OGC® DOCUMENT: 12-077R1

External identifier of this OGC® document: http://www.opengis.net/doc/bp/wami-primer/1.0

Open Geospatial Consortium

A PRIMER FOR DISSEMINATION SERVICES FOR WIDE AREA MOTION IMAGERY

OTHER

PUBLISHED

Version: 1.0
Submission Date: XXX
Approval Date: 2012-11-23
Publication Date: 2012-11-28
Editor: Rahul Thakkar

Note

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

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 standard set forth in this document, and to provide supporting documentation.

CONTENTS

LIST OF FIGURES

I EXECUTIVE SUMMARY

Management of Wide Area Motion Imagery (WAMI) is a growing Big Data problem. Current workflows have been hard pressed to "simply keep up". WAMI sensors come in different designs; from single CCDs collecting big pictures, to a matrix of cameras collecting a larger aggregate view. Gigapixel sensors collect color, luminance or IR data at more than 8-bits per band, several times a second. Storing WAMI data raw and uncompressed is ideal, but becomes unsustainable in the long term. Despite compression, moving the data from a forward source to data centers is still a challenge, even with high bandwidths.

Since 2007, PIXIA Corp has implemented an end to end solution that is sensor and data agnostic, delivering WAMI quickly and efficiently, no matter where it was located; on the aircraft, in a ground station, in a local data center or in archives. It required developing unique technologies, one of which was a set of web services. Using a RESTful HTTP grammar, the services provided a scalable, high-performance interface for WAMI in a global enterprise. The web services grammar was modeled on existing standards. Clients could get data from all WAMI sensors that until now required proprietary interfaces. The services were well received by the WAMI community. In 2011, PIXIA submitted the specifications to the Open Geospatial Consortium (OGC) for consideration for standardization.

This solution is currently deployed supporting fielded WAMI sensors in a global enterprise.

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.
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.

II SECURITY CONSIDERATIONS

No security considerations have been made for this document.

III REVISION HISTORY

DATE	RELEASE	AUTHORS	PARAGRAPH MODIFIED	DESCRIPTION
04/02/2012	Draft	R Thakkar	All	Original draft document
04/12/2012	Draft	J Zinter, M Maraist, R Thakkar	Edits, info, examples	Release candidate
04/20/2012	Draft	R Thakkar	Exec. summary, examples	Release candidate
05/02/2012	Draft	R Thakkar	Edits, appendix A & B, RS	Release candidate
08/30/2012	1.0	R Thakkar	Edits for best practice doc	For release to OGC

1 INTRODUCTION

1 INTRODUCTION

The reason for developing this specification was a WAMI community requirement to deliver high performance web services and disseminate WAMI products. 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.

The services are consumer centric. The services 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 behaviors.

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 modeled for implementation on inexpensive commodity compute and storage infrastructure permitting cloud vendors to run these at the platform layer.

PIXIA Corp has submitted document number OGC 12-032r2 for consideration as a standard for dissemination of Wide Area Motion Imagery (WAMI) data and metadata. OGC 12-032r2is titled *OGC WAMI Services – Dissemination Services for Wide Area Motion Imagery*. Every attempt was made to make the specification compliant with OGC standards. Web services based on this document have been deployed and are in use by the community. The document is currently under review as a best practice document.

Specifications are generally dry documents. This document serves as a primer for OGC 12-032r2. It introduces WAMI and the WAMI services specifications as submitted in OGC 12-032r2. It is intended for audiences that wish to review, understand, comment on, or implement using OGC 12-032r2. Portions of this document may apply to different audiences. Feel free to skip over parts that are not relevant to you.

2

WHAT IS WAMI?

2 WHAT IS WAMI?

Wide Area Motion Imagery (WAMI), in its various forms, is also referred to as Wide Area Airborne Surveillance (WAAS), Wide Area Persistent Surveillance (WAPS), Persistent Wide Area Surveillance (PWAS), Wide Area Surveillance (WAS), and Large Volume Streaming Data (LVSD). There are a host of other terms that may also be in use in the community.

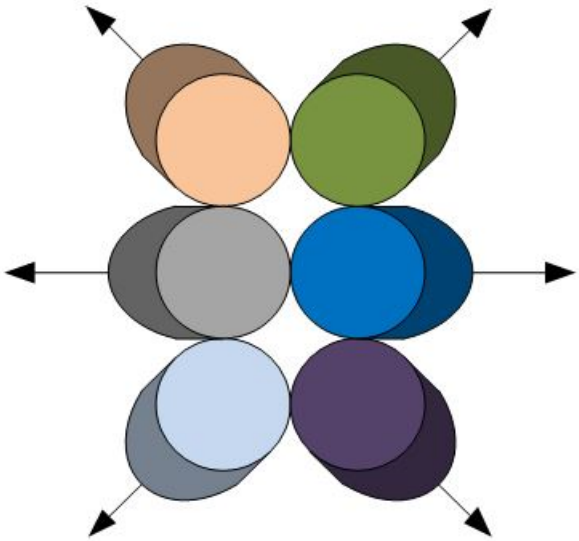


Figure 1 — Example WAMI sensor

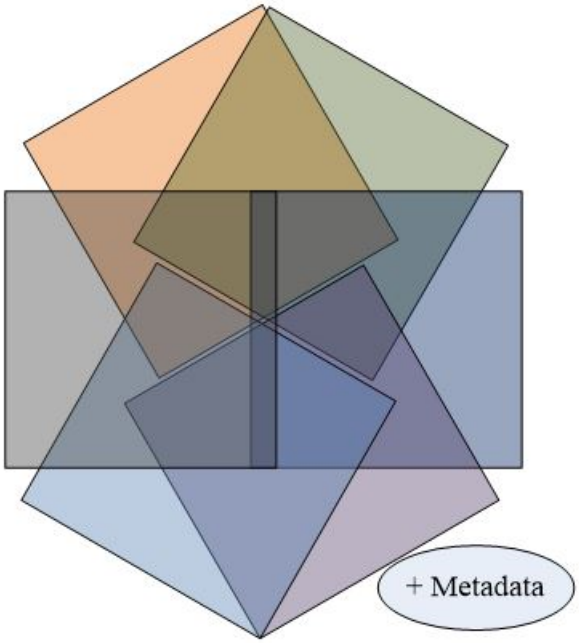


Figure 2 — Six-camera POV

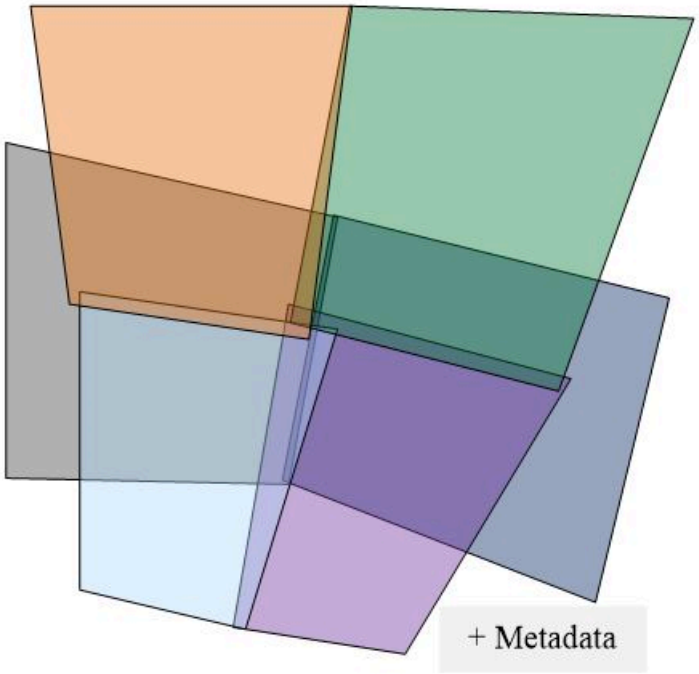


Figure 3 — Six-camera ground coverage

Figure 4 — One ortho-rectified WAMI frame with one AOI zoomed in at 100%

Simply put, WAMI captures a *video* of an area the size of a town or city, day and night. It is a system that uses one or more *cameras* mounted on the some form of a gimbal on an aircraft or blimp to capture a very large area on the ground, from about once every second up to several times per second. *Persistent surveillance* captures the same general area on the ground over a length of time.

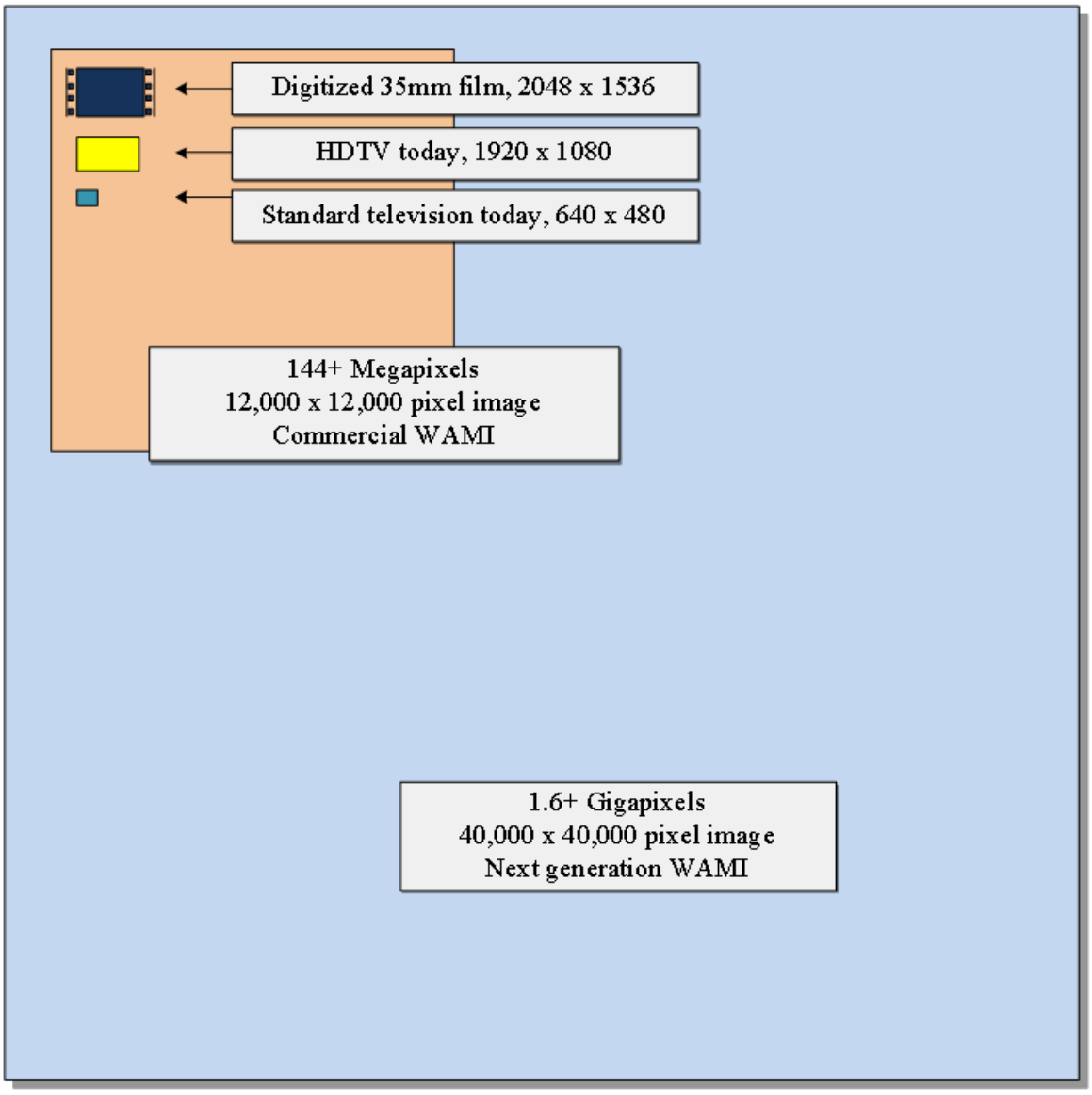


Figure 5 — WAMI image size comparison

For example, a WAMI sensor may have six high resolution cameras that may take pictures about twice a second. Set to look down and face outwards from each other at a slight angle, a picture from each camera overlaps pictures from adjacent cameras. The system is mounted inside or outside an airborne platform. The platform also has GPS / IMU capability. The system records pictures, and position data.

Using this information, for each instant in time, a software tool can combine separate images into one large image. It can perform image processing operations using digital elevation models (DEMs) to generate fairly accurate maps many times a second. Each map may have geo-spatial data, a sub-second time of acquisition, and additional metadata complimenting the captured data. WAMI can be used for pattern analysis such as tracking, forensics, and fused with sensors to create a common operating picture.

Different sensors may implement radically different technologies. Sensors may change from being a mosaic of CCDs, a single large CCD, or a mosaic of multiple cameras. The end result is a big picture of a vast region on the ground taken frequently. An example is shown in Figure 1 to Figure 4.

3

GENERATED DATA

3 GENERATED DATA

Consumers may need data in its unmodified format and as a derived product e.g. ortho-rectified image. Both can be utilized for a variety of spatiotemporal analyses.

Examples of a WAMI image size are 60, 96, 144, 150, 1100, 1600, 1800, or 2200 megapixels or more. The pixels may be 8-bit or 16-bit, may contain color, black and white, VNIR, MWIR or other EM-frequency ranges. An example WAMI image may contain 60 MB to 12 GB or more of uncompressed data. At 1600 megapixels or 1.6 gigapixels per frame, one WAMI image would be 772 times larger than full HD.

3.1. Data volumes

Data volumes quickly become an issue. To illustrate this, let us focus on WAMI as derived ortho-rectified image maps. One150-megapixel color WAMI image frame at 8-bits per band uncompressed is about 450 MB. At 10:1 compression, it is 45 MB/frame. At 2 frames per second, 86,000 frames cover 12 hours at 3.7 TB per day per sensor. At night, the same sensor would have one IR band instead of 3-bands and could be at 16-bits/band, resulting in about 2.5 TB of data for the night time. 1.6 Gigapixel sensor scan create over 100 TB a day. Moving so much data electronically is impractical. It is acquired at a one location and needs processing at another location. Clients may need data on board a platform, from local ground-stations, and archives in data centers. Maintaining multiple copies is cost prohibitive.

3.2. Proprietary formats

To understand consumption of this data, let us relate it to aerial and satellite imagery as it is captured today. GIS consumers of satellite and aerial imagery get the data as maps. Generally, raw camera data is either rasterized with rational polynomial coefficients (RPCs), or ortho-rectified into a known coordinate reference system and presented as a map mosaic.

WAMI sensors have been fielded by the US Government and commercially. Sensors pack different technologies. They deliver different effective megapixel ratings at various frame rates from 1 to 10 fps that may be non-constant. The format in which camera data is delivered is almost always proprietary. Map generator tools are also proprietary. Consequently sensors require special integration. Some sensors directly produce an ortho-rectified image per frame in a standardized file format such as GMLJP2, time of acquisition (TOA) and additional metadata. Regardless of the captured format, data transmission is a problem. Vendors have used everything from FTP to custom TCP sockets to JPIP with varied success.

4 CONSUMER

4 CONSUMER

A consumer of this data can be a human analyst or an automatic data processing algorithm. At some point, either humans or algorithms need WAMI imagery in some form.

To start, a consumer wants to know what's available. A consumer can ask, "What do you have?" or, "Give me a list of available data matching this search criteria", typically geospatial or temporal based. Once a consumer knows what is available, they may focus on a certain Area of Interest (AOI) on the ground, and across a window in time. Consumers may want images from several AOIs blended into one output that is comprised of data from various sensors, flights, geographic regions and time. Consumers have the option of specifying panning and zooming when requesting the blended output. By combining the two concepts, a service would have the ability to smoothly zoom-in, out, and pan around while flipping through time. They may want this data as maps in a known reference system. The consumer may want specific portions of the raw data, unmodified, and in a format of choice based on the same search criteria as the ones used for derived data. The consumer may want metadata about the raw data, in order to locate and interpret the raw data.

From a consumer's perspective, if they have access rights to the data, there should be no difference in the client experience whether the data is on the ground or being captured live, directly from the capture platform, raw or derived.

5

OPERATIONAL USE

5 OPERATIONAL USE

The introduction of WAMI to operational users provided a monumental gain in Intelligence, Surveillance, and Reconnaissance (ISR) collection and exploitation. Considerable Geospatial Intelligence (GEOINT) was available from a single platform. Full Motion Video (FMV) platforms provided a small field view on a specific target or area of interest. WAMI provides persistent coverage over large areas which previously required multiple FMV platforms. Operational commanders can send a single platform over an area of interest. Both forward operators and exploitation centers in the rear can use this data to meet their objectives.

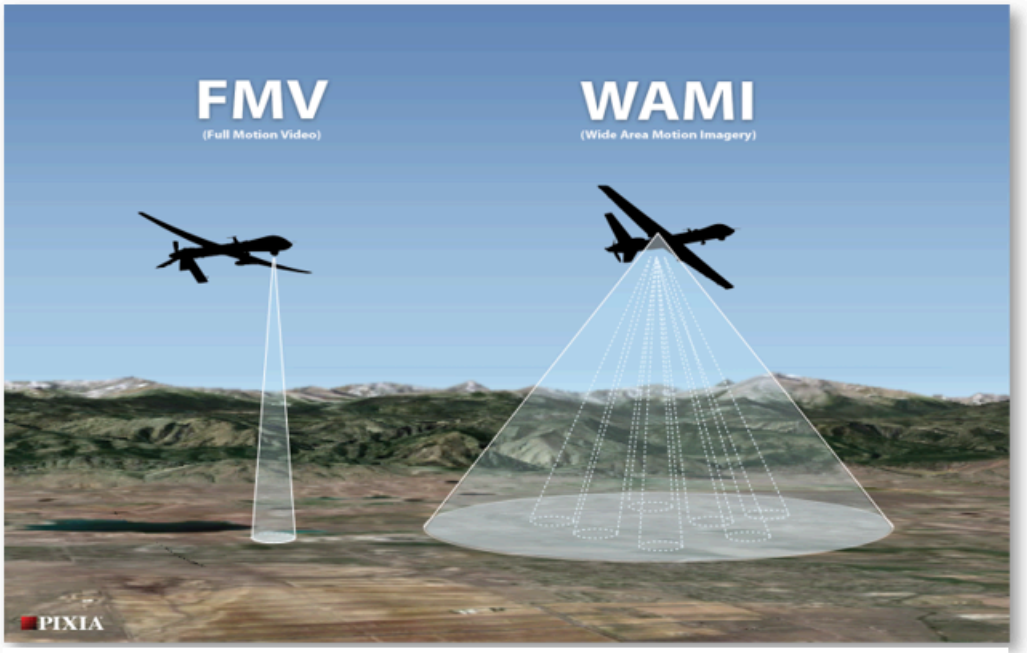


Figure 6 — FMV vs. WAMI coverage

To meet their objectives, exploitation centers employ different Electronic Light Table (ELT) software. These ELT packages can use the proposed standard to stream and exploit WAMI data. This is a positive shift in implementing services to meet changing ISR needs without impacting their architecture. The exploitation workflow is beginning to employ more automatic methods to consume imagery which in-turn generate more data, such as tracks, events, moving target indicators (MTI), and video clips.

Current initiatives will applyinnovative methods to analyze exploding amounts of big data. New analytics tools will generate what is known as activity based intelligence (ABI) from WAMI. The value of WAMI increases when merged with sources such as Light Detection and Ranging (LiDAR), Hyperspectral Imagery (HSI), National Technical Means (NTM) imagery, aerial maps, Synthetic Aperture Radar (SAR), FMV, traffic cameras, and metadata feeds such as ABI metadata, ground moving target indicators(GMTI) tracks, and signal intelligence (SIGINT).

6

WEB SERVICES

6 WEB SERVICES

To retrieve data in forms stated above, a consumer needs global access to WAMI data. This is a problem of communication, performance and scale. If we break the general problem of WAMI data dissemination down into smaller problems, each standing by itself, we can address them separately and simply. This approach produces functional, modular, small, and independent solutions.

Data is remote, often times in limited compute environments. It is hard for a client to be next to the data. This solution should make it so that it is not necessary for a data consumer to be next to the data. Therefore, each independent solution is a web service, implementing a well-designed grammar meant for today's defense networks and the Internet. WAMI web services define the grammar. The format is controlled by various format standardization bodies. Customers dictate which format is required by them. Service implementations cater to customer needs.

Clients and servers implementing WAMI services communicate in a pre-defined language using stateless, secure HTTP. RESTful web services are used to provide data on-demand, only as much as requested, and in a format of choice.

6.1. Web services grammar and data formats

When talking about web services, it is critical to decouple the protocol or *grammar* in which a client and server communicate from the *format* of the delivered content.

- The *grammar* comprises of the syntax and semantics that a client and server expect when communicating.
- The content *format* comprises of the expected layout of a server's response.

OGC 12-032r2describesa grammar. HTTP requires that content be delivered in IETF-friendly formats. WAMI specifications in OGC 12-032r2 do not impose content format types. The IETF has clearly defined rules for extensible content formats. All formats (including future formats) can be supported by the grammar.

A grammar does not control formats. The grammar facilitates extensible format support.

7

WAMI DATA DISSEMINATION

7 WAMI DATA DISSEMINATION

Someone somewhere will require WAMI image data and associated metadata, whether it is for human or machine processing, to derive data and metadata, and draw conclusions. There are two forms in which WAMI data and metadata may be disseminated:

1. In the original raw form as acquired, in camera space
2. In a derived form, e.g. planar or ortho-rectified raster tiles in a defined spatial reference system.

Both forms require WAMI data to be delivered as follows:

1. Deliver portions of original camera data and metadata for AOIs
    a) Data and metadata blobs distributed in custom or known formats
    b) Standard data descriptors describe custom data layout
2. Deliver AOIs as a sequence of one or more images plus optional metadata
    a) Images are in various globally standard formats, e.g. JPEG, PNG, GIF, etc. and geo-formats like GMLJP2, GeoJP2, GeoTIFF NITF, etc.
    b) List of image formats is extensible
3. Deliver AOIs as a video clip of one or more images plus optional metadata tracks
    a) Videos are in various globally standard and popular formats, e.g. MXF, AVI, WMV, QuickTime®, Flash®, MPEG1, MPEG2, MPEG4, OGG, WEBM, etc.
    b) Video stream supports standards from groups such as SMPTE, and MISB for the US DoD.
    c) List of video formats is extensible.

There are multiple ways of delivering WAMI metadata. They include delivering metadata for:

1. One or more AOIs on a per frame basis (metadata about individual frames)
2. One or more AOIs on a per request basis (overall metadata about a request)
3. A mission or collection (overall mission information)
4. A set of missions or collections (information about WAMI data provisioned by the service)

Metadata can be delivered in standard forms like XML, JSON, netCDF, HTML or even plain text. For XML, the schema regulates content (E.g. GML, KML, SensorML, etc.). A lot of metadata can be delivered incrementally and hierarchically through a well-designed schema and compressed on-demand.

The grammar defines the language of communication. The format is kept independent of the grammar. However, best practices warrant that interchange standardization be required with delivered content. Non-standard formats, while discouraged, are acceptable if and only if service implementations provide the format of that data explicitly, in an industry accepted (standard) and published form.

8

WAMI WEB SERVICES

8 WAMI WEB SERVICES

This section answers the question of "What types of web services should serve WAMI data?" To identify these services, the first step is to set ground rules to maintain design focus.

1. AWAMI service should perform no more than one task and do it well.
2. AWAMI service specification should be simple, extensible and easy to develop to.
3. AWAMI service grammar should be based on current generation models and specifications.
4. An existing specification should not be jerry-rigged to fit WAMI needs.
5. For WAMI, X, Y, Z and T are one. (Z → zoom-level)

Using these principles, we have based the WAMI services model on RESTful HTTP following the OGC paradigm for its grammar. We used OWS, WMS, WCS, KML, GML, SensorML, AWS, HTTP 1.1 WebDAV, and a few other specifications as a reference point and defined simple WAMI services. WAMI services were divided into three types:

1. **Core services** manage and access WAMI data directly. They may consult each other as well
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

8.1. Core services

WAMI SERVICE	ABOUT	REQUESTS
Collection Service (CS)	Informs a client as to what data is being served by this service. It provides metadata incrementally to avoid client information overload. CS can be implicitly federated. CS presents metadata about the data hierarchically with a list of available content delivery services (IS, VS…) as part of that metadata; in a format of choice.	GetCapabilities, GetHelp, GetCollectionCount, Get Collections
Image Service (IS)	Given the input of one or more areas of interest as bounding boxes, one or more time windows, a coordinate reference system and other data; IS delivers maps as images and metadata in a format of choice. CS also delivers metadata that turns an IS into a tile service and an AOI flip-	GetCapabilities, GetHelp, GetMap, GetMapInfo, Get PathMap, GetPathMapInfo

WAMI SERVICE	ABOUT	REQUESTS
	book service. The response can be an image, image + metadata, frame metadata, multiple images, multiple images + metadata, metadata for frames.	
Video Service (VS)	This service is similar to IS. Instead of delivering content as a flipbook of separate images and metadata, it delivers popular / standard video streams.	GetCapabilities, GetHelp, GetMapVideo, GetPath MapVideo
Raw Service (RS)	This service provides a simple spatiotemporal interface to deliver the original content, unmodified.	In development

8.1.1. Which service to use and when?

For any level of overall metadata, use CS. If pixels have to be manipulated (for example, planar or ortho-rectified to serve a map), use IS, or VS. For raw camera space data, unblemished, use RS. IS and RS can serve the same dataset. As web services, they can operate on the same dataset to get different results — IS provides a derived result like raster tiles and AOI images, RS provides the original raw data.

8.2. Alert Services

Alert services answer the question, “For a given datasets I am interested in, how do I know that something changed?” where, “something changed” can be: A new dataset appeared, a sensor started collecting and live data is available on-board, missions were archived due to age or lack of use, etc. They can be of two forms: *push* and *pull*. We recommend using RSS for pulled alerts and XMPP for pushed alerts. GeoRSS GML fulfills WAMI needs. GeoRSS Simple is adequate for basic WAMI needs. The GML schema can be pulled using GeoRSS services or pushed via XMPP-based service.

WAMI SERVICE	ABOUT
RSS	A client pulls content from an RSS feed. GeoRSS is recommended. GeoRSS GML delivers all WAMI needs. GeoRSS Simple is adequate for critical WAMI data.
XMPP	XMPP services push data to subscribing clients. GML is the recommended schema.

8.3. Derived services

Derived WAMI services utilize CS, IS, RS or VS to perform their task.

8.3.1. Query Service

A Query Service (QS) lives on top of collection services. It is analogous to putting a search engine on top of XML metadata. Existing search engine interfaces are becoming de facto standards. Google, Amazon, and Oracle are examples. Any of these could act as QS grammar. Research is recommended.

8.3.2. Live WAMI

By virtue of having a CS and an IS, a "flipbook" based live playback service can be implemented. A client app would get overall metadata about a mission that is being collected. For example, as part of the CS response, a service informs the client that the mission end point is unbounded (*live*), along with the most recent frame's number and time stamp. Using an IS, should a client try to get an image past *"now"*, an exception reports information about un-captured frames. The most recent frame's metadata indicates that it is so. Such coordinated metadata reporting permits a client to develop a frame-based DVR[1] with pan/zoom control. This model merges the notion of *live* and *archive* in one. The next step is to create a "streaming video channel" updating based on control commands.

PIXIA Corp is developing a Video Control and Streaming Service (VCSS) specification. VCSS employs HTTP to control what is displayed in channels of video streams. A client sends commands such as pan, zoom, and play real-time, faster, slower, jump to live, change settings, etc. A server updates AOIs in time and space. The related video stream reflects it. The feed is a "postage stamp view" over a large area. Video could be UHD, HD or less, based on connection capabilities. The video may be 30 Hz, but the pictures in the video may update at a client controlled rate or at the acquisition rate, depending on whether you are playing from the *"archive"* or *"live"*. VCSS does not control a sensor or its platform.

[1]DVR — Digital Video Recorder, for example, TiVO®.

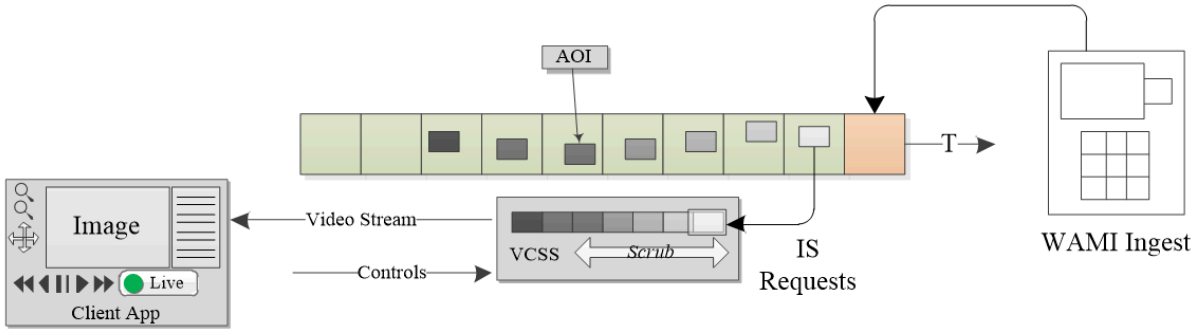


Figure 7 — Example VCSS pipeline

8.3.3. Virtual File System

While SOA caters to today's internet model, file access is still common. Tools require image and video files, and XML. Vendors may not have time to implement HTTP services. They may have to support legacy client tools that are many years old, and have been adapted to suite immediate WAMI needs. If a file-based WAMI service was available, it could assist in a phased migration to web services.

8.3.3.1. *Avoid vendor lockdown*

WAMI sensors and vendors can store and manage data the way they seem fit. If WAMI data is ingested into a system by one vendor how can a customer switch to another vendor? If a vendor presents WAMI data over HTTP as files, migration is as trivial as initiating a copy from one vendor to another.

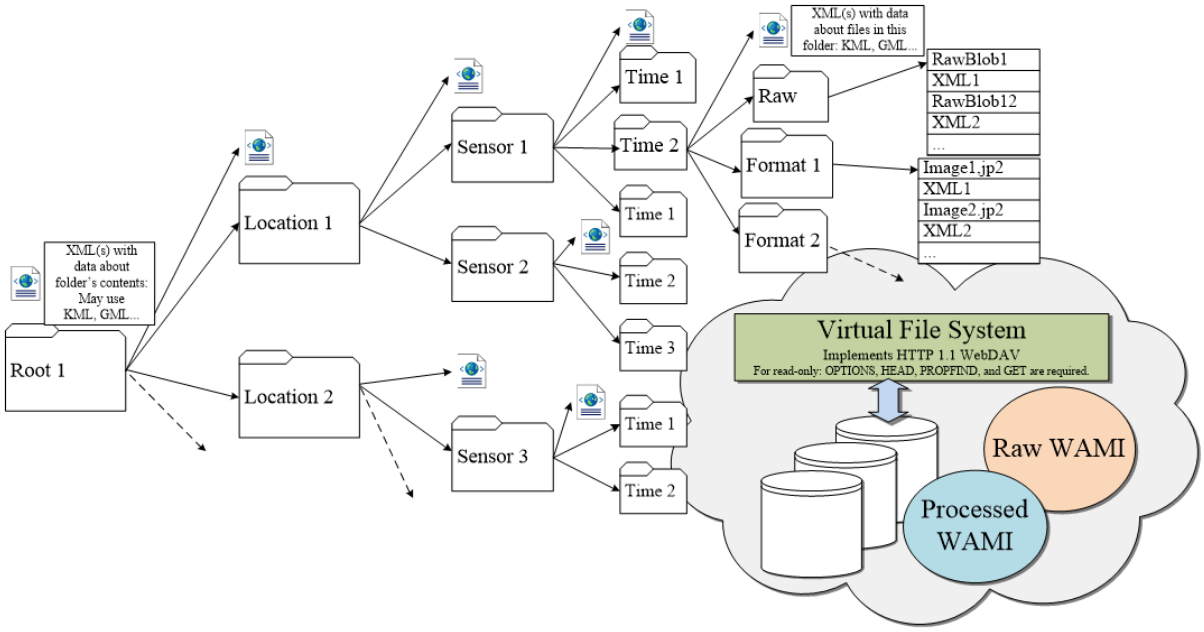


Figure 8 — Virtual file system using HTTP 1.1 WebDAV

8.3.3.2. *HTTP 1.1 WebDAV*

After extensive research, we found that HTTP 1.1 WebDAV (RFC 2518) with read-only capability is sufficient. Implementing HEAD, OPTIONS, PROPFIND and GET is adequate. Presenting WAMI data as a sequence of files in a customer supported format is critical for:

1. Avoiding vendor lockdown
2. Vendor to vendor migration
3. Providing a transitional solution for file-based tools

A vendor can also virtualize a WebDAV implementation by presenting WAMI data in multiple layouts via a Virtual File System (VFS). The folder structure in Figure 8 is just an example. A folder layout may be vendor specific. It is prudent to have XMLs (or JSONs) at each folder's level with relevant. VFS folders, files and XMLs don't really exist. A VFS makes you believe they do and presents a file system experience over secure HTTP. WebDAV clients ship free with Windows, Linux and Mac.

WAMI SERVICE	ABOUT
Query Service (QS)	QS filters one or more CS. It's a search engine over a set of CS. See NoSQL HTTP APIs by Google, Amazon, Oracle, etc.

WAMI SERVICE	ABOUT
Video Control and Streaming Service (VCSS)	VCSS provides DVR-like controls over HTTP with the video stream itself coming over a “channel” which could be streams of UDP or TCP.
Virtual File System (VFS)	VFS makes a client believe that WAMI data is a set of folders and files in a known format. It presents raw or processed WAMI in its original form as image files, and XML files, and derived ortho-rectified frames as image files of known types. The files don’t actually exist.

9

OGC 12-032R2 DESCRIBES CS, IS AND VS

9 OGC 12-032R2 DESCRIBES CS, IS AND VS

OGC 12-032r2 presents Collection, Image and Video Services. All other services have not been presented in that document. Further work is warranted and community participation is requested using an OGC SWG on WAMI. PIXIA has offered to start it.

10

COMMON MODEL

10 COMMON MODEL

The following section covers aspects common to all WAMI services.

10.1. RESTful HTTP

The term RESTful HTTP is used to represent a stateless HTTP request/response model, a client and server processing one independent HTTP request at a time. WAMI services use a RESTful architecture.

10.1.1. HTTP request

The general model of the grammar is derived from OGC. All services support HTTP GET and POST. Details are in the main document and most readers familiar with OGC specification already know all of this in great detail. If a request is short (about 2KB), use GET; whereas a POST permits unbounded multi-valued input parameter sequences, i.e. really long requests.

A GET request is of the general form `:port/path[?querystring]`. A `querystring` is a KVP[2] set of the form: `name=value[&name=value]`. Example: `http://example.com/IS?Service=IS&Version=1.0.2&Format=text/xml&Request=GetCapabilities`. A POST request: `:port/path`. The body of a POST request contains the KVP or `name=value` pair encoding. For `name=value` pairs, name is not case sensitive and is of the form `part[.part]`. For example, `VERSION=1.0.2` and `Version=1.0.2` are valid and `name` can be `options.mpeg2.codec` or `Request`.

10.1.2. HTTP response

The layout of an HTTP response is standardized based on the request type. Exception reporting is drawn directly from OGC. Each request has a corresponding section on response schema in OGC 12-032r2.

10.2. Required parameters

As with OGC services, all WAMI requests require SERVICE, REQUEST and VERSION.

1. SERVICE identifies a CS, IS, VS, RS, etc.E.g. `Service=CS`

[2]KVP – Key Value Pair or name-value pair such as `Version=1.0.2` or `Service=CS` or `Request=GetMap`

2. REQUEST identifies the operation to be performed. E.g. `Request=GetMap`
3. VERSION identifies the WAMI specification version a client wants to use. E.g. `Version=1.0.2`

10.3. Common WAMI requests

GetCapabilities and **GetHelp** are common to all WAMI services. The mandatory **GetCapabilities** request informs a client about all the requests the server supports and how. **GetHelp** is optional and allows vendor to deliver customer support documentation. WAMI GetCapabilities for all services is summarized in Annex A. See examples in Annex B.

11

TIME

11 TIME

Time value or range is set by the TIME parameter as absolute frame numbers or as ISO time. Frame based access is efficient, TIME based access is flexible. Both forms of time are explicit and do not leave room for ambiguity. The tables below have been captured from the original document to summarize the syntax. A client can ask for image data or metadata:

1. At a single instant in time
2. Within a window in time, with controlled increments
3. Within multiple windows in time, with controlled increments for each time window

11.1. Time value KVP syntax

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
$\mathbf{value}_1$, $\mathbf{value}_2$, valu ...	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 bounds are time values.	Get multiple maps, starting from min, up to and including *max*, in steps of *resolution*.
$\mathbf{min}_1/\mathbf{max}_1/\mathbf{res}_1$, $\mathbf{min}_2/\mathbf{max}_2/\mathbf{res}_2$ ...	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.
$\mathbf{int}_1/\mathbf{value}_1/\mathbf{res}_1$, $\mathbf{int}_2/\mathbf{value}_2/\mathbf{res}_2$ ...	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 (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.
$\mathbf{int}_1/\mathbf{min}_1/\mathbf{max}_1$, $\mathbf{int}_2/\mathbf{min}_2/\mathbf{max}_2$...	A list of multiple intervals.	Repeat the above process for multiple time intervals.

11.2. Frame basedTIME

SYNTAX	INTERPRETATION
F[FRAME]	Get one map from frame number FRAME.
F[START]/ F[END]/FS[STEP]	Get multiple maps from frame number START, up to and including frame number END, incrementing the frame number from START in steps of STEP. To go backward, set END < START. STEP $\neq$ 0.
R[N]/F[START]/ FS[STEP]	Get N frames starting from START, increasing each subsequent frame by STEP. STEP is an integer. STEP < 0, go backward. STEP > 0, go forward. STEP $\neq$ 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.

Frame based time is efficient. It directly maps to required data. ISO time of acquisition (TOA) requires mapping the time to an absolute frame. Client's ISO time map 1:1 with a frame's TOA. If it does not, any ambiguity is resolved as specified in the "**Temporal mismatch**" section below.

EXAMPLE KVP REQUESTS	MEANING
Time=F100	One map of frame number 11 from a collection
Time=F100/F2629	From frame 100 to 2629; expect 2530 maps
Time=F100/F2629/FS2	From frame 100 to 2629; in increments of 2; expect 1265 maps
Time=F200/F100/FS2	From frame 200 to 100 (backward); in increments of 2; expect 51 maps
Time=R10/F200/FS2	From 200; in increments of 2; expect 10 maps from 200, 202 ... 216, and 218.
Time=R10/F200	Expect 10 maps, from frame 200 up to and including 209.
Time=R10/F200/FS-2	From 200; in decrements of 2; expect 10 maps from 200, 198, 196 ... 184, and 182.
Time=F1,F11,F21,F31,F41	Expect 5 maps from frames 1, 11, 21, 31 and 41 in that order.
Time=F1/F11,F21/F31,F34/F44	Expect 33 maps 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; frames 21 to 31 in steps of 3; frames 34 to 44 in steps of 2; expect 16 maps.

11.3. ISO time based TIME

TIME parameter value is also based on ISO 8601. It is best illustrated by way of examples.

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/P3D	Retrieve all maps between specified interval(s), with each map from the 2008 date to the 2009 date at a 3 day intervals.
Time=2008-08-20T15:04:26.772Z/2009-08-20T15:29:35.9733Z/P1M15D	Retrieve all maps between specified interval(s), with each map from the 2008 date tothe 2009 date at a 1 month and 15 day interval.
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 within two separate intervals (playlist), the first range is at a month's interval per frame and the second range is at a day's interval per frame.
Time=2008-08-20T15:04:26.772Z/2008-08-20T15:29:35.9733Z/PT1.5S	Retrieve all maps between the specified ranges, with every map from the 2008 date until the 2009 date at a 1.5 second intervals.
Time=R100/2008-08-20T15:04:26.772Z	Retrieve 100 consecutive maps starting from the 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 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 2008 date, from 15:04:01 until 15:04:90 UTC, moving forward in time. Based on the request for 60 consecutive maps, the period computes to about PT1.509S.

12

COLLECTION SERVICE

12 COLLECTION SERVICE

The Collection Service or CS allows a client to discover what collections of WAMI data are being served by a service implementation. See examples in Annex B.

12.1. Definition of a WAMI data *Collection*

A **collection** of WAMI data is a **set** of WAMI data such that each **element** or **frame** of this set 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. Data storage and retrieval model is specification independent.

12.2. CS data model

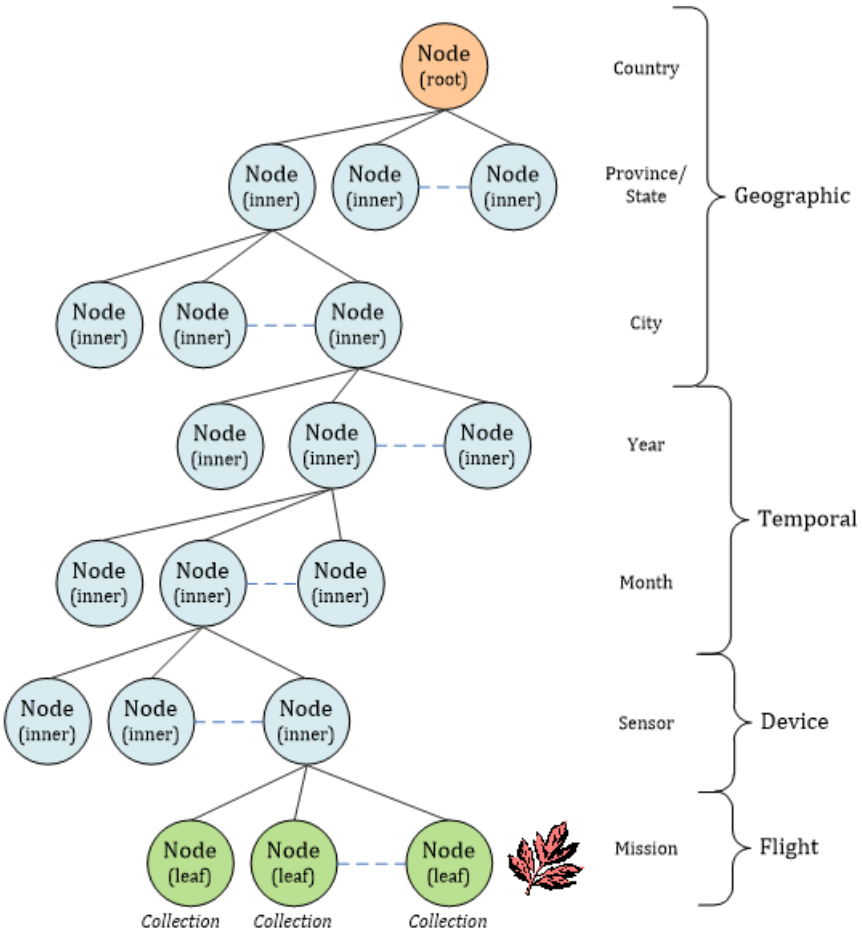


Figure 9 — Collection tree served by a collection service

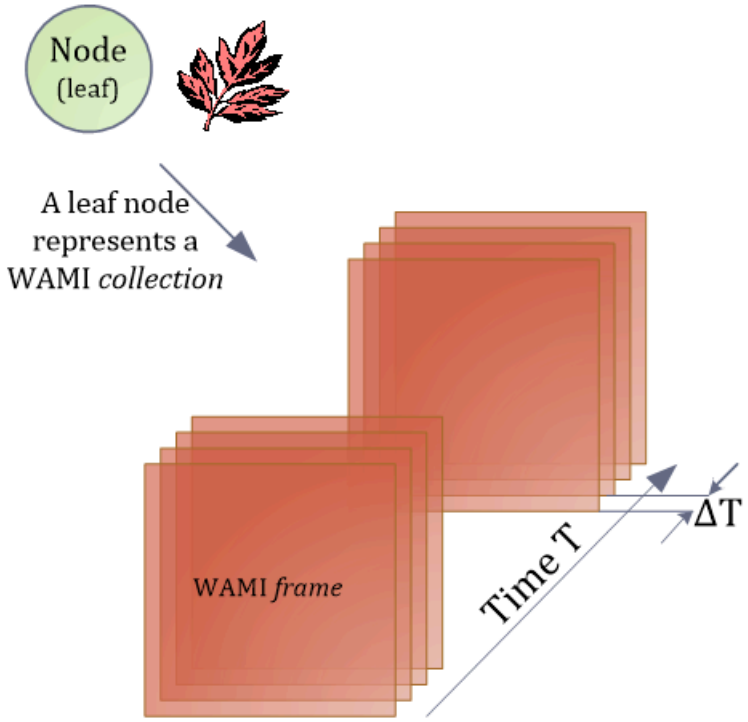


Figure 10 — A leaf node is a WAMI collection

CS draws its structure 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, a **Collection Tree** 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.

In the figure, you can also see a possible organizational structure that may get used practically.

12.3. Summary of all requests

REQUEST	MEANING	PARAMETERS	OPTIONALITY
GetCapabilities	Allows the client to retrieve metadata about the capabilities of the CS service 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
GetCollection Count	Retrieves the number of collections being served by the server	`Service, Request, Version, Format, AcceptLanguages, NID, Depth, CRS, Bbox, Time`	Optional
GetCollections	Retrieves the set of collections being served by the server. Result is as a tree. Incremental node by node traversal is available from any tree node. Complete or partial collection metadata can be retrieved.	`Service, Request, Version, Format, NID, Metadata, Depth, CRS, Bbox, Time`	Mandatory

13

IMAGE SERVICE

13 IMAGE SERVICE

As an image and metadata flipbook service, the Image Service or IS delivers a view into WAMI collections. It delivers derived content as maps and metadata from the original camera-space data or from data that is already available in a derived content space. See examples in Annex B.

13.1. Summary of all requests

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 interleaved images & metadata, in standard formats. See Appendix A for a summary.	`Service, Request, Version, Format, AcceptLanguages, Disposition, CID, CRS, BBox, Width, Height, Styles, Transparent, Bgcolor, Time, Metadata, Options`	Mandatory
GetMapInfo	The request is similar to GetMap except it retrieves metadata only.	`Service, Request, Version, Format, AcceptLanguages, CID, CRS, BBox, Width, Height, Time, Metadata, Options`	Optional
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 data for collection, time periods and bounding AOIs is supplied as a path[a].	`Service, Request, Version, Format, AcceptLanguages , Disposition, CRS , Width, Height, Styles, Transparent, Bgcolor , Metadata, Options, Path`	Optional
GetPathMapInfo	The request is similar to the GetPath Map request except that instead of images + metadata it retrieves metadata only.	`Service, Request, Version, Format, AcceptLanguages,`	Optional

REQUEST	MEANING	PARAMETERS	OPTIONALITY
		`CRS, Width, Height, Metadata, Options, Path`	

[a] A path is defined as a one or more tracks, each track has a set of key frames and a track evaluation method. The AOI (area of interest) is referenced using a known coordinate reference system. A track may go over one or more spatiotemporally overlapping collections.

14

VIDEO SERVICE

14 VIDEO SERVICE

The Video Service is notionally identical to the IS. Instead of a flipbook of images, it delivers a video clip of those images. If the content format supports the addition of metadata, it is included. See Annex B.4 in Annex B.

14.1. Summary of all requests

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. See Annex A for a summary.	`Service, Request, Version, Format, CID, CRS, BBox, Width, Height, Styles, Transparent, Bgcolor, Time, Dup, Metadata, Options`	Mandatory
GetPath MapVideo	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**.	`Service, Request, Version, Format, CRS, Width, Height, Styles, Transparent, Bgcolor, Dup, Metadata, Method, Options, Path`	Optional

15

CONSIDERATIONS

15 CONSIDERATIONS

In case of ambiguity, a consistent outcome is required. A few use cases have been described below.

15.1. High latency networks

For high latency networks, it is better to get as much data with as few requests. When using an IS flipbook, performance degrades if each frame is a separate request. If a client buffers 10 frames at a time, using a single request, the client can request those frames using TIME (TIME=F1/F10/FS1). For low speed networks, request a video stream.

15.2. Temporal mismatch

When asking for data at a particular instant in time using an ISO time string, the requested time may not match actual time. If requested time does not match captured time, pick the lowest, highest or closest matching time, and stay consistent in the choice.

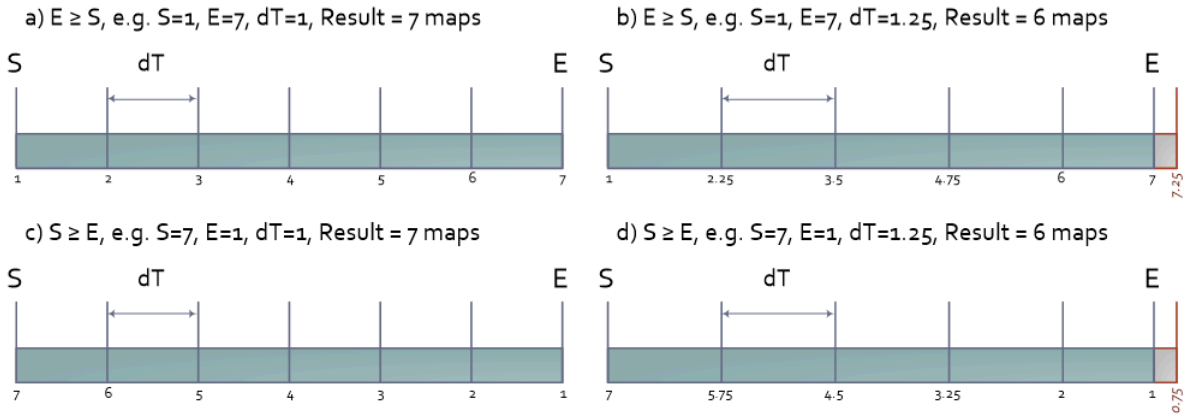


Figure 11 — Consistently resolve temporal ambiguity

15.3. Spatiotemporal overlap

In WMS, when overlapping multiple layers, the issue of sub-pixel overlap is common. Servers choose to round off to pixel boundaries and avoid compositing complexity. Servers round off to the lowest, highest or closest pixel. In either case, the error is consistent.

This choice avoids rendering artifacts and increases performance for the price of sub-pixel accuracy when overlapping maps. A similar choice has to be made when thinking of all dimensions simultaneously — X, Y, zoom and T. The choice is open to the server implementation so long as it is consistent.

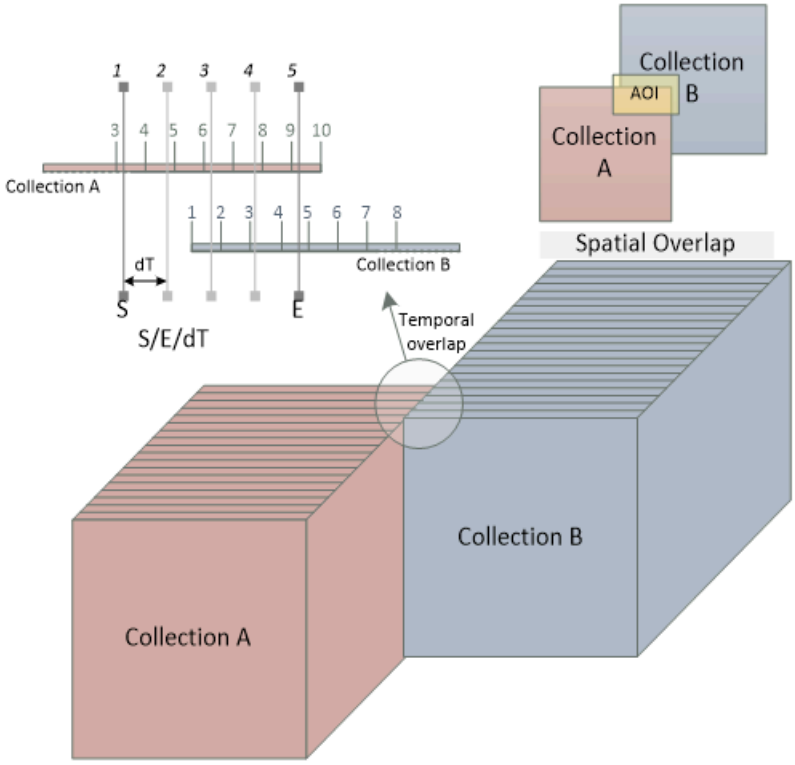


Figure 12 — Consistently resolve spatiotemporal ambiguity

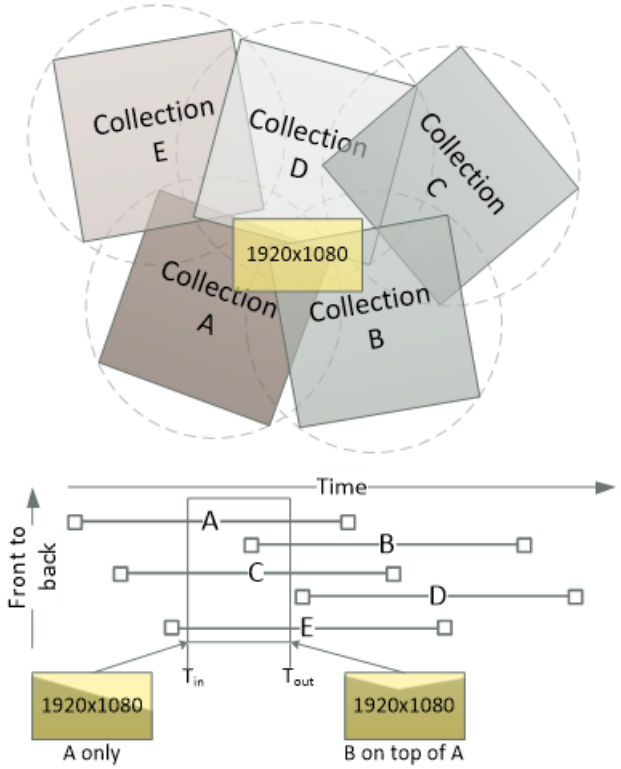


Figure 13 — Spatial overlap can change over time

15.4. Rendering maps along a pre-defined path

The requests **GetPathMap** and **GetPathMapVideo** draw heavily from the animation industry. A client describes a **path**. A **path** has one or more **tracks**. A **track** defines a curve with **entries**. Each **entry** is a **key-frame**. The key-frame specifies a bounding box and time within a set of collections. The rendering system on the server side first interpolates all points in-between the key frames to generate all points for all frames within the “clip”.

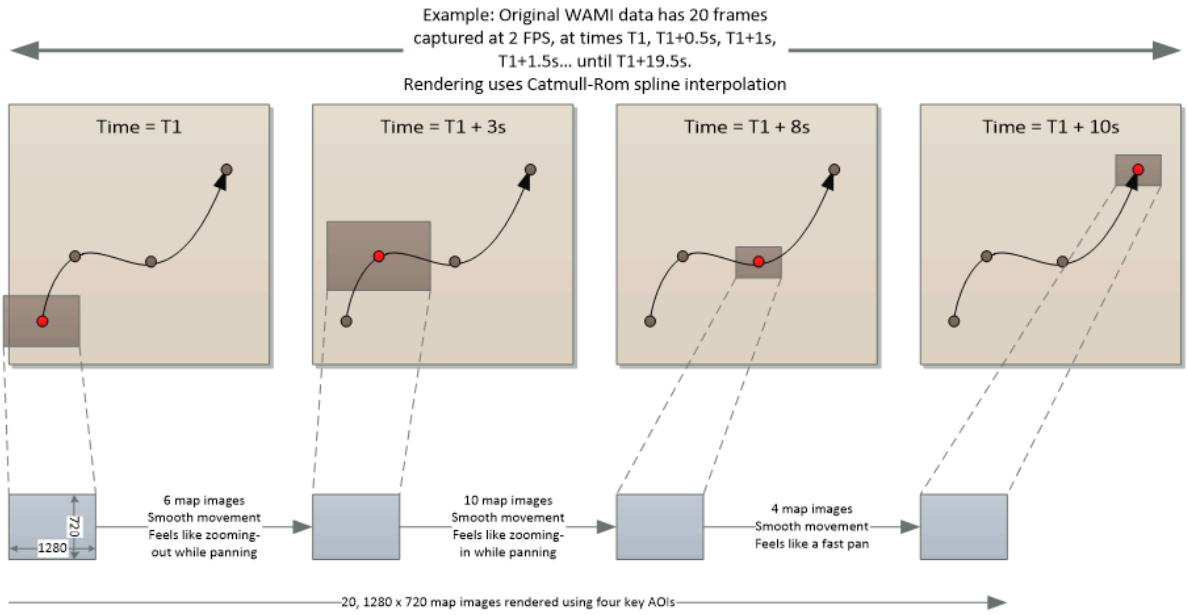


Figure 14 — Key-frame animation using GetPathMap and GetPathMapVideo

This is analogous to *in-betweening* in *key-frame animation*. Once points for all frames are generated (which is very fast), each point for each frame is merely a **GetMap** request. As each frame gets rendered it is returned to the consumer as part of a flipbook or video codestream. One request results in many frames, helping cater to networks with high latency.

15.5. Tile server

The metadata returned by a CS implementation can describe the ideal layout of a WAMI collection in terms of tiles of original map and R-sets, informing the client on an optimal way to request data. Each collection's CS metadata would provide image dimensions, number of R-sets, scale factor of each R-set, tile row and column count, tile dimensions, etc. The client could then make IS: GetMap requests on tile boundaries. Tile playback synchronization would be a client-side issue to tackle.

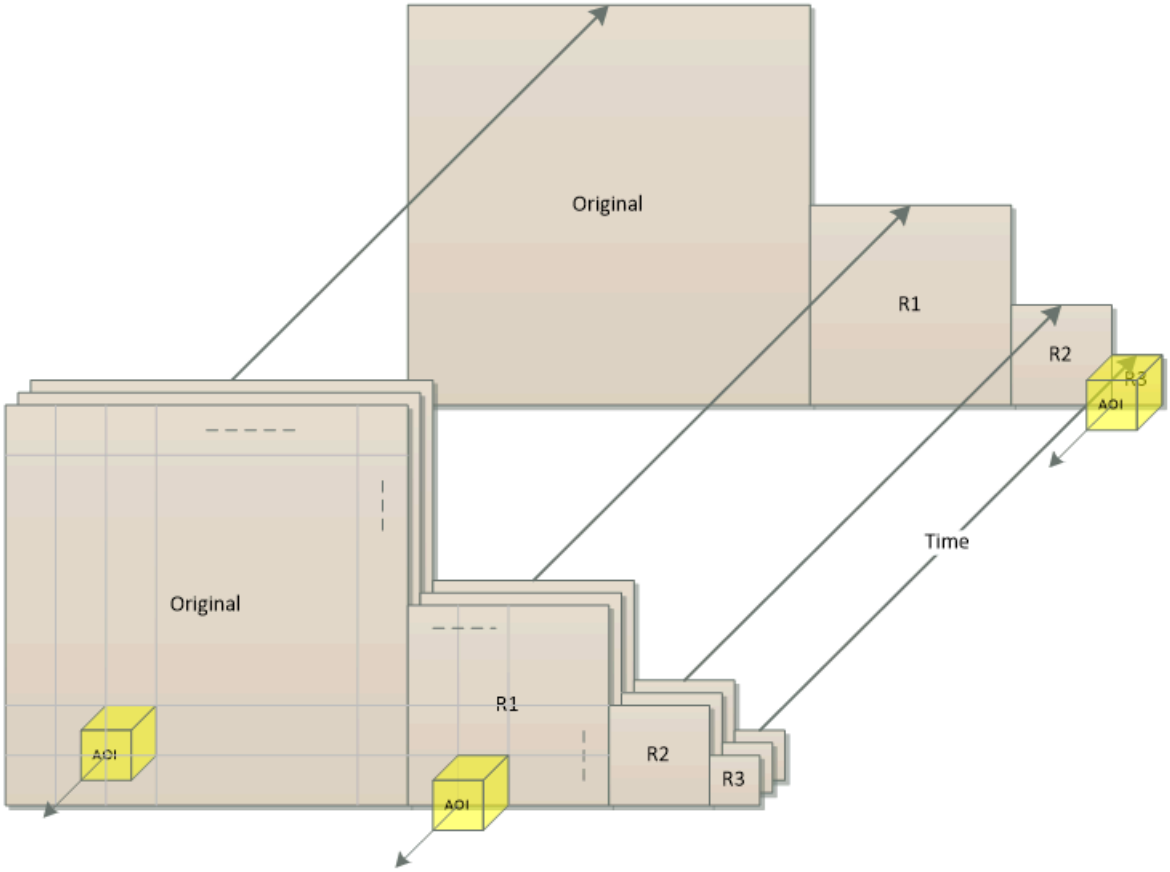


Figure 15 — Tile based dissemination

16

CONCLUSION

16 CONCLUSION

WAMI is a big data problem. OGC 12-032r2 is critical to effectively utilizing WAMI data in the www.For further questions, please contact the authors, preferably through the OGC. We welcome feedback and support to help make a commercially successful implementation of a global best practice.

A

ANNEX A (INFORMATIVE) REQUESTS SUMMARIZED

A ANNEX A (INFORMATIVE) REQUESTS SUMMARIZED

A.1. GetCapabilities

All services shall implement the GetCapabilities request. It tells the client what a service delivers.

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ 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=1.0.0,1.0.2,1.0.2`	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.

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
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.

A.2. GetHelp

This request is optional.

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ 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.

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
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.

A.3. CS:GetCollectionCount

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ 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 **Get CollectionCount**.
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 NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
			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).

A.4. CS:GetCollections

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ 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]**

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
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**

A.5. IS:GetMap

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ 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.

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
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.

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
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.

A.6. IS:GetMapInfo

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ 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.

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
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 or 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. 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.

A.7. IS:GetPathMap

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
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.

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
Path	`Path=URL EncodedXML`	Mandatory, one	Specifies the tracks to be rendered by the service. Shall be implemented by both client and server.

A.8. IS:GetPathMapInfo

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ 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.

A.9. VS:GetMapVideo

This request analogous to GetMap. Recommend reading the section on DUP in OGC 12-032r2.

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ 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.

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
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.

A.10. VS:GetPathMapVideo

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ 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

PARAMETER NAME	KVP EXAMPLE	OPTIONALITY/ MULTIPLICITY	DEFINITION, FORMAT AND USE
			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.
Path	`Path=URL Encoded XML`	Mandatory, one	Specifies the tracks to be rendered by the service. Shall be implemented by both client and server.

B

ANNEX B (INFORMATIVE) EXAMPLES

B ANNEX B (INFORMATIVE) EXAMPLES

B.1. GetCapabilities

An example of a **GetCapabilities** HTTP GET request and XML response from an actual server is as below. DNS names were changed.

http://example.com/wami/CS?service=CS&request=GetCapabilities&version=1.0.0&AcceptFormats=text/xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:Capabilities version="1.0.0" xmlns:ns2="http://www.w3.org/1999/xlink" xmlns:
ns1="http://www.opengis.net/ows/2.0" xmlns:ns3="http://www.pixia.com/wami">
<ns1:ServiceIdentification>
<ns1:Title>Collection Service</ns1:Title>
<ns1:ServiceType>CS</ns1:ServiceType>
<ns1:ServiceTypeVersion>1.0.0</ns1:ServiceTypeVersion>
</ns1:ServiceIdentification>
<ns1:ServiceProvider>
<ns1:ProviderName>Pixia Corp</ns1:ProviderName>
<ns1:ServiceContact/>
</ns1:ServiceProvider>
<ns1:OperationsMetadata>
<ns1:Operation name="GetCollectionCount">
<ns1:DCP>
<ns1:HTTP>
<ns1:Get ns2:type="simple" ns2:href="http://example.com/wami/CS"/>
</ns1:HTTP>
</ns1:DCP>
<ns1:Parameter name="Service">
<ns1:AllowedValues>
<ns1:Value>CS</ns1:Value>
</ns1:AllowedValues>
</ns1:Parameter>
<ns1:Parameter name="Request">
<ns1:AllowedValues>
<ns1:Value>GetCollectionCount</ns1:Value>
</ns1:AllowedValues>
</ns1:Parameter>
<ns1:Parameter name="Version">
<ns1:AllowedValues>
<ns1:Value>1.0.0</ns1:Value>
</ns1:AllowedValues>
</ns1:Parameter>
<ns1:Parameter name="Format">
```

```
<ns1:AllowedValues>
<ns1:Value>application/xml</ns1:Value>
<ns1:Value>application/json</ns1:Value>
</ns1:AllowedValues>
<ns1:DefaultValue>application/xml</ns1:DefaultValue>
</ns1:Parameter>
<ns1:Parameter name="AcceptLanguages">
<ns1:AllowedValues>
<ns1:Value>en-US</ns1:Value>
<ns1:Value>en</ns1:Value>
</ns1:AllowedValues>
<ns1:DefaultValue>en-US</ns1:DefaultValue>
</ns1:Parameter>
<ns1:Parameter name="NID">
<ns1:AnyValue/>
</ns1:Parameter>
<ns1:Parameter name="Depth">
<ns1:AllowedValues>
<ns1:Value>1</ns1:Value>
<ns1:Value>All</ns1:Value>
</ns1:AllowedValues>
<ns1:DefaultValue>All</ns1:DefaultValue>
</ns1:Parameter>
<ns1:Parameter name="Bbox">
<ns1:AnyValue/>
<ns1:Meaning>Bounding box to filter collections wholly outside the box.</Meaning>
</ns1:Parameter>
<ns1:Parameter name="CRS">
<ns1:AllowedValues>
<ns1:Value>EPSG:4326</Value>
<ns1:Value>EPSG:4269</Value>
<ns1:Value>EPSG:32645</Value>
</ns1:AllowedValues>
<ns1:Meaning>Coordinate reference system for BBox</Meaning>
</ns1:Parameter>
<ns1:Parameter name="Time">
<ns1:AnyValue/>
<ns1:Meaning>ISO 8601 time interval</Meaning>
</ns1:Parameter>
</ns1:Operation>
<ns1:Operation name="GetCollections">
<ns1:DCP>
<ns1:HTTP>
<ns1:Get ns2:type="simple" ns2:href="http://example.com/wami/CS"/>
</ns1:HTTP>
</ns1:DCP>
<ns1:Parameter name="Service">
<ns1:AllowedValues>
<ns1:Value>CS</ns1:Value>
</ns1:AllowedValues>
</ns1:Parameter>
<ns1:Parameter name="Request">
<ns1:AllowedValues>
<ns1:Value>GetCollections</ns1:Value>
</ns1:AllowedValues>
</ns1:Parameter>
<ns1:Parameter name="Version">
<ns1:AllowedValues>
<ns1:Value>1.0.0</ns1:Value>
</ns1:AllowedValues>
</ns1:Parameter>
<ns1:Parameter name="Format">
<ns1:AllowedValues>
```

```
<ns1:Value>application/xml</ns1:Value>
<ns1:Value>application/json</ns1:Value>
<ns1:Value>json</ns1:Value>
</ns1:AllowedValues>
<ns1:DefaultValue>application/xml</ns1:DefaultValue>
</ns1:Parameter>
<ns1:Parameter name="AcceptLanguages">
<ns1:AllowedValues>
<ns1:Value>en-US</ns1:Value>
<ns1:Value>en</ns1:Value>
</ns1:AllowedValues>
<ns1:DefaultValue>en-US</ns1:DefaultValue>
</ns1:Parameter>
<ns1:Parameter name="NID">
<ns1:AnyValue/>
</ns1:Parameter>
<ns1:Parameter name="Metadata">
<ns1:AllowedValues>
<ns1:Value>All</ns1:Value>
<ns1:Value>None</ns1:Value>
<ns1:Value>Collection</ns1:Value>
<ns1:Value>GeoBox</ns1:Value>
<ns1:Value>File</ns1:Value>
</ns1:AllowedValues>
<ns1:DefaultValue>None</ns1:DefaultValue>
</ns1:Parameter>
<ns1:Parameter name="Depth">
<ns1:AllowedValues>
<ns1:Value>0</ns1:Value>
<ns1:Value>1</ns1:Value>
<ns1:Value>All</ns1:Value>
</ns1:AllowedValues>
<ns1:DefaultValue>0</ns1:DefaultValue>
</ns1:Parameter>
<ns1:Parameter name="Bbox">
<ns1:AnyValue/>
<ns1:Meaning>Bounding box to filter collections wholly outside the box.</Meaning>
</ns1:Parameter>
<ns1:Parameter name="CRS">
<ns1:AllowedValues>
<ns1:Value>EPSG:4326</Value>
<ns1:Value>EPSG:4269</Value>
<ns1:Value>EPSG:32645</Value>
</ns1:AllowedValues>
<ns1:Meaning>Coordinate reference system for BBox</Meaning>
</ns1:Parameter>
<ns1:Parameter name="Time">
<ns1:AnyValue/>
<ns1:Meaning>ISO 8601 time interval</Meaning>
</ns1:Parameter>
</ns1:Operation>
</ns1:OperationsMetadata>
<ns3:Language>en-US</ns3:Language>
<ns3:Language>en</ns3:Language>
</ns3:Capabilities>
```

Listing

B.2. CS

1. It is good to know in advance if a service is serving up a lot of data. This request gives you the number of nodes, edges, and collections being served. Request and response as below.

   http://example.com/CS?service=CS&request=GetCollectionCount&version=1.0.2&format=text/xml

```
<wami:CS_CollectionCount xmlns:wami="http://www.pixia.com/wami"
xmlns:ns2="http://www.opengis.net/ows/2.0" xmlns:ns3="http://www.w3.
org/1999/xlink" root="true" depth="0" NID="Pixia.Root" childNodes=
"3" totalNodes="7" collections="2" edgeDepth="4" lang="en" version=
"1.0.0"/>
```

Listing

2. If there is a lot of data being served, you can get just the top level list of nodes as follows:

   http://example.com/CS?service=CS&request=GetCollections&version=1.0.2&format=text/xml&Depth=1

```
<wami:CS_Collections xmlns:wami="http://www.pixia.com/wami/v101"
xmlns:ns2="http://www.opengis.net/ows/2.0" xmlns:ns3="http://www.
w3.org/1999/xlink" updateSequence="1333460158002" lang="en" version=
"1.0.2">
  <wami:Node id="Pixia.Root" name="Master Node" NID="Pixia.Root"
updateSequence="1333460158002">
    <wami:Node id="71cf67b97c6a4045ae9ce27b7ca7f475" name="dir1"
 parentNID="Pixia.Root" NID="71cf67b97c6a4045ae9ce27b7ca7f475"
updateSequence="1333460155579"/>
    <wami:Node id="06c5f1c2e641428ebeed061213b69a00" name="ch2009-
07-18-v25" parentNID="Pixia.Root" NID="06c5f1c2e641428ebeed0612
13b69a00" CID="ece48e24ad2b41eab709984e272cbbc2" updateSequence=
"1333399468333">
      <wami:Service name="IS">
        <wami:Request get="true" ns3:type="simple" ns3:href="http://
example.com/wami-soa-server/wami/IS"/>
      </wami:Service>
      <wami:Service name="VS">
        <wami:Request get="true" ns3:type="simple" ns3:href="http://
example.com/wami-soa-server/wami/VS"/>
      </wami:Service>
    </wami:Node>
    <wami:Node id="94484e794b59494c9226cc03c6ee9ff7" name="dir3"
 parentNID="Pixia.Root" NID="94484e794b59494c9226cc03c6ee9ff7"
updateSequence="1333460158002"/>
  </wami:Node>
</wami:CS_Collections>
```

Listing

3. Starting from a specific node (identified by **NID**) in the tree, get information about its child-nodes.

http://example.com/CS?service=CS&request=GetCollections&version=1.0.2&format=xml&Depth=1&nid=71cf67b97c6a4045ae9ce27b7ca7f475

```
<wami:CS_Collections xmlns:wami="http://www.pixia.com/wami/v101"
xmlns:ns2="http://www.opengis.net/ows/2.0" xmlns:ns3="http://www.
w3.org/1999/xlink" updateSequence="1333460155579" lang="en" version=
"1.0.2">
  <wami:Parent NID="Pixia.Root">
    <wami:Service name="CS"/>
  </wami:Parent>
  <wami:Node id="71cf67b97c6a4045ae9ce27b7ca7f475" name="dir1"
 parentNID="Pixia.Root" NID="71cf67b97c6a4045ae9ce27b7ca7f475"
updateSequence="1333460155579">
    <wami:Node id="c7c2755546e041928c2b324e3949207c" name="dir2"
parentNID="71cf67b97c6a4045ae9ce27b7ca7f475" NID="c7c2755546e041928c2
b324e3949207c" updateSequence="1333460155579"/>
  </wami:Node>
</wami:CS_Collections>
```

Listing

4. Get metadata about a specific WAMI collection, which is a leaf-node in the tree.

http://example.com/CS?service=CS&request=GetCollections&version=1.0.2&format=xml&Depth=1&ece48e24ad2b41eab709984e272cbbc2&metadata=All

```
<wami:CS_Collections xmlns:wami="http://www.pixia.com/wami/v101"
xmlns:ns2="http://www.opengis.net/ows/2.0" xmlns:ns3="http://www.
w3.org/1999/xlink" updateSequence="1333460158002" lang="en" version=
"1.0.2">
  <wami:Node id="Pixia.Root" name="Master Node" NID="Pixia.Root"
updateSequence="1333460158002">
    <wami:Node id="71cf67b97c6a4045ae9ce27b7ca7f475" name="dir1"
 parentNID="Pixia.Root" NID="71cf67b97c6a4045ae9ce27b7ca7f475"
updateSequence="1333460155579"/>
    <wami:Node id="06c5f1c2e641428ebeed061213b69a00" name="ch2009-
07-18-v25" parentNID="Pixia.Root" NID="06c5f1c2e641428ebeed0612
13b69a00" CID="ece48e24ad2b41eab709984e272cbbc2" updateSequence=
"1333399468333">
      <wami:Service name="IS">
        <wami:Request get="true" ns3:type="simple" ns3:href="http://
example.com/wami-soa-server/wami/IS"/>
      </wami:Service>
      <wami:Service name="VS">
        <wami:Request get="true" ns3:type="simple" ns3:href="http://
example.com/wami-soa-server/wami/VS"/>
      </wami:Service>
      <wami:Metadata>
        <wami:Collection startFrame="0" endFrame="3959" frameCount=
"3960" startTime="2009-07-18T18:45:25.27425Z" endTime="2009-07-
18T19:22:32.62225Z" timeSpan="2227.348" frameInterval="0.562461"
frameJitter="0.280668039" live="false"/>
        <wami:GeoBox nativeCRS="EPSG:32611">
          <wami:BoundingBox crs="EPSG:32611" minx="434544.0" miny=
"3940726.999682188" maxx="442542.00031781197" maxy="3948725.0" resx=
"0.6000000238418579" resy="0.6000000238418579"/>
          <wami:BoundingBox crs="EPSG:4326" minx="-
117.72334408548379" miny="35.60863513139489" maxx=
"-117.63439433102808" maxy="35.68024593060201" resx=
"6.6728998091306546E-6" resy="5.372152978778811E-6"/>
        </wami:GeoBox>
```

```
        <wami:File fileName="/virtual/tmp/ch2009-07-18-v25.nmv"
modifyTime="2011-05-05T21:26:15.000Z" pixelWidth="13330" pixelHeight=
"13330" bands="1" bitsPerBand="8" bandDataType="u"/>
      </wami:Metadata>
    </wami:Node>
    <wami:Node id="94484e794b59494c9226cc03c6ee9ff7" name="dir3"
 parentNID="Pixia.Root" NID="94484e794b59494c9226cc03c6ee9ff7"
updateSequence="1333460158002"/>
  </wami:Node>
</wami:CS_Collections>
```

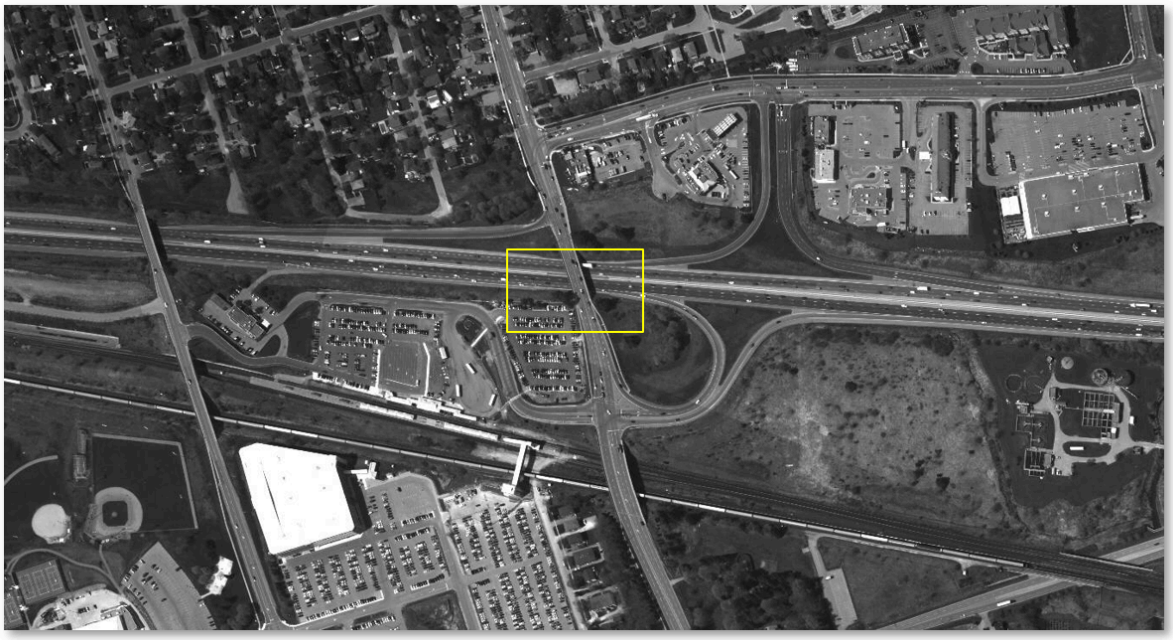
Listing

B.3. IS

All the OPTIONS in the requests below are specific to the server implementation. They were communicated to the client through a prior GetCapabilities request. All the images below are courtesy of PV Labs, Inc.

1. This request resulted in the following zoomed-out image.

   http://example.com/path/wami/IS?CID=XXXXXX&Request=GetMap&Service=IS&Version=1.0.0&CRS=EPSG:4326&Options.jpeg_quality=70&Format=image/jpeg&Exceptions=IMAGE&BBox=-78.94398740364547,43.862939714093955,-78.92885536723168,43.869013314093955&Width=1356&Height=730&Time=F1



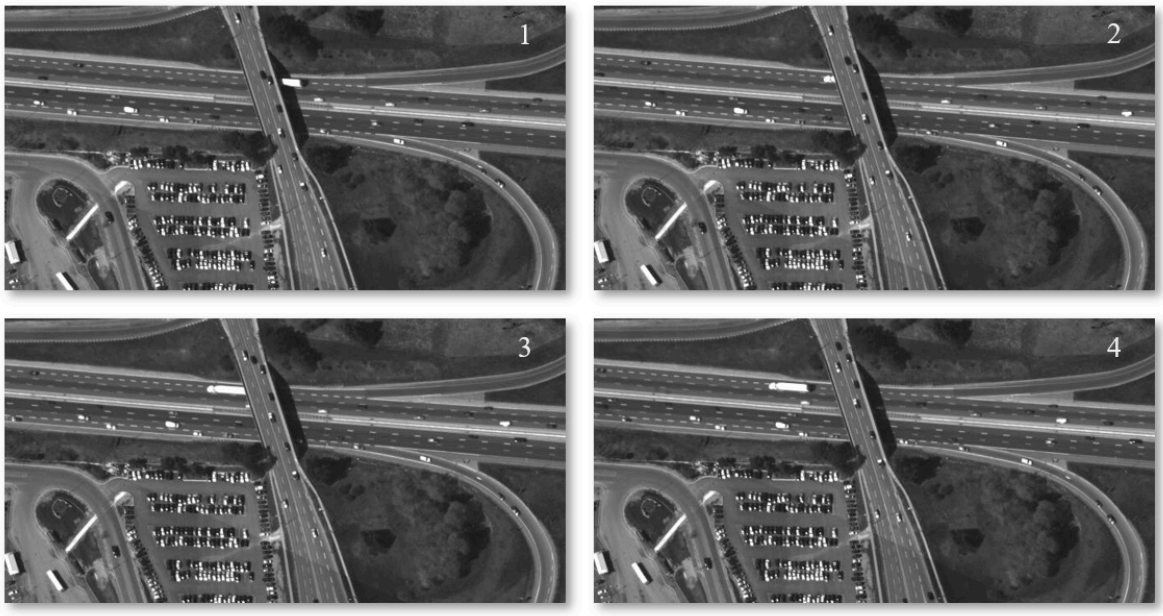
2. The client zoomed into the bridge at the center of the image.

   http://example.com/path/wami/IS?CID=XXXXXX&Request=GetMap&Service=IS&Version=1.0.0&CRS=EPSG:4326&Options.jpeg_quality=70&Format=image/

jpeg&Exceptions=IMAGE&BBox=-78.93736713771443,43.86579815409395,-78.9354756331627,43.86651263409395&Width=1356&Height=687&Time=F1

3. This request results in a single MIME multi-part response comprising of four images of the selected area of interest from frames 1, 2, 3 and 4. The Chrome browser quickly flips through these JPEG images. The four images are as shown below.

http://example.com/path/wami/IS?CID=XXXXXX&Request=GetMap&Service=IS&Version=1.0.0&CRS=EPSG:4326&Options.jpeg_quality=70&Format=image/jpeg&Exceptions=IMAGE&BBox=-78.93831288999029,43.86544091409395,-78.93452988088684,43.86686987409395&Width=678&Height=343&Time=F1/F4&Disposition=replace



http://example.com/path/wami/IS?CID=XXXXXX&Request=GetMap&Service=IS&Version=1.0.0&CRS=EPSG:4326&Options.jpeg_quality=70&Format=image/jpeg&Exceptions=IMAGE&BBox=-78.94398740364547,43.862939714093955,-78.92885536723168,43.869013314093955&Width=1356&Height=730&Time=F1

4. This request gets a client a single image of an AOI set using a bounding box at frame 101 from a collection in JPEG format (assuming the server supports JPEG).

http://example.com/CS?CID=ece48e24ad2b41eab709984e272cbbc2&Request=GetMap&Service=IS&Version=1.0.0&CRS=EPSG:4326&Options.jpeg.quality=50&Format=image/jpeg&Exceptions=IMAGE&BBox=-78.94355126506584,43.85916706820535,-78.93842759427986,43.86007044647508&Width=1674&Height=400&Time=F101

5. The same request delivers the same image in some other format such as GeoTIFF, in a compressed unsigned 16-bit format. Optional advanced features need to be provided by the server.

http://example.com/CS?CID=ece48e24ad2b41eab709984e272cbbc2&Request=GetMap&Service=IS&Version=1.0.0&CRS=EPSG:4326&Options.tif.band.datatype=u16&Options.tif.compress=gzip&Options.tif.type=geotiff&Format=image/tif&Exceptions=IMAGE&BBox=-78.94355126506584,43.85916706820535,-78.93842759427986,43.86007044647508&Width=1674&Height=400&Time=F101

6. Instead of asking for one frame at a time, a smart client in a high latency high bandwidth environment could ask for say 10 frames at a time, interlaced with metadata.

http://example.com/CS?CID=ece48e24ad2b41eab709984e272cbbc2&Request=GetMap&Service=IS&Version=1.0.0&CRS=EPSG:4326&Options.jpeg.quality=50&Format=image/jpeg&Exceptions=IMAGE&BBox=-78.94355126506584,

43.85916706820535,-78.93842759427986,43.86007044647508&Width=1674&Height=400&Time=F101/F110/FS1&disposition=replace&metadata=All

This will respond with a MIME multi-part message, starting with an XML manifest document providing a list of what follows, followed by a JPEG image for frame 101, followed by metadata for frame 101, followed by a JPEG image for frame 102, followed by metadata for frame 102, until frame 110. Just the XML manifest would look like:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<ns3:IS_Map xmlns:ns2="http://www.w3.org/1999/xlink" xmlns:ns1="http://www.
opengis.net/ows/2.0" xmlns:ns3="http://www.pixia.com/wami">
  <ns3:Reference imageReference="image0" metadataReference="metadata0"/>
  <ns3:Reference imageReference="image1" metadataReference="metadata1"/>
  <ns3:Reference imageReference="image2" metadataReference="metadata2"/>
  ...
</ns3:IS_Map>
```

Listing

B.4. VS

As soon as this request is received by the server, it starts to stream an MPEG2 transport stream that embeds MISB compliant KLV metadata. All the OPTIONS in this request are specific to the server. They were communicated to the client through a prior GetCapabilities request.

http://example.com/VS?CID=XXXXXXX&Version=1.0.0&CRS=EPSG%3A4326&Request=GetMapVideo&Service=VS&Exceptions=XML&Time=F0%2FF4112&BBox=-78.94355126506584%2C43.85916706820535%2C-78.93842759427986%2C43.86007044647508&Height=400&Width=1674&FORMAT=video/mpeg2&options.mpeg2.codec=mpeg2&options.mpeg2.stream=transport&options.mpeg2.kbps=3000&options.mpeg2.fps=30&Dup=0&options.mpeg2.gop=15&options.mpeg2.direction=forward&METADATA=All