameter>
<Parameter name="CRS">
<AllowedValues>
<Value>EPSG:4326</Value>
```

```
<Value>EPSG:4269</Value>
<Value>EPSG:32641</Value>
<Value>EPSG:32642</Value>
<Value>EPSG:32643</Value>
<Value>EPSG:32644</Value>
<Value>EPSG:32645</Value>
</AllowedValues>
<Meaning>Coordinate reference system for BBox </Meaning>
</Parameter>
<Parameter name="BBox">
<AnyValue/>
<Meaning>
                    The bounding box of the map in the specified CRS.
                    General model: Bbox=minx,miny, maxx,maxy.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Time">
<AnyValue/>
<Meaning>
                    ISO 8601 time-range (in UTC time) or absolute frame-number range and optional
step value.
                    Shall be implemented by both client and server.
                    For implementation purposes, if the first character is ‘F’, then it is a
frame or frame-range based request.
                    Otherwise it must comply with ISO 8601 time or time-range.
                    For the frame-range the format is ‘F’ \d+ ( ‘/F’ \d+ ( ‘/FS’ \d+ )? )?
</Meaning>
</Parameter>
<Parameter name="Width">
<AnyValue/>
<Meaning>
                    Width in pixels of the output map pictures.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Heigth">
<AnyValue/>
<Meaning>
                    Height in pixels of the output map pictures.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Dup">
<AnyValue/>
<Meaning>
                    Duplicate frames. It specifies the number of times each frame within the time
range shall be duplicated.
                    Shall be implemented by both client and server.
</Meaning>
<DataType>Integer</DataType>
</Parameter>
<Parameter name="Styles">
<AnyValue/>
<Meaning>
                    Comma-separated list of one rendering style per requested collection.
                    An empty list implies default styles.
                    Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Transparent">
<AllowedValues>
<Value>TRUE</Value>
<Value>FALSE</Value>
</AllowedValues>
<DefaultValue>FALSE</DefaultValue>
<Meaning>
                    Background transparency of the maps (Default: Transparent=FALSE).
                    Shall be implemented by both client and server.
```

```
</Meaning>
<DataType>Boolean</DataType>
</Parameter>
<Parameter name="BgColor">
<AnyValue/>
<DefaultValue>000000</DefaultValue>
<Meaning>
                    Hexadecimal red-green-blue color value for the background color.
                    (Default: Bgcolor=0x000000). Shall be implemented by both client and server.
</Meaning>
</Parameter>
<Parameter name="Metadata">
<AllowedValues>
<Value>All</Value>
<Value>Basic</Value>
<Value>Geospatial</Value>
</AllowedValues>
<Meaning>
                    A comma-separated ordered list of zero or more names of sections of metadata
to be returned.
                    If not set or empty, it means do not send metadata with image data.
                    Shall be optionally implemented by both client and server.
                    When specified, the Disposition parameter needs to be set to allow multipart
documents.
</Meaning>
</Parameter>
</Operation>
```

30.3.6 Response

The response from a **GetMapVideo**request shall be a video stream. It is highly recommended that as soon as the initial elements of video become available, the server start transmitting.

30.3.6.1 XML Response schema

See **[WAMI XSD]** element "**ExceptionReport**" for requests with errors.

On success, there is no associated XML schema – Only raw video feeds with embedded metadata.

30.3.6.2 Exceptions

An error in a **GetMapVideo** request may generate the following **exceptionCodes**:

1. MissingParameterValue
2. InvalidParameterValue
3. OptionNotSupported
4. NoApplicableCode

30.4 GetPathMapVideo

This request returns a video of a sequence of one or more map images. For each returned image map, the request also optionally returns metadata about the rendered frame. The metadata is embedded in the video stream.

It is highly recommended that the reply be streamed as soon as it is available. Performance is key.

It is necessary that this request be implemented as a GET and a POST request. Due to the fact that data structures are going from the client to the server, a POST is recommended over a GET.

A server may choose not to implement the **GetPathMapVideo** request. If the server has implemented **GetPathMapVideo**, the capabilities response shall reflect so.

30.4.1 Request Parameter Summary

A table of the request parameters for **GetPathMapVideo** is provided below.

Table 33: GetPathMapVideo request parameter summary

Parameter name	KVP Example	Optionality and multiplicity	Definition, format and use
Service	`Service=VS`	Mandatory, one	The value shall be VS
Request	`Request=GetPathMapVideo`	Mandatory, one	The value shall be **GetPathMapVideo**
Version	`Version=1.0.2`	Mandatory, one	Version number of service. Shall be implemented by both client and server.
Accept Languages	`AcceptLanguages=en-US`	Optional, zero or one	Please refer to **[WAMI OVERVIEW]**
Exceptions	`Exceptions=XML`	Optional, zero or one	Sets the format of the exception.
Format	`Format= video/mpeg`	Mandatory, one	Output format of the video stream. Shall be implemented by both client and server.
CRS	`CRS=EPSG:4326`	Mandatory, one	The coordinate reference system of the requested maps. Shall be implemented by both client and server.
Width	`Width=1280`	Mandatory, one	Width in pixels of the output video stream. Shall be implemented by both client and server.
Height	`Height=720`	Mandatory, one	Height in pixels of the output video stream. Shall be implemented by both client and server.
Styles	`Styles=`	Mandatory, one	Comma separated list of one rendering style per requested collection. Shall be implemented by both client and server.
Transparent	`Transparent= TRUE`	Optional, zero or one	Background transparency of the maps (Default: Transparent=FALSE). Shall be implemented by both client and server.
Bgcolor	`Bgcolor= 0x000000`	Optional, zero or one	Hexadecimal red-green-blue color value for the background color. (Default: Bgcolor=0x000000). Shall be implemented by both client and server.
Metadata	`Metadata=MISP61`	Optional, zero or one	A comma-separated ordered list of zero or more names of sections of metadata to be returned. If not set or empty, it means do not send metadata with image data. Shall be optionally implemented by both client and server.
Options	`mpeg.type%3Dmpeg2%26mpeg2.codec%3Dh264%26mpeg2.stream%3Dtransport%26mpeg2.kbps%3D3000%26mpeg2.fps%3D30.00`	Optional, zero or one	Vendor specific options for this request. Extends the request. (Default: Options=). Shall be optionally implemented by both client and server.
Path	`Path=`*URL Encoded XML*	Mandatory, one	Specifies the tracks to be rendered by the service. Shall be implemented by both client and server.

30.4.2 XML Encoding schema

See **[WAMI XSD]** element "**VS_GetPathMapVideoRequest**"

30.4.3 Request Parameter Details

30.4.3.1 Service

The value for this parameter shall be VS

30.4.3.2 Request

The value for this parameter shall be **GetPathMapVideo**

30.4.3.3 Version

Please refer to **[WAMI OVERVIEW].**

30.4.3.4 AcceptLanguages

Please refer to **[WAMI OVERVIEW].**

30.4.3.5 Format

Specifies the MIME type of the format in which the user wants results. Its constraints are identical to the specification of Format within a GetMapVideo request.

30.4.3.6 Width

This required parameter specifies the width of the output map images in pixels. This number shall be a positive integer greater than zero. A server may choose not to allow a width greater than some known value to avoid bandwidth and server-side resource saturation. All map images generated by this request shall be of the same pixel dimensions.

30.4.3.7 Height

This required parameter specifies the height of the output map images in pixels. This number shall be a positive integer greater than zero. A server may choose not to allow a height greater than some known value to avoid bandwidth and server-side resource saturation. All map images generated by this request shall be of the same pixel dimensions.

30.4.3.8 Dup

This optional parameter specifies how to duplicate frames in the video stream. It follows the same rules and constraints as its **GetMapVideo** counterpart.

30.4.3.9 CRS

This required parameter specifies the coordinate reference system (CRS) of the output map images in pixels. This value is also the coordinate reference of the bounding boxes for each key AOI in a track.

30.4.3.10 Styles

This parameter is identical in behavior to **Styles** specifications for a **GetMapVideo** request.

30.4.3.11 Transparent

This parameter is identical in behavior to **Transparent** specifications for a **GetMapVideo** request.

30.4.3.12 Bgcolor

This parameter is identical in behavior to **Bgcolor** specifications for a **GetMapVideo** request.

30.4.3.13 Metadata

This parameter is identical in behavior to **Metadata** specifications for a **GetMapVideo** request.

30.4.3.14 Options

This parameter provides additional server specific sub-options to existing parameters. For example, if the service supports `Format=image/jpeg`, it may choose to provide control over various JPEG parameters such as image quality, JPEG encoding methods, a YUV profile, etc. Please refer to **[WAMI OVERVIEW]**, version 1.0.2 for details on **Options**.

30.4.3.15 Path

The Path parameter is a URL encoded XML document specifying a complex sequence of Tracks with a constant ViewPort. The Tracks can span multiple collections.

See **[WAMI XSD]** element "**PathMapType**" for details

30.4.4 Example GetPathMapVideo Capabilities Response

```
<Operation name="GetPathMap">
<DCP>
<HTTP>
<Get xlink:href="http://example.com/IS?"/>
</HTTP>
</DCP>
<Parameter name="Service">
<AllowedValues>
<Value>IS</Value>
</AllowedValues>
</Parameter>
<Parameter name="Request">
<AllowedValues>
<Value>GetMap</Value>
</AllowedValues>
</Parameter>
<Parameter name="Version">
<AllowedValues>
<Value>1.0.2</Value>
</AllowedValues>
</Parameter>
<Parameter name="Format">
<AllowedValues>
<Value>application/xml</Value>
<Value>application/x-json</Value>
</AllowedValues>
<DefaultValue>application/xml</DefaultValue>
</Parameter>
<Parameter name="AcceptLanguages">
<AllowedValues>
<Value>en-US</Value>
<Value>en</Value>
</AllowedValues>
<DefaultValue>en-US</DefaultValue>
```

```
</Parameter>
<Parameter name="ID">
<AnyValue/>
<Meaning>A comma-separated ordered list of collection identifiers.
                       Each identifier is unique to a collection.
                       A multi-valued ID parameter implies compositing of collections.
                       Shall be implemented by both client and server. </Meaning>
</Parameter>
<Parameter name="Disposition">
<AllowedValues>
<Value>ordered</Value>
<Value>unordered</Value>
<Value>replace</Value>
</AllowedValues>
<DefaultValue>ordered</DefaultValue>
<Meaning>Required if Metadata is requested or if multiple images are requested
        Valid values are “ordered”, “unordered”, “replace”.
        Servers that support multipart responses MUST implement “ordered”.
        The “unordered” and “replace” values are optional both for client and server.</Meaning>
</Parameter>
<Parameter name="CRS">
    <AllowedValues>
    <Value>EPSG:4326</Value>
    <Value>EPSG:4269</Value>
    <Value>EPSG:32645</Value>
    </AllowedValues>
    <Meaning>Coordinate reference system for BBox </Meaning>
</Parameter>
<Parameter name="BBox">
<AnyValue/>
<Meaning>The bounding box of the map in the specified CRS.
                       General model: Bbox=minx,miny, maxx,maxy.
                       Shall be implemented by both client and server. </Meaning>
</Parameter>
<Parameter name="Path">
<AnyValue/>
<Meaning>Specifies the tracks to be rendered by the service. Shall be implemented by both client
and server. </Meaning>
</Parameter>
<Parameter name="Width">
<AnyValue/>
<Meaning>Width in pixels of the output map pictures.
                       Shall be implemented by both client and server. </Meaning>
</Parameter>
<Parameter name="Heigth">
<AnyValue/>
<Meaning>Height in pixels of the output map pictures.
                       Shall be implemented by both client and server. </Meaning>
</Parameter>
<Parameter name="Styles">
<AnyValue/>
<Meaning>Comma-separated list of one rendering style per requested collection.
                       An empty list implies default styles.
                       Shall be implemented by both client and server. </Meaning>
</Parameter>
<Parameter name="Transparent">
<AllowedValues>
<Value>TRUE</Value>
<Value>FALSE</Value>
</AllowedValues>
<DefaultValue>FALSE</DefaultValue>
<Meaning>
        Background transparency of the maps (Default: Transparent=FALSE).
        Shall be implemented by both client and server. </Meaning>
<DataType>Boolean</DataType>
</Parameter>
<Parameter name="BgColor">
<AnyValue/>
<DefaultValue>000000</DefaultValue>
<Meaning>
```

```
Hexadecimal red-green-blue color value for the background color.
(Default: Bgcolor=0x000000). Shall be implemented by both client and server. </Meaning>
</Parameter>
<Parameter name="Metadata">
<AllowedValues>
<Value>All</Value>
<Value>Basic</Value>
<Value>Geospatial</Value>
</AllowedValues>
<Meaning>
A comma-separated ordered list of zero or more names of sections of metadata to be returned.
If not set or empty, it means do not send metadata with image data.Shall be optionally
implemented by both client and server.When specified, Disposition parameter needs to be set to
allow multipart documents. </Meaning>
</Parameter>
</Operation>
```

30.4.5 Response

The response from a **GetPathMapVideo**request can be in a format of the clients choosing from a list of formats that the server supports. The Options parameter further qualifies subtler controls available from the server. The response is a video stream.

30.4.5.1 XML Response schema

See **[WAMI XSD]** element “**ExceptionReport”** for requests with errors.

On success, there is no associated XML schema – Only raw video feeds with embedded metadata.

30.4.5.2 Exceptions

An error in a **GetPathMapVideo** request may generate the following **exceptionCodes**:

1. MissingParameterValue
2. InvalidParameterValue
3. OptionNotSupported
4. NoApplicableCode

-End of document-