Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <2020-07-28>

External identifier of this OGC® document: http://www.opengis.net/doc/EG/ogcapi-common/1.0

Internal reference number of this OGC® document: 20-071

Version: n.n

Category: OGC® Engineering Guidance

Editor: Charles Heazel

OGC API-Common Users Guide

Warning

This document defines an the OGC Best Practices on the implementation and use of the OGC API-Common standard. This document is not an OGC Standard and may not be referred to as an OGC Standard. It is subject to change without notice. However, this document is an official position of the OGC membership on this particular technology topic.

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

Document type:	OGC® Engineering Guidance
Document subtype:	
Document stage:	Draft
Document language:	English

Table of Contents

i. Abstract

<Insert Abstract Text here>

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, <tags separated by commas>

iii. Preface

NOTE

Insert Preface Text here. Give OGC specific commentary: describe the technical content, reason for document, history of the document and precursors, and plans for future work. > 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.

iv. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

Organization name(s)

v. Submitters

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

Name Affiliation

Chapter 1. Scope

NOTE Insert Scope text here. Give the subject of the document and the aspects of that scope covered by the document.

Chapter 2. Conformance

This Best Practice defines XXXX.

Requirements for N target types are considered: * AAAA * BBBB

Conformance with this Best Practice shall be checked using all the relevant tests specified in Annex A (normative) of this document.

In order to conform to this OGC® Best Practice, a software implementation shall choose to implement: * Any one of the conformance levels specified in Annex A (normative). * Any one of the Distributed Computing Platform profiles specified in Annexes TBD through TBD (normative).

All requirements-classes and conformance-classes described in this document are owned by the document(s) identified.

Chapter 3. References

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

NOTE

Insert References here. If there are no references, state “There are no normative references”.

References are to follow the Springer LNCS style, with the exception that optional information may be appended to references: DOIs are added after the date and web resource references may include an access date at the end of the reference in parentheses. See examples from Springer and OGC below.

Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)

May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)

Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)

Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)

Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)

National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov

ISO / TC 211: ISO 19115-1:2014 Geographic information — Metadata — Part 1: Fundamentals (2014)

ISO / TC 211: ISO 19157:2013 Geographic information — Data quality (2013)

ISO / TC 211: ISO 19139:2007 Geographic information — Metadata — XML schema implementation (2007)

ISO / TC 211: ISO 19115-3: Geographic information — Metadata — Part 3: XML schemas (2016)

OGC: OGC 15-097 OGC Geospatial User Feedback Standard. Conceptual Model (2016)

OGC: OGC 12-019, OGC City Geography Markup Language (CityGML) Encoding Standard (2012)

OGC: OGC 14-005r3, OGC IndoorGML (2014)

Chapter 4. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this Best Practice.

For the purposes of this document, the following additional terms and definitions apply.

4.1. term name

text of the definition

Chapter 5. Conventions

This sections provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. Identifiers

The normative provisions in this document are denoted by the URI

http://www.opengis.net/spec/{standard}/{m.n}

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

Chapter 6. OGC API Philosophy

The Best Practices document contains information useful to a developer of user of OGC API-Common, but not required by the standard.

Chapter 7. Modular APIs

This section of the Best Practice discusses OGC principals and practices regarging modular APIs.

Chapter 8. Navigation

Navigating the API

This section of the Best Practice addresses the different ways to navigate an API, the features of OGC Web APIs to facilitate navigation, and the impact of different navigation techniques on the client.

Chapter 9. Architecture

The following "architecture" captures the core concepts which inform all OGC API standards.

9.1. Datasets

Web APIs implement a Resource Oriented approach to Web-based distributed computing. A coherent set of APIs must be based on a common understading of the Resources to be shared. For OGC APIs, those resources would be identified using an abstract concept of a geospatial dataset. Such a dataset would have the following characteristics:

- A dataset can be a vector feature collection, a coverage/imagery, or a collection of datasets (therefore potentially a mix of any of these things)
- A dataset has associated metadata, including some essential information:
  - Information about what type of data set it is (vector features (and what type of vector features if limited to one type, e.g. polygons,lines, points), coverages (values) / imagery (pixels), or sub-datasets-- more than one of those things)
  - A textual identifier (e.g. which figures in the resource path)
  - A title (short name / description)
  - Access point for the dataset (could be hosted locally or remote)
  - Geospatial & temporal extent
  - Resolution/scale
  - Units/Range/Bit-Depth/Channels/Dimensions etc. for imagery/coverages
  - A description of queryables, if applicable
- Keywords/Tags, and longer descriptions are also a commonly useful piece of metadata information
- Any other ISO 19115 metadata fields can also be associated with the dataset, but are nowhere near as essential to discovering and using geospatial data as those mentioned above. Meta data containing at minimum those essential elements can always be retrieved in ISO 19115 and potentially other formats.

9.2. Processes

Processes take one or more datasets as input, add parameters, and produce one or more datasets as output. This ties the processes together with the data delivery services on both ends.

I suggested so far 3 kinds of processes which can all run on a server where the data lives:

1. The complex process built as a container or executable, as typical of WPS
2. Process description languages such as WCPS
3. Pre-defined named processes such as 'vectorization', 'buffering', 'rasterization' or 'rendering of a styled map'

All of these kinds of processes could share aspects such as taking in an OGC API dataset as input and their output being usable as an OGC API dataset, for direct access and/or asynchronous delivery, and support multiple data partitioning/access mechanisms, estimates/billing elements, and so on.

9.3. Server-side rendering

Highlighting here that the rendering of a styled maps based on multiple source data sets and a style as a parameter, outputting a styled 'rendered map' imagery dataset as a result, fits perfectly well the description of a process (3).

9.4. Data partitioning

For all of data access / exchange mechanisms, e.g. to retrieve coverages/vector features directly, for a process to access its input, or to retrieve the output from a process, or throughout a daisy chain of processes, there are a variety of ways in which data can be partitioned for efficient access. The most efficient way most likely depends on the overall workflow and the implementations on both ends. I am suggesting most of the OGC API should be agnostic of this and support many such ways, so that both ends of such connection or a workflow manager can negotiate the best approach. Examples of different ways to partition/access data include:

- Bounding boxes
- Tiles
- n-dimensional sub-setting
- DGGS cells
- Single point value

This is the overall diagram of these ideas that I presented a couple weeks back at the OWS Common telecon and got the chance to explain in more details in person to some of you:

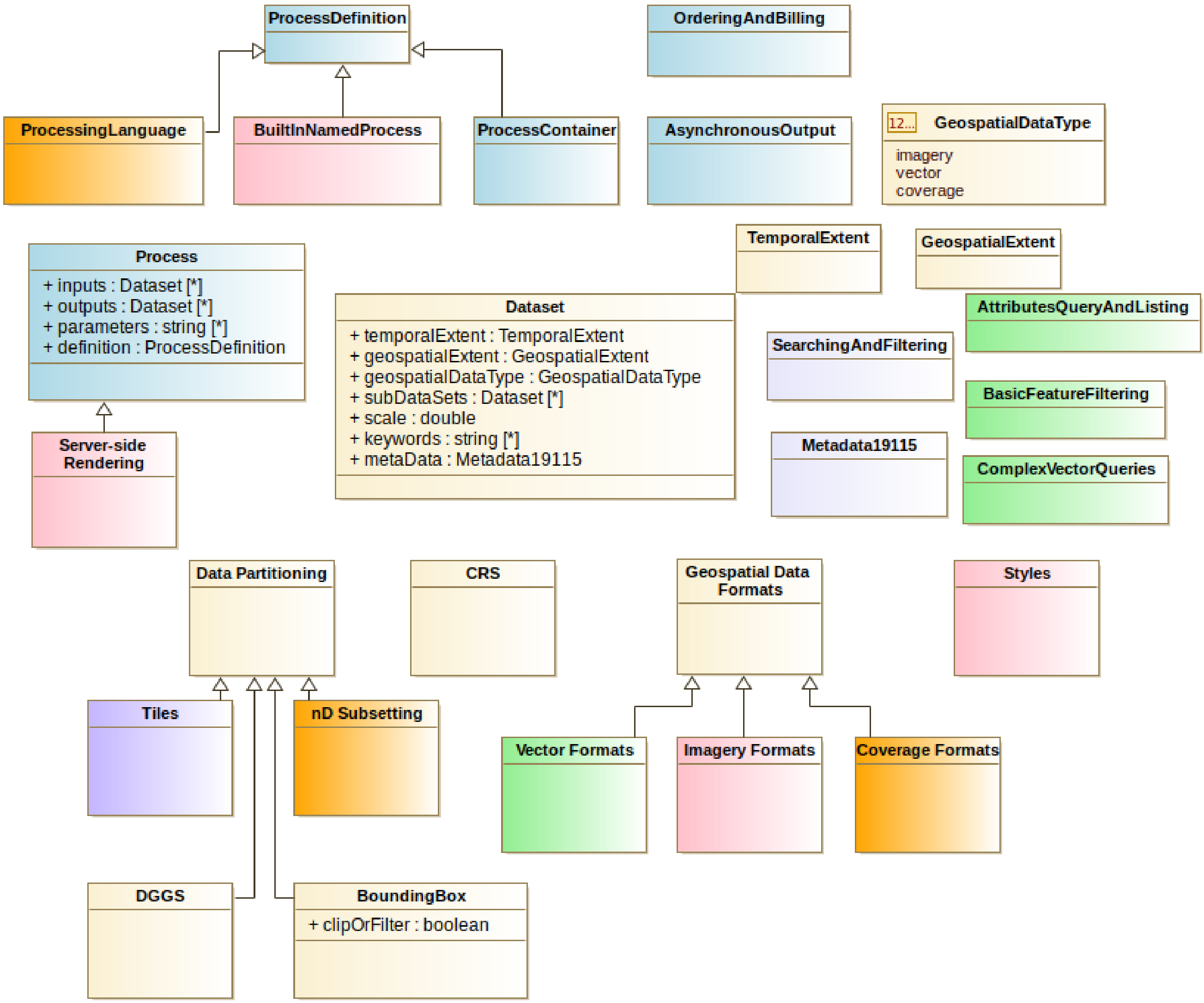

Figure 1. Dataset Concept

You might notice that this is color-coded based on where these API blocks originated from in classic services:

- Orange: Coverages (WCS)
- Red: Server-side rendering (WMS)
- Blue: Processes (WPS)
- Green: Vector features (WFS)
- Very light blue: Catalogs (CSW)
- Violet: Tiles (WMTS)

9.5. Resource Paths

The resources exposed through OGC APIs are accessed through standardized URL templates. These "paths" and their associated resources are described in Table 1. This table documents the paths and resources as of December 2019. This information will be updated as these standards mature.

Table 1. OGC Dataset Paths

Path Template	Resource
Common	
/collections	Metadata describing the spatial collections available from this API.
/collections/{collectionId}	Metadata describing the collection which has the unique identifier `{collectionId}`
Features	
/collections/{collectionId}	The Feature Collection resource identified by the `{collectionId}` parameter.
/collections/{collectionId}/items	The individual Features in a Feature Collection
Coverages	
/collections/{collectionId}/coverage	A general description of the coverage identified by {collectionId} including the coverage’s envelope.
/collections/{collectionid}/coverage/description	returns the whole coverage description consisting of domainset, rangetype, and metadata (but not the rangeset)
/collections/{collectionid}/coverage/domainset	returns the coverage’s domain set definition
/collections/{collectionid}/coverage/rangetype	returns the coverage’s range type information (i.e., a description of the data semantics)
/collections/{collectionid}/coverage/metadata	returns the coverage’s metadata (may be empty)
/collections/{collectionid}/coverage/rangeset	returns the coverage’s range set, i.e., the actual values in the coverage’s Native Format (see format encoding for ways to retrieve inspecific formats)
/collections/{collectionid}/coverage/all	returns all of the above namely the coverage’s domainset, rangetype, meatadata, and rangeset comparable to a GetCoverage response
Maps and Styles	
Note: A map is associated with a resource. {resource} is a place holder for a path segment appropriate for a resource type.	
/collections/{collectionId}/{resource}/map	A data structure with specific information necessary to get a fragment of the map representing the resource collection.
/collections/{collectionId}/{resource}/map/{styleId}	A map representing the geospatial resource identified by /collections/{collectionId}/{resource}.
Tiles	
Note: A tile is associated with a resource. {resource} is a place holder for a path segment appropriate for a resource type.	

Path Template	Resource
/collections/{collectionId}/{resource}/tile/{styleId}/{tileMatrixSetId}/{tileMatrixId}/{tileRow}/{tileCol}	Tile representation of real-world elements at a given resolution restricted by the selected Tile Matrix Set. {styleId} is optional.
/collections/{collectionId}/{resource}/tiles	An enumeration of the Tiles and Styles that are available at some TileMatrixSetId
/tileMatrixSet/{tileMatrixSetId}	A description of the TileMatrixSet identified by the {tileMatrixSetId} identifier.
Processes	
/process	Lists the processes this API offers.
/processes/{process-id}	Returns a detailed description of a process.
/processes/{process-id}/jobs	Returns the running and finished jobs for a process (GET), Executes a process, i.e. creates a new job. Inputs and outputs will have to be specified in a JSON document that needs to be send in the POST body. (POST)
/processes/{process-id}/jobs/{job-id}	Returns the status of a job of a process.
/processes/{process-id}/jobs/{job-id}/results	Returns the result of a job of a process.
Records (Catalog)	
/collections/{collectionId}/tbd	TBD

Chapter 10. API Description

Using and Abusing ApenAPI

OpenAPI is a useful tool for documenting the design of an API. In addition, numerous tool vendors have produced products to assist in the use of OpenAPI documents. These go so far as to generate stub-code for the use of API and client developers.

However like so many other good tools, OpenAPI was not designed for the size and complexity that we are seeing in Geospatial APIs. This section discusses various features of the OpenAPI specification that can be abused to provide a scalable API Description.

10.1. Web APIs

The following principles are derived from Fielding 2000.

The central feature that distinquished Web APIs is their emphasis on a uniform interface. In particular, their nearly exclusive use of the HTTP protol and the associated operators (GET, PUT, etc.).

In order to achieve a usable uniform interface, Web APIs are subject to four constraints:

1. identification of resources,
2. manipulation of resources through representations,
3. self-descriptive messages; and,
4. hypermedia as the engine of application state.

10.1.1. Resources

The key abstraction of information is a resource. Any information that can be named can be a resource: a document or image, a temporal service (e.g. “today’s weather in Los Angeles”), a collection of other resources, a non-virtual object (e.g. a person), and so on. In other words, any concept that might be the target of an author’s hypertext reference must fit within the definition of a resource. A resource is a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time.

10.1.2. Uniform Resource Identifiers

REST uses a resource identifier to identify the particular resource involved in an interaction between components. REST connectors provide a generic interface for accessing and manipulating the value set of a resource, regardless of how the membership function is defined or the type of software that is handling the request. The naming authority that assigned the resource identifier, making it possible to reference the resource, is responsible for maintaining the semantic validity of the mapping over time (i.e., ensuring that the membership function does not change).

10.1.3. Representations

REST components perform actions on a resource by using a representation to capture the current or intended state of that resource and transferring that representation between components. A representation is a sequence of bytes, plus representation metadata to describe those bytes.

A representation consists of data, metadata describing the data, and, on occasion, metadata to describe the metadata (usually for the purpose of verifying message integrity).

messages may include both representation metadata and resource metadata: information about the resource that is not specific to the supplied representation.

Depending on the message control data, a given representation may indicate the current state of the requested resource, the desired state for the requested resource, or the value of some other resource, such as a representation of the input data within a client's query form, or a representation of some error condition for a response. For example, remote authoring of a resource requires that the author send a representation to the server, thus establishing a value for that resource that can be retrieved by later requests. If the value set of a resource at a given time consists of multiple representations, content negotiation may be used to select the best representation for inclusion in a given message.

The data format of a representation is known as a media type.

The messages exchanged are self-describing. No a-priori knowledge of the Web API should be required.

The central feature that distinguishes the REST architectural style from other network based styles is its emphasis on a uniform interface between components

REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; selfdescriptive messages; and, hypermedia as the engine of application state.

10.2. Collections

Geospatial data is rarely considered as a single entity. Feature Collections, Coverages, Data Sets, they are all aggregations of Spatial or Temporal Things. It stands to reason that an OGC Web API would also expose its' holdings as aggregates of spatial/temporal resources.

Therefore, OGC APIs organize geospatial data into `collections`. The role of the API-Common GeoData Standard is to define the operations to discover and navigate the geospatial data `collections` exposed through an OGC API.

The term `collection` means different things to different people. For the purpose of this standard, a `collection` is defined as:

- Collection: A geospatial data resource that may be available as one or more sub-resource distributions that conform to one or more OGC API standards.

/collections = a set of metadata about a collection of geospatial* data

/collections/{collectionId} = metadata about a specific collection of geospatial* data with links to distribution

/collections/{collectionId}/{} = *a specific distribution of geospatial* data

OGC Web API standards can extend this definition to address the specific properties of the resources they describe.

10.3. URIs and Relationships

Resources exposed through an OGC API may be accessed through a Universal Resource Identifier (URI). URIs are composed of three sections:

- Service Offering: The service endpoint (subsequently referred to as Base URI or {root})
- Access Paths: Unique paths to Resources
- Query: Parameters to adjust the representation of a Resource or Resources like encoding format or subsetting

Some resources are also accessible through links on previously accessed resources. Unique relation types are used for each resource.

Table 2 summarizes the access paths and relation types defined in this standard.

Table 2. Coverage API Paths

Path Template	Relation	Resource
Common		
{root}/	none	Landing page
{root}/api	`service-desc` or `service-doc`	API Description (optional)
{root}/conformance	`conformance`	Conformance Classes
{root}/collections	`data`	Metadata describing the spatial collections (coverages) available from this API.
{root}/collections/{coverageid}		Metadata describing the coverge which has the unique identifier `{coverageid}`
Coverages		
{root}/collections/{coverageid}/coverage	`items`	A general description of the coverage identified by {coverageid} including the coverage’s envelope.
{root}/collections/{coverageid}/coverage/description	none	returns the whole coverage description consisting of domainset, rangetype, and metadata (but not the rangeset)

Path Template	Relation	Resource
{root}/collections/{coverageid}/coverage/domainset	none	returns the coverage's domain set definition
{root}/collections/{coverageid}/coverage/rangetype	none	returns the coverage's range type information (i.e., a description of the data semantics)
{root}/collections/{coverageid}/coverage/metadata	none	returns the coverage's metadata (may be empty)
{root}/collections/{coverageid}/coverage/rangeset	none	returns the coverage's range set, i.e., the actual values in the coverage's Native Format (see format encoding for ways to retrieve in specific formats)
{root}/collections/{coverageid}/coverage/all	none	returns all of the above namely the coverage's domainset, rangetype, metadata, and rangeset comparable to a GetCoverage response

Where:

- {root} = Base URI for the API server
- {coverageid} = an identifier for a specific coverage (collection)

Chapter 11. Clause containing normative material

Paragraph

11.1. Requirement Class A or Requirement A Example

Paragraph – intro text for the requirement class.

Use the following table for Requirements Classes.

Requirements Class	
http://www.opengis.net/spec/ABCD/m.n/req/req-class-a	
Target type	Token
Dependency	http://www.example.org/req/blah
Dependency	urn:iso:ts:iso:19139:clause:6
Requirement 1	http://www.opengis.net/spec/ABCD/m.n/req/req-class-a/req-name-1 requirement description
Requirement 2	http://www.opengis.net/spec/ABCD/m.n/req/req-class-a/req-name-2 requirement description
Requirement 3	http://www.opengis.net/spec/ABCD/m.n/req/req-class-a/req-name-3 requirement description

11.1.1. Requirement 1

Paragraph - intro text for the requirement.

Use the following table for Requirements, number sequentially.

Requirement 1	/req/req-class-a/req-name-1 Requirement 'shall' statement

Dictionary tables for requirements can be added as necessary. Modify the following example as needed.

Names	**Definition**	**Data types and values**	**Multiplicity and use**
name 1	definition of name 1	float	One or more (mandatory)

Names	Definition	Data types and values	Multiplicity and use
name 2	definition of name 2	character string type, not empty	Zero or one (optional)
name 3	definition of name 3	GML:: Point PropertyType	One (mandatory)

Annex A: Conformance Class Abstract Test Suite (Normative)

NOTE Ensure that there is a conformance class for each requirements class and a test for each requirement (identified by requirement name and number)

A.1. Conformance Class A

A.1.1. Requirement 1

Test id:	/conf/conf-class-a/req-name-1
Requirement:	/req/req-class-a/req-name-1
Test purpose:	Verify that…
Test method:	Inspect…

A.1.2. Requirement 2

Annex B: Title ({Normative/Informative})

NOTE	Place other Annex material in sequential annexes beginning with "B" and leave final two annexes for the Revision History and Bibliography

Annex C: Revision History

Date	Release	Editor	Primary clauses modified	Description
2016-04-28	0.1	G. Editor	all	initial version

Annex D: Bibliography

NOTE

Example Bibliography (Delete this note).

The TC has approved Springer LNCS as the official document citation type.

Springer LNCS is widely used in technical and computer science journals and other publications

- For citations in the text please use square brackets and consecutive numbers: [1], [2], [3]

– Actual References:

[n] Journal: Author Surname, A.: Title. Publication Title. Volume number, Issue number, Pages Used (Year Published)

[n] Web: Author Surname, A.: Title, http://Website-Url

[1] OGC: OGC Testbed 12 Annex B: Architecture. (2015).