

Getty Vocabularies: Linked Open Data

Semantic Representation

Version: 2.1
Last updated: 12 Aug 2014
HTML version: <http://vocab.getty.edu/doc/> (for linking)
PDF version: <http://vocab.getty.edu/doc/gvp-lod.pdf> (for printing)
Formerly at: http://www.getty.edu/research/tools/vocabularies/lod/aat_semantic_representation.pdf
Initial version: Vladimir Alexiev, Joan Cobb, Gregg Garcia, Patricia Harpring.
Updates: Vladimir Alexiev, Joan Cobb

Table of Contents

1	Introduction.....	5
1.1	The Getty Vocabularies and LOD.....	5
1.1.1	About the AAT.....	5
1.1.2	About the TGN.....	5
1.2	Revisions, Review, Feedback.....	6
1.2.1	Revisions.....	6
1.2.1.1	Version 1.0.....	6
1.2.1.2	Version 1.1.....	6
1.2.1.3	Version 1.2.....	6
1.2.1.4	Version 1.3.....	6
1.2.1.5	Version 2.0.....	7
1.2.1.6	Version 2.1.....	7
1.2.1.7	Future Versions.....	7
1.2.2	External Review Process.....	7
1.2.3	Providing Feedback.....	7
1.2.4	Disclaimer.....	8
1.3	Abbreviations.....	9
1.4	RDF Turtle.....	10
1.5	Prefixes.....	10
1.5.1	External Prefixes.....	11
1.5.2	Descriptive Prefixes.....	12
1.6	GVP URLs and Prefixes.....	12
1.6.1	Common GVP URLs.....	13
1.6.2	AAT URLs.....	13
1.6.3	TGN URLs.....	14
1.6.4	Using GVP URLs.....	14
1.6.5	Named Graphs.....	14
1.7	Semantic Resolution.....	15
1.8	External Ontologies.....	16
1.8.1	DC and DCT.....	16
1.8.2	SKOS and SKOS-XL.....	16
1.8.3	ISO 25964.....	17
1.8.4	BIBO.....	18
1.8.5	FOAF.....	18
1.8.6	PROV.....	18
1.8.6.1	dct:modified.....	19
1.8.6.2	dct:creator+dct:created.....	19
1.8.7	Geographic Ontologies.....	19
1.8.7.1	W3C WGS Geo Ontology.....	20



1.8.7.2	Schema.org Geographic Features	20
1.9	GVP Ontology	21
2	Semantic Representation	21
2.1	Semantic Overview	21
2.2	Subject	24
2.2.1	Subject Types	25
2.3	Subject Hierarchy	25
2.3.1	Standard Hierarchical Relations	26
2.3.2	GVP Hierarchical Relations	26
2.3.3	Hierarchy Structure	27
2.3.4	Top Concepts	28
2.4	Sort Order	30
2.4.1	Sorting with Thesaurus Array	30
2.4.1.1	skos:member Structure	30
2.4.1.2	skos:memberList Structure	31
2.4.1.3	Full Representation	32
2.5	Associative Relationships	33
2.5.1	Relationships Table	33
2.5.2	Relationship Cross-Walk	33
2.5.3	Relationship Representation	33
2.6	Obsolete Subject	34
2.7	Language	35
2.7.1	IANA Language Tags	35
2.7.2	GVP Language Tags	36
2.7.3	Language Tag Case	36
2.7.4	Language Tags and Sources	37
2.7.5	Language Dual URLs	37
2.8	Term	38
2.8.1	Term Characteristics	38
2.8.2	Importance of the Vernacular Flag	38
2.9	Scope Note	39
2.10	Identifiers	39
2.11	Notations	39
2.12	Source	40
2.12.1	Local Sources	40
2.13	Contributor	41
2.14	Historic Information	42
2.14.1	Applying to Terms	42
2.14.2	Applying to Relations and Place Types	42
2.15	Revision History	43
2.15.1	Revision History Representation	44
2.15.2	Revision History for Subject	44
2.15.3	Revision History for Source	45
3	TGN Specifics	46
3.1	TGN Overview	46
3.2	TGN Place Types	46
3.2.1	Why not broaderInstantial?	47
3.3	Concept vs Place Duality	49
3.3.1	Cons of the Dual Approach	50
3.3.2	Co-reference and Co-denotation	50
3.3.3	VIAF: pro	51
3.3.4	FR BnF: pro	52
3.3.5	UK BL: pro	53

3.3.6	SE KB: pro	53
3.3.7	US LoC: cons	54
3.3.8	DE DNB: cons	54
3.4	Coordinate Information	54
4	Additional Features	55
4.1	Inference.....	55
4.1.1	Extended Property Constructs	55
4.1.2	Reduced SKOS Inference	57
4.1.3	SKOS member vs memberList	59
4.1.4	SKOS-XL Inference	59
4.1.5	BTG, BTP, BTI Inference	59
4.1.6	BTG, BTP, BTI Axioms.....	60
4.1.7	broaderPreferredExtended Rules	61
4.1.8	ISO Insert Queries	61
4.1.9	ISO Rules	62
4.1.10	Hierarchical Relations Inference	63
4.1.11	FTS Insert Queries.....	64
4.1.12	OntoGeo Insert Query	65
4.2	Alignment.....	65
4.2.1	LCSH Alignment.....	65
4.2.2	AATNed Alignment	65
4.3	Forest UI.....	66
4.4	Full Text Search	67
4.5	Descriptive Information	68
4.5.1	VOID	68
4.5.2	DCAT	70
4.5.3	ADMS	71
4.5.3.1	W3C ADMS.....	72
4.5.4	Descriptive Entities	73
4.5.5	Descriptive Relations.....	74
4.5.6	Descriptive Properties.....	74
4.5.7	License Info	76
4.5.8	VOID Subsets	76
4.5.9	VOID Linksets.....	77
4.5.10	Dynamic Descriptive Properties	77
4.5.11	VOID Deployment	78
4.5.12	Possible Future Additions.....	79
4.6	Export Files	79
4.6.1	Explicit Exports	79
4.6.2	Per-Entity Exports	80
4.6.3	Total Exports	80
5	Sample Queries	81
5.1	Finding Subjects.....	81
5.1.1	Top-level Subjects.....	81
5.1.2	Descendants of a Given Parent	81
5.1.3	Subjects by Contributor Id.....	81
5.1.4	Subjects by Contributor Abbrev	81
5.1.5	Preferred Ancestors	81
5.1.6	Full Text Search Query.....	82
5.1.7	Find Person Occupations by broaderExtended.....	82
5.1.8	Find Person Occupations by Double FTS.....	82
5.1.9	Find Quartz Timepieces by Double FTS	83
5.1.10	Find Subject by Exact English PrefLabel	83

5.1.11	Find Subject by Language-Independent PrefLabels	83
5.1.12	Find Subject by Any Label	83
5.1.13	Find Terms by Language Tag	84
5.1.14	Find Ordered Subjects	84
5.1.15	Find Ordered Hierarchies	84
5.1.16	Get Subjects in Order	84
5.1.17	Find Contributors by Vocabulary	84
5.1.18	Find Sources by Vocabulary	85
5.2	Getting Information	85
5.2.1	Subject Preferred Label	85
5.2.2	All Data for Terms of Subject	85
5.2.3	Preferred and Vernacular Terms	85
5.2.4	Scientific Names by Language	85
5.2.5	Scientific Names not in English and Latin	86
5.2.6	All Data For Subject	86
5.2.7	Historic Information on Relations	87
5.2.8	Historic Information of Terms	87
5.2.9	Preferred Terms for Contributors	88
5.2.10	Preferred Terms for Sources	88
5.2.11	Concepts Related by Particular Associative Relation	88
5.2.12	Languages and ISO Codes	89
5.2.13	Language URLs	89
5.3	TGN-Specific Queries	89
5.3.1	Places by Type	89
5.3.2	Places, with English or GVP Label	89
5.3.3	Places by Direct and Hierarchical Type	90
5.3.4	Breakdown of Sovereign States by Type	91
5.3.5	Inhabited Places That Were Sovereign States	91
5.3.6	Places by Type and Parent Place	91
5.3.7	Places by Type, with placeTypePreferred	92
5.3.8	Places by Triple FTS	92
5.3.9	Places by FTS Parents	93
5.3.10	Capitals by Type	94
5.3.11	Capitals by Association	94
5.3.12	Members of the European Union	94
5.3.13	Members of the United Nations	94
5.3.14	Geo Chart with SPARQL	95
5.3.15	Column Chart with SPARQL	96
5.3.16	Countries and Capitals By Type and Containment	97
5.3.17	Places by Coordinate Bounding Box	97
5.3.18	Places Within Bounding Box	98
5.3.19	Places by Type Within Bounding Box	98
5.3.20	Places Outside Bounding Box (Overseas Possessions)	98
5.3.21	Places Nearby Each Other	98
5.4	Counting and Descriptive Info	99
5.4.1	Descriptive Info from VOID	99
5.4.2	Number of Entities from VOID	99
5.4.3	Number of Entities (Dynamic)	99
5.4.4	Number of Terms per Language	99
5.4.5	Number of Revision Actions	99
5.5	Explore the Ontology	99
5.5.1	Ontology Classes and Properties	100
5.5.2	Ontology Values	100

1 Introduction

This document explains the representation of the Getty Vocabularies in semantic format, using RDF and appropriate ontologies.

For the time being, it covers the Art and Architecture Thesaurus (AAT)® and the Thesaurus of Geographic Names (TGN)®. It will eventually also include the remaining two Getty Vocabularies, the Union List of Artist Names (ULAN)® and the Cultural Objects Name Authority (CONA)®, as they become available.

The document is published in HTML format, with appropriately and permanently named anchors for each section that can be shared in discussions. It is also published in PDF appropriate for printing.

1.1 The Getty Vocabularies and LOD

The Getty Vocabularies were first built to help people categorize, describe, and index cultural heritage objects and information. The Getty Vocabularies are compliant with international standards and grow through contribution.

Now we have the technology to transform these human-defined relationships into machine-readable data sets and embed them into the evolving semantic web. By publishing the Getty Vocabularies as Linked Data in an open environment for anyone to freely use, we are sharing with the world the results of over thirty years of research and scholarship.

1.1.1 About the AAT

AAT is a structured, multilingual vocabulary including terms, descriptions, and other information for generic concepts related to art, architecture, other cultural heritage, and conservation. For decades now, the AAT has been used as a primary reference by museums, art libraries, archives, visual resource catalogers, conservation specialists, archaeological projects, bibliographic projects, researchers, and information specialists who are dealing with the needs of these users.

Terms for any concept may include the plural form of the term, singular form, natural order, inverted order, spelling variants, scientific and common forms, various forms of speech, and synonyms that have various etymological roots. Among these terms, one is flagged as the term (*or descriptor*) preferred by the Getty Vocabulary Program. There may be multiple *descriptors* reflecting usage in multiple languages. Preferences for individual contributors may differ and are noted.

The AAT is a *thesaurus* in compliance with ISO and NISO standards.

The focus of each AAT record is a concept. Linked to each concept are terms, related concepts, its position in the hierarchy, sources for the data, and notes. The conceptual framework of facets and hierarchies in the AAT is designed to allow a general classification scheme for art, architecture and conservation. The framework is not subject-specific; for example, there is no defined portion of the AAT that is specific only for *Renaissance painting*. The terms to describe Renaissance paintings will be found in many locations in the AAT hierarchies. There may be multiple broader contexts, making AAT *polyhierarchical*. In addition the AAT has equivalence and associative relationships. The temporal coverage of the AAT ranges from Antiquity to the present and the scope is global.

Currently the AAT includes info about 42k Subjects, 300k Terms, 84k Scope Notes, 40k Sources and 165 Contributors (see [Counting and Descriptive Info](#)). See [more information](#) about the history, purpose and scope of the AAT.

1.1.2 About the TGN

TGN is a structured vocabulary containing names and other information about places. Names for a place may include names in the vernacular language, English, other languages, historical names and in natural and inverted order. Among these names, one is flagged as being preferred by the Vocabulary Program. Preferences for individual contributors may differ and are noted.

TGN is a thesaurus, compliant with ISO and NISO standards for thesaurus construction; it contains hierarchical, equivalence, and associative relationships. Currently there are two facets, *World* and *Extraterrestrial Places*. Under the *World*, places are arranged in hierarchies generally representing the current political and physical world, although some historical nations and empires are also included. There may be multiple broader contexts making the TGN *polyhierarchical*.

The focus of each TGN record is a place. Linked to each place are names, the place's position in the hierarchy, other relationships, geographic coordinates, notes, sources for the data, and *place types*, which are terms from the AAT describing the roles of the place over time (e.g., *inhabited place* and *state capital*). The temporal coverage of the TGN ranges from prehistory to the present and the scope is global. See [more information](#) about the history, purpose and scope of the TGN.

Note that TGN is not a GIS (Geographic Information System): it is a thesaurus. While many records in TGN include coordinates, these coordinates are approximate and are intended for reference ("finding purposes") only. Geographic

coordinates in TGN typically represent a single point, corresponding to a point in or near the center of the inhabited place, political entity, or physical feature. For linear features such as rivers, the point represents the source of the feature. Some areas and regions (e.g. the Great Lakes region) have bounding box coordinates.

Currently the TGN includes info about 1.26M places (Subjects), 1.85 names (Terms), 22k Scope Notes, 3.1k Sources, 166 Contributors, 1.33M Place Type instances, 722 distinct Place Types

1.2 Revisions, Review, Feedback

1.2.1 Revisions

We anticipate that in the future, the GVP LOD data will be refreshed every two weeks, on the same schedule as the online public data and the data files available to licensees through Web services. This document and the underlying ontology and mapping is also updated frequently.

- A record of all significant changes is in the following subsections
- Appropriate metadata (dct:modified, dct:issued, owl:versionInfo) is available in the [Descriptive Information](#)

1.2.1.1 Version 1.0

19 Feb 2014: Initial Version

1.2.1.2 Version 1.1

28 Feb 2014: Draft that was posted for about 2 weeks. Do not use.

1.2.1.3 Version 1.2

14 Mar 2014

- Mapping change: removed intermediate bibo:DocumentPart node in [Language Tags and Sources](#)
- Mapping change: removed [Top Concepts](#) indication since it is not considered useful
- Mapping change: touched [Relationship Representation](#) a bit to fit better the used ontology documentation tool (Parrot).
- Added comprehensive [Descriptive Information](#) (VOID etc)
- Added documentation: [Language Tag Case](#), ontology documentation at end of [GVP Ontology](#), clarification at end of [Hierarchy Structure](#), clarification at end of [Language Dual URLs](#).
- Sample Queries: added language-related ([Find Subject by Exact English PrefLabel](#), [Find Subject by Language-Independent PrefLabels](#), [Find Subject by Any Label](#), [Find Terms by Language Tag](#), [Languages and ISO Codes](#), [Language URLs](#)), added [Explore the Ontology](#) (3 queries), updated the queries in the SPARQL endpoint, provided link to this documentation (exclamation point icon)
- Site change: removed parasitic word "/resource" from [GVP URLs](#)

1.2.1.4 Version 1.3

15 Apr 2014

- Mapping change: omitted [Sort Order](#) of Facets and Hierarchies since it's not considered useful
- Omitted blank nodes from [Per-Entity Exports](#) and [Total Exports](#) (see [Construct Subject](#))
- Added another illustration of the [VOID](#) domain model
- Added URLs for [VOID Linksets](#), e.g. <http://vocab.getty.edu/dataset/aat/alignment/lcsh>
- Added the full VOID info to the repository (in addition to a designated file), see [VOID Deployment](#). Changed the queries for [Dynamic Descriptive Properties](#) to INSERT instead of CONSTRUCT. Added all [Prefixes](#) to [prefixes.ttl](#)
- Added 2 descriptive info queries at the end of [Counting and Descriptive Info](#)
- Produced this document in HTML (inside <http://www.getty.edu/research/tools/vocabularies/loa/aat-images.zip>), in addition to PDF

1.2.1.5 Version 2.0

11 Jul 2014

Added the second vocabulary: TGN (see [TGN Specifics](#) and [TGN-Specific Queries](#)), which is currently in BETA (official launch is expected in late August 2014).

Surprisingly few changes were needed:

- Mapping change: Made [Term Characteristics](#) independent of AAT (moved [GVP URLs](#) one level up), in preparation for adding TGN.
- Mapping change: Introduced "Extended" versions of [GVP Hierarchical Relations](#) and removed gvp:broaderTransitive. You should [Use gvp:broaderExtended not skos:broaderTransitive](#) for appropriate query expansion. Infer [Standard Hierarchical Relations](#) (e.g. iso:broaderGeneric) from these Extended versions (e.g. gvp:broaderGeneric), see [BTG](#), [BTP](#), [BTI Inference](#) and [ISO Insert Queries](#)
- Added [Hierarchical Relations Inference](#) that should clarify the different relations and which contributes to which others
- Mapping change: omitted Undetermined language in [GVP Language Tags](#)
- Mapping change: use schema:startDate,endDate for [Historic Information](#) and [Obsolete Subject](#), instead of custom properties gvp:startDate,endDate.
- Mapping change: map [Scope Note](#) to custom class gvp:ScopeNote with rdf:value, instead of xl:Label with xl:literalForm.
- Declared [Associative Relationships](#) as sub-properties of skos:related (see [Relationship Representation](#)).
- Added more dct:formats to [Descriptive Properties](#) (both NTriples and ZIP for total exports, "meta/void" for descriptor, "meta/rdf-schema" for ontology)
- Added [Named Graphs](#) (used for a very limited purpose).
- Moved this document to a more logical location (see first page)
- Made the HTML version primary, and made properly named anchors for each section

1.2.1.6 Version 2.1

8 Aug 2014

- After experimentation with inferred data, decided not to map [TGN Place Types](#) to gvp:broaderInstantial, iso:broaderInstantial, nor skos:broader; see [Why not broaderInstantial?](#) Rewrote all [TGN-Specific Queries](#) accordingly, see in particular [Places by Direct and Hierarchical Type](#). This issue may be revisited in the future.
- Removed some superfluous relations, see [Reduced SKOS Inference](#). Also see struck-out properties in [Standard Hierarchical Relations](#), [GVP Hierarchical Relations](#). This is in order to reduce the number of triples, which was 440M, representing an inference expansion ratio of 4x.

1.2.1.7 Future Versions

Your feedback is appreciated on the following mapping changes under consideration:

- Map TGN to more [Geographic Ontologies](#)
- Remap [TGN Place Types](#) to gvp:broaderInstantial

1.2.2 External Review Process

[Numerous individuals](#) are serving as External Advisors on this project. Most have been recommended by colleagues in our community (e.g., International Terminology Working Group members who are currently translating the AAT into various languages). We ended up inviting a fairly large group because we wanted to make sure that we had expertise in many areas. It has been very important to us that these trusted colleagues had a chance to comment on our ontology choices prior to the release of the datasets.

1.2.3 Providing Feedback

We welcome comments if you find something in this document or our dataset that needs clarification or improvement. We ask specifically for your opinion in several places in this document.

We have established an open community and we welcome collaboration. The preferred way of communicating technical questions and comments is to use the SemanticWeb.com Q&A forum (previously we were getting feedback at an email

address but now the forum is preferred). We will be monitoring this forum regularly. The tags to use are: Getty, AAT, TGN, ULAN, CONA. The following links will pre-populate a question form with the appropriate tag:

- <http://answers.semanticweb.com/questions/ask/?tags=Getty,AAT>
- <http://answers.semanticweb.com/questions/ask/?tags=Getty,TGN>
- <http://answers.semanticweb.com/questions/ask/?tags=Getty,ULAN>
- <http://answers.semanticweb.com/questions/ask/?tags=Getty,CONA>

Questions and comments specifically about the Beta release of TGN should be directed to vocablod@getty.edu

Questions and comments about editorial content or general information regarding the Getty Vocabularies (not LOD) should be directed to vocab@getty.edu.

1.2.4 Disclaimer

The vocabulary datasets are provided "as is". The Getty disclaims all other warranties, either express or implied, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose, with respect to the database. The Getty Vocabularies are compiled by the Getty Vocabulary Program from contributions from various contributors, including museums, libraries, archives, bibliographic indexing projects, international translation projects, and others. Not all contributor data complies precisely with the GVP Editorial Guidelines; therefore, absolute consistency in the dataset is not possible. The data is subject to frequent updates and corrections.

Please keep in mind that the TGN portion of the data is considered by the Getty to be a BETA release. It is being offered early to give the external community a chance to comment prior to the official launch date in late August 2014.

- LOD data is provided as [Export Files](#)
- See [more information](#) about various ways in which the Getty vocabularies may be obtained.

1.3 Abbreviations

The following abbreviations are used in the document. In addition, [External Prefixes](#) and [Descriptive Prefixes](#) includes abbreviations of the ontologies used (e.g. RDF is Resource Description Framework, VoID is Vocabulary of Interlinked Datasets, etc)

Abbrev	Term
AACR2	Anglo-American Cataloging Rules 2
AAT	Art and Architecture Thesaurus
AATNed	AAT Netherlands: the project to make the Dutch translation of the AAT
BTG	Broader Than Generic (Genus/Species) relation: property broaderGeneric
BTP	Broader Than Partitive (Part/Whole) relation: property broaderPartitive
BTI	Broader Than Instantial (Kind/Instance) relation: property broaderInstantial
Forest	Ontotext semantic UI framework
FTS	Full Text Search
GCI	Getty Conservation Institute
GRI	Getty Research Institute
GVP	Getty Vocabulary Program
IANA	Internet Assigned Numbers Authority
ISO	International Standardization Organization
LCSH	Library of Congress Subject Headings
LoC	Library of Congress
LOD	Linked Open Data
Lucene	Lucene FTS engine
OWLIM	Ontotext semantic (RDF) repository
RDFa	RDF in Attributes: allows the embedding of RDF data in HTML
SPARQL	SPARQL Protocol and RDF Query Language
TGN	Thesaurus of Geographic Names
UI	User Interface
URI	Unified Resource Identifier. Following LOD principles, all our URIs are resolvable, i.e. URLs
URL	Unified Resource Locator
W3C	World Wide Web Consortium

1.4 RDF Turtle

This document uses examples written in [Turtle](#), which is much more readable than other RDF representations such as RDF XML or NTriples.

We use URLs derived from AAT database IDs, so our prefixed names start with a digit.

Examples:

- `aat:300198841`: a [Subject](#)
- `aat_term:1000198841-el-Latn`: a [Term](#)
- `aat_source:2000051089-term-1000198841`: a [Source](#) (namely the "local source") pertaining to this term

The Turtle 1.1 Candidate Recommendation of 19 February 2013 allows the [local part of a prefixed name](#) to start with a digit. We provide [Export Files](#) in Turtle, RDF XML, NTriples and JSON. To load the Turtle files you need recent parsing tools that support this Turtle 1.1 feature. For example:

- Sesame RIO 2.7.9 (2013-12-18)
- Jena ARQ 2.8.8 (2011-04-21)

Turtle 1.1 names may also include colon ":" (CURIE) and may include other special characters through escaping (e.g. at-sign "@" and slash "/"). However we limit the characters used in local names to letters, digits and dash "-" (e.g. the Jena ARQ version cited above does not handle ":" and escaped chars). Because Turtle 1.1 capable tools are not very widely deployed as of the end of 2013, we provide the [Total Export](#) files in **NTriples** format.

1.5 Prefixes

The prefixes that we use (both internal and external) are defined in the following sections.

The easiest way to find prefixes is the [prefix.cc](#) service.

- For example, if you wonder what the `rr:` prefix means, go to <http://prefix.cc/rr>. You will quickly find the URL, and whether any other alternatives have been registered.
- For the [GVP Prefixes](#), we have registered `gvp:`, `aat:`, `tgn:` and `ulan:` in this service (unfortunately it does not support prefixes including "_")
- If you need the prefix to put in a Turtle file, go directly to e.g. <http://prefix.cc/rr.ttl>. This page allows you to copy it to the clipboard, so you can paste it into a file
- You can request several prefixes at once, e.g.: <http://prefix.cc/gvp.aat.tgn.ulan.ttl>

For your convenience, all prefixes that we use are provided at <http://vocab.getty.edu/doc/prefixes.ttl> (formerly <http://www.getty.edu/research/tools/vocabularies/lot/prefixes.ttl>)

1.5.1 External Prefixes

We use the following prefixes for [External Ontologies](#).

- rr: and rrx: (R2RML) are used only during the conversion process.
- luc: is used for [Full Text Search Query](#)

Prefix	URL	Explanation
bibo:	http://purl.org/ontology/bibo/	Bibliography Ontology
dc:	http://purl.org/dc/elements/1.1/	Dublin Core Elements
dct:	http://purl.org/dc/terms/	Dublin Core Terms
foaf:	http://xmlns.com/foaf/0.1/	Friend of a Friend ontology
iso:	http://purl.org/iso25964/skos-thes#	ISO 25946 ontology (latest ISO standard on thesauri)
luc:	http://www.ontotext.com/owlim/lucene#	OWLIM's built-in Lucene Full Text Search
ontogeo:	http://www.ontotext.com/owlim/geo#	OWLIM geo-spatial extensions, e.g. Places Within Bounding Box , Places Near Each Other
owl:	http://www.w3.org/2002/07/owl#	Web Ontology Language
prov:	http://www.w3.org/ns/prov#	Provenance Ontology
ptop:	http://www.ontotext.com/proton/protontop#	PROTON ontology, used in Extended Property Constructs
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Resource Description Framework
rdfs:	http://www.w3.org/2000/01/rdf-schema#	RDF Schema
rr:	http://www.w3.org/ns/r2rml#	Relational to RDF Mapping Language, used for conversion only
rrx:	http://purl.org/r2rml-ext/	R2RML extension (rrx:languageColumn)
schema:	http://schema.org/	Schema.org common properties
skos:	http://www.w3.org/2004/02/skos/core#	Simple Knowledge Organization System
skosxl:	http://www.w3.org/2008/05/skos-xl#	SKOS Extension for Labels
wgs:	http://www.w3.org/2003/01/geo/wgs84_pos#	W3C Geo ontology (WGS stands for the World Geodetic Survey 1984 datum)
xsd:	http://www.w3.org/2001/XMLSchema#	XML Schema Datatypes

1.5.2 Descriptive Prefixes

We use some of the following prefixes for [Descriptive Information](#):

Prefix	URL	Explanation
adms:	http://www.w3.org/ns/adms#	Asset Description Metadata Schema
cc:	http://creativecommons.org/ns#	Creative Commons Rights Expression Language
dc:	http://purl.org/dc/elements/1.1/	Dublin Core Metadata Element Set
dcat:	http://www.w3.org/ns/dcat#	Data Catalog Vocabulary
dct:	http://purl.org/dc/terms/	DCMI Metadata Terms
dctype:	http://purl.org/dc/dcmitype/	DCMI Type Vocabulary
fmt:	http://www.w3.org/ns/formats/	RDF formats used in datasets
sd:	http://www.w3.org/ns/sparql-service-description#	SPARQL Service Description
vaem:	http://www.linkedmodel.org/schema/vaem#	Vocabulary for Attaching Essential Metadata
vann:	http://purl.org/vocab/vann/	Vocabulary for annotating vocabulary descriptions
vcad:	http://www.w3.org/2006/vcard/ns#	vCard (contact info)
vdpp:	http://data.lirmm.fr/ontologies/vdpp#	Vocabulary for Dataset Publication Projects
voaf:	http://purl.org/vocommons/voaf#	Vocabulary of a Friend
voag:	http://voag.linkedmodel.org/voag#	Vocabulary Of Attribution and Governance
void:	http://rdfs.org/ns/void#	Vocabulary of Interlinked Datasets
wdrs:	http://www.w3.org/2007/05/powder-s#	Protocol for Web Description Resources
wv:	http://vocab.org/waiver/terms/	A vocabulary for waivers of rights

1.6 GVP URLs and Prefixes

The layout of GVP URLs (both ontology and vocabulary entities) is shown below.

- We use a template notation in the URLs: {m} indicates a numeric identifier, {v} some value, {lang} a language tag
- We have defined prefixes for the most important URLs.
- The AAT subject URIs **aat:{m}** are the only ones that will be used in external datasets, so they are more important than all the other URIs. We keep them as short as possible, allowing shortest Turtle and SPARQL.
- The TGN subject URIs are of the same form **tgn:{m}**. However in your data you will most often want to use the place URIs **tgn:{m}-place**, see [Concept vs Place Duality](#).
- URLs shown in roman font represent independent entities. The data of URLs shown in *italic font* is returned together with the owning Subject so that you won't have to chase different URLs to gather the information (these URLs can also be resolved on their own).

1.6.1 Common GVP URLs

Prefix	URL	Explanation
base	http://vocab.getty.edu	Prepend to all URIs. Returns a home page describing the LOD vocabularies and giving links to the ontology, vocabularies, sample resources, etc
	/dataset/	Directories with data dumps: (aat tgn)/(explicit full).zip
gvp:	/ontology#	GVP Ontology. Uses SKOS, SKOS-XL, ISO 25964, DC, DCT, BIBO, FOAF and PROV. Also holds Associative Relations
gvp_lang:	/language/{lang}	Languages used by GVP. URLs use IANA language tags. Have owl:sameAs links to corresponding concepts in the AAT Languages hierarchy
	/historic/{v}	Historic Flag: Current, Historic or Both
	/term/display/{v}	Term use: for Display or Indexing?
	/term/flag/{v}	Term Flag: Vernacular, Loan Term
	/term/kind/{v}	Term Kind: Neologism, Scientific Name, etc
	/term/POS/{v}	Term Part of Speech: Noun Plural, Noun Singular, etc
	/term/type/{v}	Term Type: Descriptor, Alternate Descriptor, Use For term
	/.well-known/void	VOID Descriptive Information (see VOID Deployment)
	/void.ttl	

1.6.2 AAT URLs

Prefix	URL	Explanation
aat:	/aat/	AAT vocabulary (skos:ConceptScheme)
	/aat/{m}	AAT subject {m}: Facet, Hierarchy, Guide Term, or Concept
	/aat/{m}-array	Anonymous iso:ThesaurusArray used to represent the ordered concept {m}
	/aat/{m}-list-{n}	rdf:List element for child subject {n} of the skos:OrderedCollection used to represent the ordered subject {m}
aat_contrib:	/aat/contrib/{m}	AAT Contributor {m}
aat_rel:	/aat/rel/{m}-{type}-{n}	AAT Relation: from subject {m} to subject {n}, having {type} ("broader" or an associative relation)
aat_rev:	/aat/rev/{m}	AAT Revision of a subject
aat_scopeNote:	/aat/scopeNote/{m}	AAT Scope Note (definition) {m}
aat_source:	/aat/source/{m}	AAT Source {m}
	/aat/source/{m}-scopeNote-{n}	AAT Source {m} applied to scope note {n}
	/aat/source/{m}-subject-{n}	AAT Source {m} applied to subject {n}
	/aat/source/{m}-term-{n}	AAT Source {m} applied to term {n}
aat_term:	/aat/term/{m}	AAT Term {m}

Sources and Contributors are thesaurus-dependent, i.e. even if the same organization contributes to both AAT and TGN, it will get two different URLs `aat_contrib:123` and `tgn_contrib:456`. Also, contributors are not correlated to ULAN. Sorry about that.

1.6.3 TGN URLs

Prefix	URL	Explanation
tn:	/tn/	TGN vocabulary (skos:ConceptScheme)
	/tn/{m}	TGN subject {m}: Facet, gvp:PhysPlaceConcept, gvp:AdminPlaceConcept, or gvp:PhysAdminPlaceConcept
	/tn/{m}-place	Place (wgs:SpatialThing and schema:Place) corresponding to TGN concept /tn/{m} (see Concept vs Place Duality)
	/tn/{m}-geometry	Geometry (schema:GeoCoordinates, schema:GeoShape) of /tn/{m}-place. Schema coordinates are in this node, but WGS coordinates are in /tn/{m}-place.
	/tn/{m}-array	Anonymous iso:ThesaurusArray used to represent the ordered concept {m}
	/tn/{m}-list-{n}	rdf:List element for child subject {n} of the skos:OrderedCollection used to represent the ordered concept {m}
tn_contrib:	/tn/contrib/{m}	TGN Contributor {m}
tn_rel:	/tn/rel/{m}-{type}-{n}	TGN Relation {m}-{type}-{n}: from subject {m} to subject {n}, having {type} ("broader", "placeType", or an associative relation)
tn_rev:	/tn/rev/{m}	TGN Revision of a subject
tn_scopeNote:	/tn/scopeNote/{m}	TGN Scope Note (definition) {m}
tn_source:	/tn/source/{m}	TGN Source {m}
	/tn/source/{m}-scopeNote-{n}	TGN Source {m} applied to scope note {n}
	/tn/source/{m}-subject-{n}	TGN Source {m} applied to subject {n}
	/tn/source/{m}-term-{n}	TGN Source {m} applied to term {n}
tn_term:	/tn/term/{m}	TGN Term {m}

Please note that URLs including the parasitic word "/resource" (e.g. <http://vocab.getty.edu/resource/aat/{m}>) are NOT valid GVP URLs and you should not use them in your applications, nor make statements about them. This word was used in earlier versions of the <http://vocab.getty.edu> website due to the internal architecture of [Forest UI](#). People were copying them from the browser address bar and sharing them in discussions, so we went to the trouble of reworking Forest to remove this word, in order to avoid confusion.

1.6.4 Using GVP URLs

GVP URLs are guaranteed to stay stable, as explained in [Identifiers](#). If a subject is discontinued or merged to another, it'll be present in the GVP LOD as [Obsolete Subject](#) for at least 5 years.

- For AAT, you'll probably only want to use the Subject URLs `aat:{m}`
- For TGN, you should **use the Place URL `tn:{m}-place` in your Cultural Heritage data** (e.g. as place of creation of a CH artifact or place of birth of a painter). An event cannot take place in a skos:Concept, so an appropriately typed node is needed for this purpose (see [Concept vs Place Duality](#)).
- You may also want to use a specific label `aat_term:{m}` or `tn_term:{m}`, if you are describing the use of a particular term/name in history.

1.6.5 Named Graphs

We split the GVP data in different named graphs:

- <http://vocab.getty.edu/dataset/aat>: AAT data (subjects, terms, revisions, relations, sources, contributors)
- <http://vocab.getty.edu/dataset/tn>: TGN data (subjects, terms, revisions, relations, sources, contributors, places, geometries)
- <http://vocab.getty.edu/.well-known/void>: [Descriptive Information](#)

- default (empty) graph: all inferred statements (see [Inference](#))

We do this only so we can provide per-vocabulary triple counts in [Dynamic Descriptive Properties](#). For most purposes, you should ignore the existence of these named graphs:

- You can get all data of a particular vocabulary using the separate [Export Files](#).
- The named graphs are not available in the export files.
- You cannot get per-vocabulary data with filtering by named graph, since all inferred statements are in the empty graph (due to a limitation of OWLIM).
- It would not be smart to deploy only TGN data, since TGN refers to AAT ([TGN Place Types](#) are AAT subjects)

You should follow these patterns:

- To get all subjects from a particular vocabulary, filter by `skos:inScheme`
- To get all data about a subject, use the [Per-Entity Exports](#) or the query [All Data For Subject](#)
- To get all sources or contributors for a particular vocabulary: [Find Contributors by Vocabulary](#)

1.7 Semantic Resolution

All GVP, AAT and TGN URLs resolve, returning human or machine readable content through content negotiation.

- We followed the recommendation [Cool URIs for the Semantic Web](#)
- We followed [Best Practice Recipes for Publishing RDF Vocabularies](#)
- We validated the resolution with [Vapour](#)

Example about the ontology:

- <http://vocab.getty.edu/ontology> : semantic URI, content-negotiated using 303 redirect
- <http://vocab.getty.edu/ontology.html> : HTML page (application/xhtml+xml).
- <http://vocab.getty.edu/ontology.rdf> : application/rdf+xml
- <http://vocab.getty.edu/ontology.ttl> : text/turtle

Example about an AAT subject: aventurine (quartz)

- <http://vocab.getty.edu/aat/300011154> : semantic URI, content-negotiated using 303 redirect
- <http://vocab.getty.edu/aat/300011154.html> : Forest HTML page (application/xhtml+xml).
- <http://vocab.getty.edu/aat/300011154.rdf> : application/rdf+xml
- <http://vocab.getty.edu/aat/300011154.ttl> : text/turtle
- <http://vocab.getty.edu/aat/300011154.nt> : NTriples
- <http://vocab.getty.edu/aat/300011154.json> : JSON

Example about an AAT contributor: AATNed

- <http://vocab.getty.edu/aat/contrib/10000205> : semantic URI, content-negotiated using 303 redirect
- <http://vocab.getty.edu/aat/contrib/10000205.html> : Forest HTML page (application/xhtml+xml)
- <http://vocab.getty.edu/aat/contrib/10000205.rdf> : application/rdf+xml
- <http://vocab.getty.edu/aat/contrib/10000205.ttl> : text/turtle
- <http://vocab.getty.edu/aat/contrib/10000205.nt> : NTriples
- <http://vocab.getty.edu/aat/contrib/10000205.json> : JSON

Example about an AAT source: Van Nostrand's Scientific Encyclopedia

- <http://vocab.getty.edu/aat/source/2000041891> : semantic URI, content-negotiated using 303 redirect
- <http://vocab.getty.edu/aat/source/2000041891.html> : Forest HTML page (application/xhtml+xml)
- <http://vocab.getty.edu/aat/source/2000041891.rdf> : application/rdf+xml
- <http://vocab.getty.edu/aat/source/2000041891.ttl> : text/turtle
- <http://vocab.getty.edu/aat/source/2000041891.nt> : NTriples
- <http://vocab.getty.edu/aat/source/2000041891.json> : JSON

We use suffixes (file extensions) instead of prefixes (folders), since such URIs:

- Emphasize they all are about one semantic entity, instead of being put in "different folders", see [Hierarchical URIs Pattern](#)
- Are more hackable (easier to add suffix than spot the prefix), see [Patterned URIs Pattern](#)
- Provide correct file extensions for downloading

1.8 External Ontologies

Our mapping uses a number of external ontologies (as listed in [External Prefixes](#)):

- SKOS, SKOSXL, ISO 25964 for representing thesaurus info;
- DC, DCT for common properties;
- BIBO, FOAF for sources and contributors;
- WGS, Schema for geographic information;
- PROV for revision history;
- RDF, RDFS, OWL, XSD for system properties;
- R2RML for implementing the conversion.

The current versions of external ontologies should be loaded in the semantic repository. We use RDF instead of Turtle versions, to avoid the addition of spurious prefixes to the repository (e.g. skos.ttl defines an empty prefix "s:").

- SKOS: <http://www.w3.org/2004/02/skos/core.rdf>
- SKOS-XL: <http://www.w3.org/2008/05/skos-xl.rdf>
- ISO 25964: [iso-thes.rdf](#) obtained with

```
wget --header accept:application/rdf+xml http://purl.org/iso25964/skos-thes#
```

We don't load the other ontologies since they don't make useful inferences for us. In particular, DCT makes a lot of highly irrelevant inferences, e.g. dct:source infers dct:relation and dc:relation. (Feel free to load them in your own repository if you like.)

Brief notes about some of these ontologies follow.

1.8.1 DC and DCT

Dublin Core is an often-used ontology for defining basic metadata (e.g. title, creator, created, modified, issued, source, language, etc).

It defines two metadata sets: DC Elements (older, often denoted simply DC and using prefix dc:) and DC Terms (newer, often denoted DC and using prefix dct:). Their distinguishing characteristics are:

- DC properties allow any values, literal or URI alike. DCT properties are more strict, and a lot of them require URI.
- DC properties stand alone; DCT properties are often defined as sub-properties of DC (or other DCT properties)

By way of example, the two corresponding properties dc:language and dct:language are defined as if:

```
dc:language a rdf:Property .
dct:language a owl:ObjectProperty; rdfs:subPropertyOf dc:language; rdfs:range dct:LinguisticSystem .
```

- dc:language can take any value, either literal or URI
- dct:language takes only URIs, infers dc:language, and infers that the URI has class dct:LinguisticSystem.

We use DC/DCT for various common properties, e.g. dc:identifier, dct:source, dct:contributor, dct:created, dct:modified. If both a DC and DCT property fit a purpose, we use the DCT property if the target is a URL.

1.8.2 SKOS and SKOS-XL

SKOS is a widely used ontology for representing thesauri. SKOS-XL allows you to represent labels as nodes, and attach additional information

We assume the reader has basic knowledge of SKOS. You can consult the following sources:

- [SKOS Primer](#)
- [SKOS Reference](#)
- [SKOS-XL in Primer](#)
- [SKOS-XL in Reference](#)

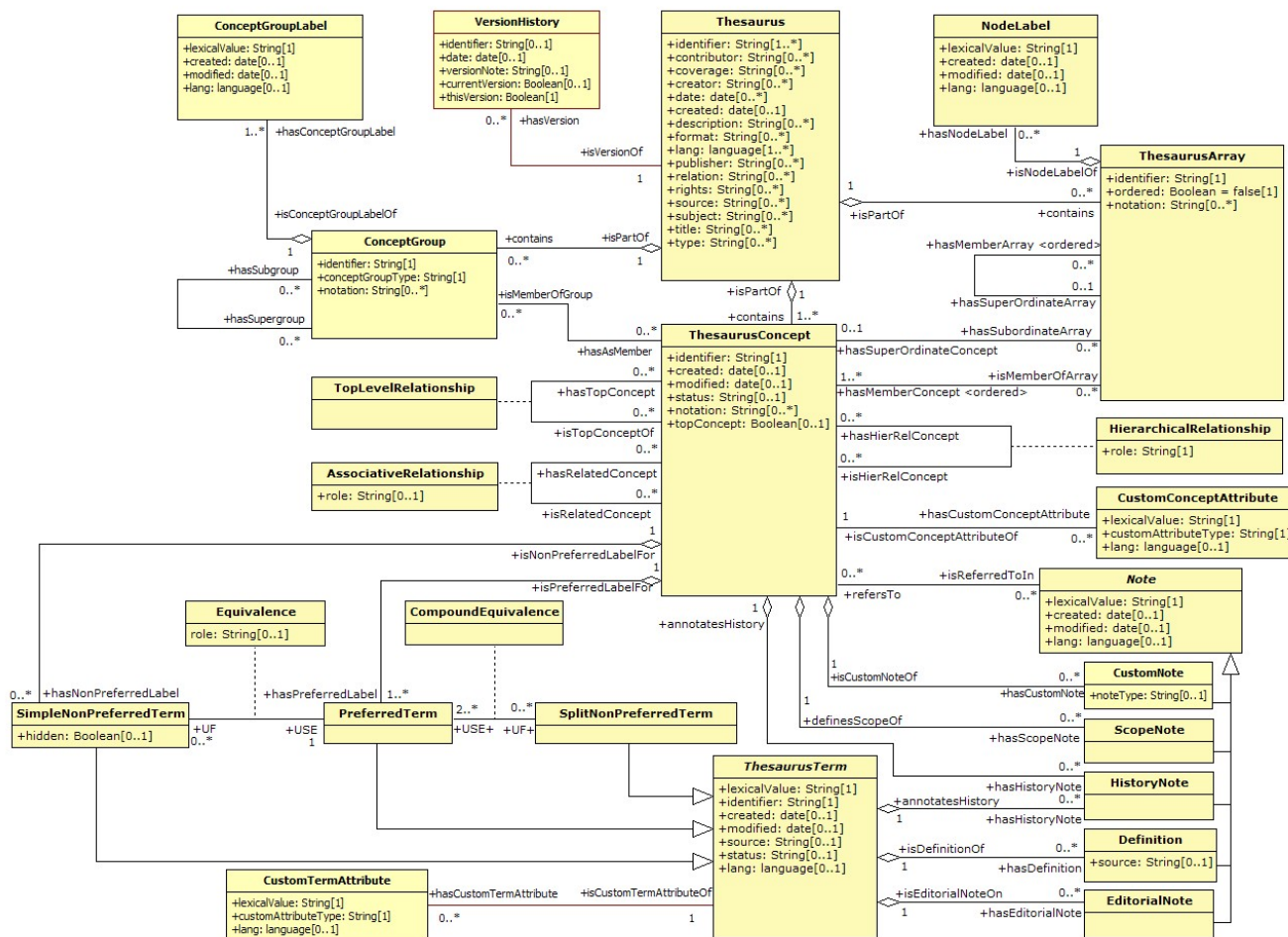
The SKOS standard, as any other standard, is the result of certain tradeoffs and timeline constraints. Therefore various issues and topics that were proposed for consideration did not make it into the final standard. The semantic representation of the GVP thesauri pushes the SKOS envelope in many cases, so it is useful to learn about approaches that go beyond the current SKOS standard.

We found this paper very useful: [Key choices in the design of Simple Knowledge Organization System \(SKOS\)](#), Journal of Web Semantics, May 2013.

See also sections [SKOS Inference](#) and [SKOS-XL Inference](#) in this document.

1.8.3 ISO 25964

[ISO 25964 - the international standard for thesauri and interoperability with other vocabularies](#) is the latest ISO standard on thesauri. The ISO 25964 domain model is shown below:



A lot of it corresponds to SKOS/SKOS-XL, see [Correspondence between ISO 25964 and SKOS/SKOS-XL Models](#). But it has additional constructs not covered by SKOS. In particular, skos:Collection has these limitations:

- you can't put them under a Concept

- you can't say explicitly which are Top Collections in a scheme
- you don't have inverse/transitive versions of skos:member

We use iso:ThesaurusArray, which is a subclass of skos:Collection but can be put under a Concept using iso:superOrdinate, see [Sorting with Thesaurus Array](#).

Despite the risks inherent in early adoption of new technology, Getty willingly undertakes this trail-blazing role, because the ISO ontology allows a faithful representation of GVP data, and in order to promote the adoption and deployment of the standard, which is the way to make technical progress.

We provided implementation experience to the ISO technical committee, and contributed suggestions and fixes to the isothes ontology (first published on 30 Sep 2013 at the public-esw-thes@w3.org mailing list). To the best of our knowledge, the application to AAT is the first industrial use of ISO 25964.

1.8.4 BIBO

The Bibliographic Ontology (BIBO) is used by various library projects, including the British National Bibliography. It is described at <http://bibliontology.com>, and is kept at [GitHub](#). The definition is available in [XML OWL](#), [N3](#), [OWLDoc](#). We use BIBO to:

- Represent [Source](#) information: a source is represented as bibo:Document
- We use bibo:DocumentPart if there is location information.

1.8.5 FOAF

The Friend of a Friend ontology (FOAF) is a well-known ontology for people, organizations, contacts, etc. We use FOAF to:

- Represent [Contributor](#) information: a contributor is represented as foaf:Agent.
- Link a place concept to its place using foaf:focus (see [Concept vs Place Duality](#))

1.8.6 PROV

Provenance is information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness.

PROV is a formalization of Provenance by the Provenance Working Group at W3C (PROVG). It defines a model, corresponding serializations, definitions supporting mapping to other provenance models, and examples how to use PROV. The PROV family of documents includes the following:

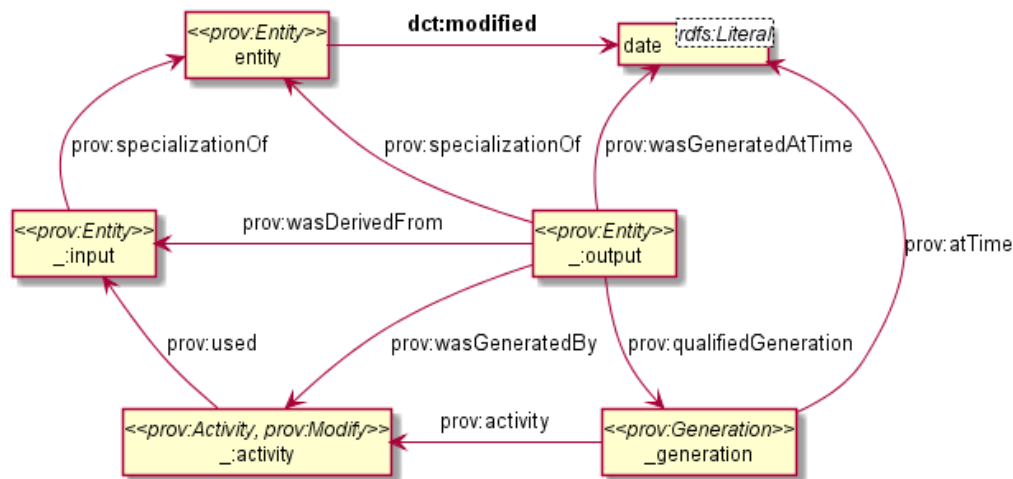
- [PROV-OVERVIEW](#), an overview of the PROV family of documents
- [PROV-PRIMER](#), a primer for the PROV data model
- [PROV-DM](#), the PROV data model
- [PROV-O](#), the PROV ontology, an OWL2 ontology allowing the mapping of the PROV data model to RDF.
- [PROV-XML](#), an XML schema for the PROV data model
- [PROV-N](#), a notation for provenance aimed at human consumption
- [PROV-CONSTRAINTS](#), a set of constraints applying to the PROV data model
- [PROV-AQ](#), mechanisms for accessing and querying provenance
- [PROV-DICTIONARY](#) introduces a specific type of collection, consisting of key-entity pairs
- [PROV-DC](#) provides a mapping between Dublin Core Terms and PROV-O
- [PROV-SEM](#), a declarative specification in terms of first-order logic of the PROV data model
- [PROV-LINKS](#): introduces a mechanism to link across bundles of provenance info

The PROV-O and PROV-DC ontologies are available as: prov-o.ttl, prov-dc-refinements.ttl.

The PROV-DC mapping illustrates the complexity of PROV best. Below are two examples:

1.8.6.1 *dct:modified*

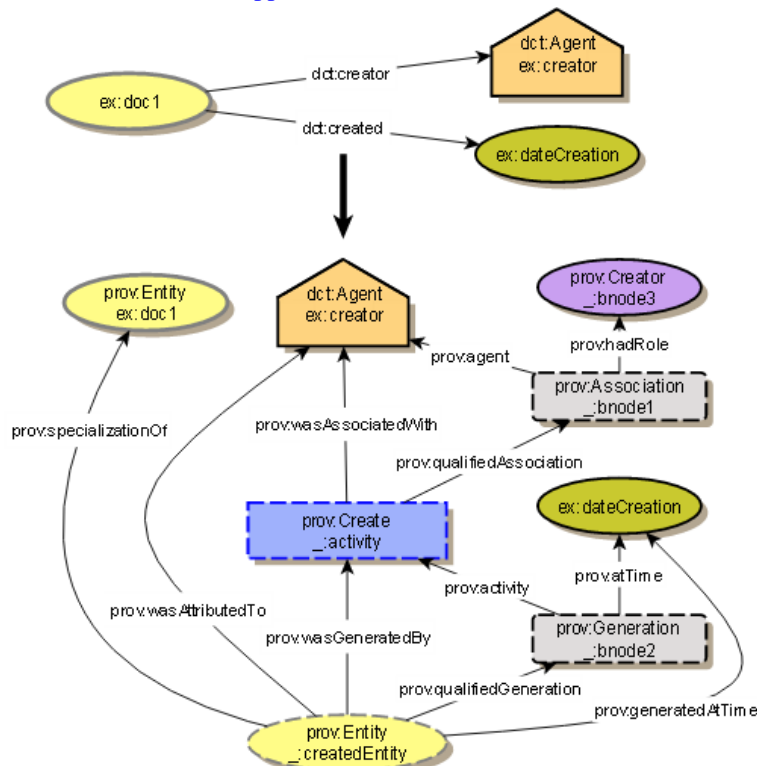
A single statement [dct:modified is mapped](#) to a network of 6 nodes and 9 statements:



PROV considers that the prov:Modify activity uses an unknown old entity (**_input**) and generates an unknown new entity (**_output**), both being specializations of the **entity** under consideration. Furthermore, we need to use a prov:Generation node to be able to use [prov:atTime](#) and reflect accurately that the modification is in fact a [prov:InstantaneousEvent](#).

1.8.6.2 *dct:creator+dct:created*

Two statements [dct:creator and dct:created are mapped](#) to a network of 8 nodes and 11 statements:



1.8.7 Geographic Ontologies

There is a plethora of ontologies related to geography out there. Here are some lists:

- [LOV Space and Geography vocabs](#): 20 ontologies
- [LOV Geometry vocabs](#): 8 ontologies
- [Spatial Ontology Community of Practice \(SOCoP\)](#), [Open Ontology Repository \(OOR\)](#). Unfortunately the site is dead, so these are links from archive.org (2012-06). It includes 40 [Ontologies](#) with 2578 terms (counts per ontology). But archive.org doesn't have the ontology pages, so we can use only the names

We considered the use of the following ontologies for TGN representation:

- **WGS** and **Schema.org**, as described below
- **GeoSPARQL**: [A Geographic Query Language for RDF Data](#) by the Open Geospatial Consortium (document OGC 11-052r4, Version: 1.0, Approval Date: 2012-04-27, Publication Date: 2012-09-10). A comprehensive treatment of complex GIS geometries and features. Defines 3 related ontologies: geo: ([namespace](#), [ontology](#)), gml: ([namespace](#), [ontology](#)) and sf: ([namespace](#), [ontology](#)). Defines two datatypes for expressing geometries: geo:WKTLiteral and geo:GMLLiteral.
- **GeoNames**: large gazetteer of geographic places with coordinate info. See [site](#), [ontology doc](#). Defines a comprehensive hierarchy of classes and feature codes, which are comparable to TGN Place Types.
- **DBpedia**: includes a huge amount of information about all kinds of topics. But it has a crowd-sourced ontology definition, which shows some problems. E.g. looking at the class [Place](#), two properties are defined for elevation (altitude and elevation), it is unclear which properties to use for latitude/longitude (if you look at a particular resource e.g. http://live.dbpedia.org/page/Great_Lakes, you see it's geo:lat, geo:long, georss:point), and which to use for bounding box.
- **LOCN**: EC Core Location Vocabulary. Recently submitted to W3C for adoption. See [doc & namespace](#), [ontology](#), [Joinup News](#). Defines properties to connect to GeoSPARQL geometries.
- **FAO Geopolitical**: the UN Food and Agriculture organization has created a comprehensive dataset about countries, economic groupings, etc. See [ontology](#), a [country profile \(US\)](#)
- **CRMGeo**: [CIDOC CRM](#) (ISO 21127) is a foundational ontology in the cultural heritage domain. In particular, it will likely be used for representing the Getty [Cultural Objects Name Authority](#) (CONA) thesaurus. [CRMgeo](#) is a recent extension for integrating GeoSPARQL geometries in CRM: see [specification](#) (version 1.0, created Apr 2013, presented Jun 2013 at CRM-SIG), [presentation](#), [ontology](#). CRMGeo can accommodate complex archaeological scenarios: it considers Spacetime Volumes (i.e. variation of something in time and place, or 4D space); and distinguishes between Phenomenal (defined in relation to some historic/cultural event or thing) and Declarative (defined through some coordinate expressions).
- **NGA/USGS**: the primary source of coordinates in TGN are either of the two large U.S. government databases: [United States Geological Survey \(USGS\)](#) and [National Geospatial-Intelligence Agency \(NGA\)](#) (formerly NIMA). See [ontology documentation](#), [ontology materials](#) (in particular [OWL](#) and [vocabulary](#))

Due to scoping considerations, this initial version of TGN includes only WGS and Schema.org. We welcome your feedback about which other geographic ontologies we should include in the TGN representation, with some justification.

1.8.7.1 W3C WGS Geo Ontology

The W3C Geo Ontology (often called WGS84) is the least common denominator for geographic info, see [document](#). Because of its simplicity, it has found wide use. It defines:

- Class wgs:SpatialThing: anything that can have a location
- Properties wgs:lat (latitude), wgs:long (longitude) and wgs:alt (altitude).

1.8.7.2 Schema.org Geographic Features

Schema.org defines common classes and properties to be used on web pages, and is supported by the big 4 search engines: Google, Yahoo, Microsoft, and Yandex. It defines geography-related classes, making a distinction between a place ([schema:Place](#)) and its geometry ([schema:GeoCoordinates](#) and/or [schema:GeoShape](#)). We use the following relevant properties:

- schema:geo to connect a place to its geometry
- schema:latitude, schema:longitude, schema:elevation
- schema:box to describe the bounding box of a schema:GeoShape

1.9 GVP Ontology

The GVP ontology includes various classes, properties and individuals (values) used in the mapping. The prefix **gvp:** stands for "Getty Vocabulary Program". We considered using the more telling prefix **getty:** but decided against it because other Getty institutions (e.g. the Getty Museum) may start publishing LOD soon. Most of the classes and properties are applicable not only to AAT, but also to the other GVP vocabularies to be published in the future.

The ontology is documented at the "namespace document" <http://vocab.getty.edu/ontology> and is summarized below. You can also get the ontology in XML/RDF and Turtle, either using content negotiation (see [Semantic Resolution](#)), or using the direct links <http://vocab.getty.edu/ontology.rdf> and <http://vocab.getty.edu/ontology.ttl> respectively. You can also [Explore the Ontology](#) with SPARQL queries.

The context where these classes and properties are used is best seen at [Semantic Overview](#).

- [Subject Types](#): gvp:Facet, gvp:Hierarchy, gvp:GuideTerm, gvp:Concept, gvp:ObsoleteSubject. These are implemented as subclasses of skos:Concept, skos:Collection, iso:ThesaurusArray.
- [Subject Hierarchy](#) relations: gvp:broader, gvp:narrower, gvp:broaderExtended, gvp:narrowerExtended, gvp:broaderPreferred, gvp:broaderPreferredExtended, gvp:broaderNonPreferred, gvp:broaderGeneric, gvp:broaderPartitive, gvp:broaderInstantial, gvp:broaderGenericExtended, gvp:broaderPartitiveExtended, gvp:broaderInstantialExtended. These are analogous to skos:broader and friends, but apply to any subject type, and introduce finer distinctions.
- Properties gvp:prefLabelGVP, gvp:prefLabelLoC: most often these are "parallel to" (used with) skosxl:prefLabel
- Other [Subject](#) properties: gvp:parentString, gvp:parentStringAbbrev,
- [Term Characteristics](#): gvp:termKind, gvp:termDisplay, gvp:termType, gvp:termPOS, gvp:termFlag
- Other [Term](#) relations: gvp:contributorPreferred, gvp:contributorNonPreferred, gvp:contributorAlternatePreferred (sub-properties of dct:contributor); gvp:sourcePreferred, gvp:sourceNonPreferred, gvp:sourceAlternatePreferred (sub-properties of dct:source)
- [Sort Order](#) (applies to Subject, Term, Place Type): gvp:displayOrder
- [Historic Information](#) properties (apply to Subject, Term, Place Type): gvp:historicFlag, schema:startDate, schema:endDate
- Finally, [Associative Relations](#) are defined as sub-properties of skos:related (there is a large number of these)

Acknowledgements:

- We created the [ontology documentation \(namespace document\)](#) with [Parrot](#). Note that going to the URL of a class/property (e.g. http://vocab.getty.edu/ontology#aat2000_related_to) will jump directly to the definition of that class/property. The documentation also includes embedded RDFa "about" attributes.
- We intend to validate the ontology using [OOPS! \(the Ontology Pitfall Scanner!\)](#).

2 Semantic Representation

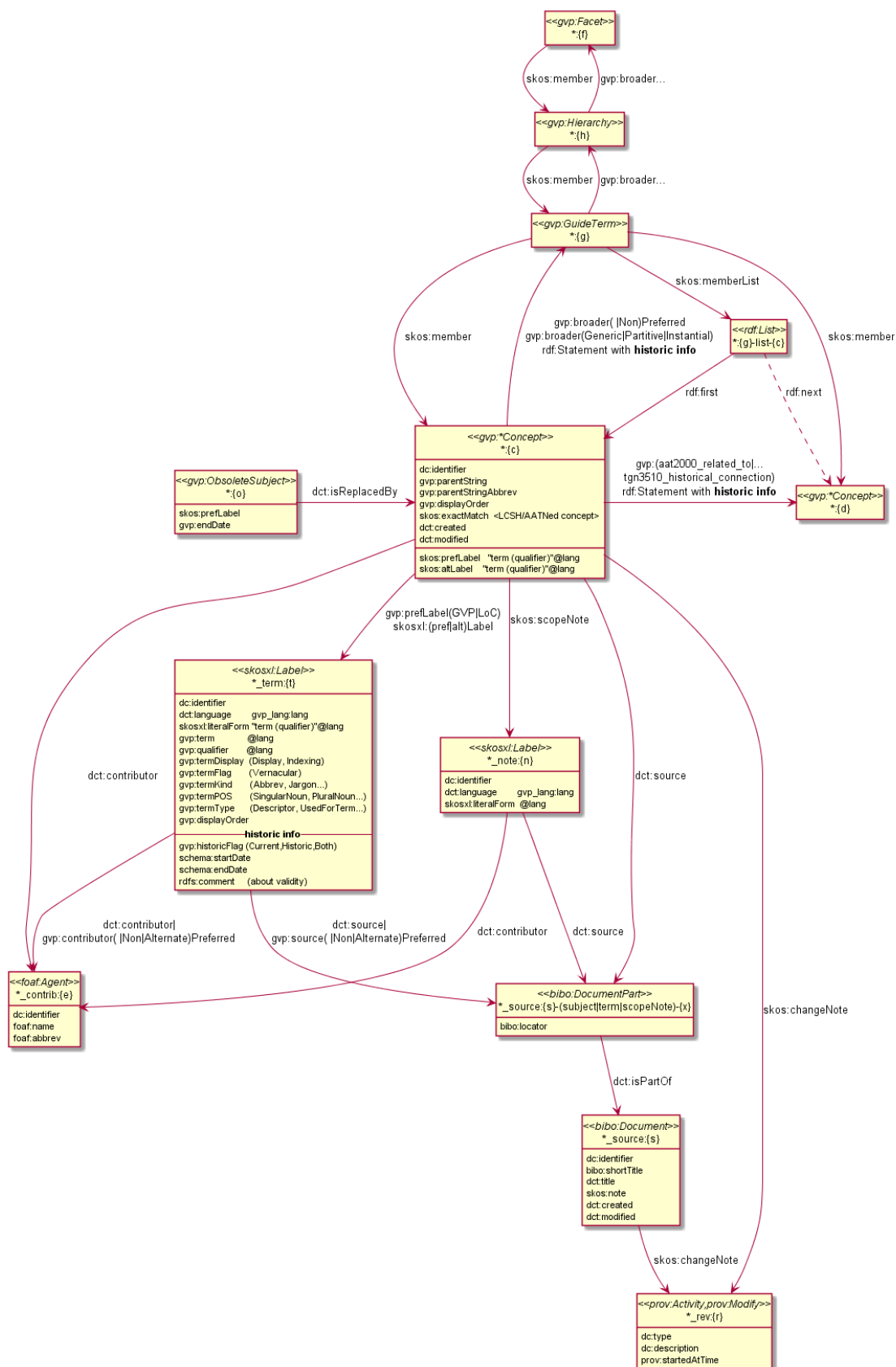
All the information in this section applies equally to all GVP vocabularies. This bears repeating: the basic thesaurus structure of all GVP vocabularies is **the same**. AAT uses only this common structure, while the other vocabularies add some specific data (e.g. see [TGN Specifics](#)).

This principle of representing various kinds of authority info using a common structure is widely used in large libraries. For example the [Linked Data Service](#) of the German National Library (DNB) uses such approach. See the [Ontology of the Integrated Authority File \(GND\)](#) and the document [Linked Data Service of the German National Library: Modelling of bibliographic data](#)

2.1 Semantic Overview

The diagram below provides an overview of the semantic representation.

- Classes are shown like this: `<<gvp:Facet>>`. * is a wildcard, e.g. gvp:*Concept stands for gvp:Concept, gvp:PhysPlaceConcept, gvp:AdminPlaceConcept, or gvp:PhysAdminPlaceConcept
- URLs are shown below the classes, where {x} indicates a numeric ID. * is a wildcard that stands for aat or tgn.
- (...|...) indicates a choice, e.g. gvp:broader(|Non)Preferred stands for gvp:broaderPreferred or gvp:broaderNonPreferred



The diagram shows pretty much everything but glosses over some details:

- [Subjects](#) are the main entities of GVP vocabularies. They have various [Subject Types](#) including [Obsolete Subject](#) (details are glossed over in the diagram).
 - They are implemented using the standard types `skos:Collection`, `iso:ThesaurusArray`, `skos:Concept`
 - `skos:OrderedCollection` and `rdf:List` are also used when a Subject's children are ordered (see [Sorting with Thesaurus Array](#))
- Subjects (except Obsolete) have the following info (exactMatch and Associative Relations apply to Concepts only):
 - Connected to the `aat:` or `tn:` `ConceptScheme` using `skos:inScheme` (not shown on the diagram)
 - `dc:identifier` (see [Identifiers](#))
 - `gvp:parentString` and `gvp:parentStringAbbrev`
 - `gvp:displayOrder` (see [Sort Order](#))
 - [Historic Information](#)
 - `skos:exactMatch` to other thesauri (see [Alignment](#))
 - `gvp:prefLabelGVP`, `gvp:prefLabelLoC`, `skosxl:prefLabel` or `skosxl:altLabel` links to [Terms](#); and `skos:prefLabel`, `skos:altLabel` as dumbed-down versions of these links ([SKOS-XL Inference](#))
 - `skos:scopeNote` links to [Scope Notes](#)
 - `dct:source` links to [Source](#) (can be to a `LocalSource` as shown, or directly to a global Source)
 - `dct:contributor` links to [Contributor](#)
- [Subject Hierarchy](#) relations come in several varieties: Preferred|NonPreferred and Generic|Partitive|Instantial (details are glossed over in the diagram).
 - They can be accessed uniformly through [GVP Hierarchical Relations](#). Appropriate closures (`gvp:broaderExtended`, `gvp:broaderPreferredExtended`, etc) are provided.
 - They are implemented using [Standard Hierarchical Relations](#) (SKOS and ISO). Transitive closures (e.g. `skos:broaderTransitive`) are also provided, but you should use the GVP appropriate closures instead. The ISO BTG, BTP, BTI relations (i.e. `iso:broader(Generic|Partitive|Instantial)`) are provided between `skos:Concepts`.
- [Associative Relations](#) (numbered properties like `gvp:aat2000_*` or `gvp:tn3510_*`) provide lateral relations between subjects.
 - Both Hierarchical and Associative relations may carry [Historic Information](#) attached to a `rdf:Statement`
- [Obsolete Subjects](#) provide some continuity to clients who have used such subjects in their data. They have little info: only `skos:prefLabel`, `schema:endDate` (when it was discontinued), and `dct:isReplacedBy` (if it was merged to another subject).
- [Terms](#) provide multilingual subject labels and carry the following information:
 - `dc:identifier` (see [Identifiers](#))
 - Literals: `gvp:term`, `gvp:qualifier` and `skosxl:literalForm` (being "term (qualifier)", or equal to term if there's no qualifier). All these have the same language tag (`@lang`)
 - `dct:language` ([Language](#)). Coincidentally, if the language is `gvp_lang:<lang>`, then the URL of the Term is `*_term:<identifier>-<lang>`.
 - Enumerated [Term Characteristics](#): `termDisplay`, `termFlag`, `termKind`, `termPOS` and `termType`
 - `gvp:displayOrder` (see [Sort Order](#))
 - [Historic Information](#)
 - Links to [Source](#) (can be to a `LocalSource` as shown, or directly to a global Source). These links can be plain `dct:source`; or sub-properties thereof, specifying whether the term is Preferred, NonPreferred or AlternatePreferred for the source
 - Links to [Contributor](#). Similar to Source links, these can be plain `dct:contributor`, or sub-properties thereof
- [Scope Notes](#) provide multilingual subject definitions and carry the following information:
 - `dc:identifier` (see [Identifiers](#))

- Literal: rdf:value with a language tag (@lang)
- dct:language ([Language](#)).
- dct:source links to [Source](#) (can be to a LocalSource as shown, or directly to a global Source)
- dct:contributor links to [Contributor](#)
- [Contributors](#) are foaf:Agents and have this info: dc:identifier, foaf:name and foaf:abbrev
- [Sources](#) (bibo:Documents) have this info: dc:identifier, bibo:shortTitle, dc:title and skos:note
 - When an entity (Subject, Term or Scope Note) is cited in a particular spot in a source, then we have a Local Source (bibo:DocumentPart) with bibo:locator giving the spot, and dct:isPartOf pointing to the global source

2.2 Subject

Subjects are the main entities (units of thought) in the GVP vocabularies. Subjects (except Obsolete) have the following info, described in the respective sections:

- Subjects are connected to the aat: ConceptScheme using skos:inScheme.
- [Subject Hierarchy](#): threads the subjects using custom (gvp:) properties. Standard skos: and iso: properties are also provided
- [Associative Relations](#): apply to Concepts only
- dc:identifier (see [Identifiers](#))
- gvp:displayOrder (see [Sort Order](#))
- [Historic Information](#) about historic applicability
- skos:exactMatch to other thesauri (see [Alignment](#)): applies to Concepts only
- skos:scopeNote links to [Scope Notes](#)
- dct:source links to [Source](#) (can be to a LocalSource as shown, or directly to a global Source)
- dct:contributor links to [Contributor](#)

Subjects also have these specific properties:

Property	range	Definition
gvp:prefLabelGVP	skosxl:Label	Term preferred by the Getty Vocabulary Program. The language is usually English. Applicable to AAT, ULAN, TGN. Most often used with skosxl:prefLabel
gvp:prefLabelLoC	skosxl:Label	Term preferred by Library of Congress, thus used for cataloging according to AACR2. Applicable to AAT and ULAN. Most often used with skosxl:prefLabel
skosxl:prefLabel	skosxl:Label	Preferred term (descriptor)
skosxl:altLabel	skosxl:Label	Non-preferred term (AlternateDescriptor or UsedForTerm)
gvp:parentString	literal	Preferred labels of the subject's preferred ancestors, listed bottom up. Useful to show the subject's full context. E.g. for 300226882 "baking dishes" is: "bakeware, <vessels for cooking food>, <containers for cooking food>, <culinary containers>, <containers by function or context>, containers (receptacles), Containers (Hierarchy Name), Furnishings and Equipment (Hierarchy Name), Objects Facet"
gvp:parentStringAbbrev	literal	Same but skips middle levels for brevity. E.g. for 300226882 "baking dishes" is: "bakeware, <vessels for cooking food>, ... Furnishings and Equipment (Hierarchy Name)"

Subjects also have data properties skos:prefLabel, skos:altLabel as dumbed-down versions of skosxl:prefLabel and skosxl:altLabel (see [SKOS-XL Inference](#)).

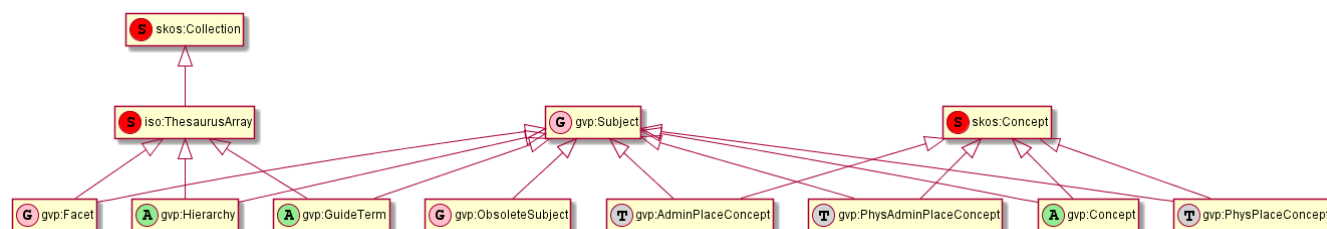
2.2.1 Subject Types

GVP uses different types of subject to construct the levels of the [Subject Hierarchy](#):

Type (Vocab)	Definition	Example
Facet	One of the major divisions of a vocabulary	Objects Facet (AAT), World (TGN)
Hierarchy Name (AAT)	The top of a hierarchy. Not used for indexing or cataloguing.	Containers (Hierarchy Name)
Guide Term (AAT)	Place holder to create a level in the hierarchy. Not used for indexing or cataloguing.	<vessels for serving and consuming food>
Concept (AAT)	Proper concept. Used for indexing and cataloguing.	rhyta
Physical Place Concept (TGN)	Physical feature, defined by its physical characteristics on planet Earth, including mountains, rivers, and oceans	Amazon River
Administrative Place Concept (TGN)	Place defined by administrative boundaries and conditions, including inhabited places, nations, and empires	Burgundy region in France
Physical and Administrative Place Concept (TGN)	Place that is both administrative and physical. Rarely used	Kiik-Koba
Obsolete Subject	Obsolete: moved out of the publishable hierarchy, or merged to another	300375205 "shrank" was merged to 300039264 "schranks" (reflected by dct:isReplacedBy)

We have introduced GVP classes for each type, and arranged them in the following class hierarchy. Legend:

- (S): standard classes (SKOS or ISO 25964)
- (G): common GVP classes
- (A): AAT-specific classes
- (T): TGN-specific classes



- All are subclasses of **gvp:Subject**, to allow the user to find all GVP subjects with a single query
- gvp:Facet, gvp:Hierarchy and gvp:GuideTerm are implemented as a subclass of **iso:ThesaurusArray** (which itself is a subclass of **skos:Collection**) since they can hold other subjects, but cannot be used for indexing
- gvp:*Concept are implemented as a subclass of **skos:Concept**, since they are used for indexing
- gvp:ObsoleteConcept is not a subclass of any standard class, since normal thesaurus operations should NOT use nor display them

When a Subject's children are ordered, it is also declared **skos:OrderedCollection** (see [Sorting with Thesaurus Array](#))

2.3 Subject Hierarchy

The GVP hierarchy includes subjects other than Concepts (see [Subject Types](#)); distinguishes between a Preferred parent and optional NonPreferred parents, and makes distinctions between broaderGeneric, broaderPartitive and broaderInstantial.

There is a bunch of hierarchical relation properties, which we explain in detail below, including which properties are derived from which others. Then we explain how the properties are used across the [Hierarchy Structure](#), and which property is used for which subject types.

2.3.1 Standard Hierarchical Relations

SKOS and [ISO 25964](#) provide a number of hierarchical relations. We use the following (**d** shows the direction up/down):

Relation	d	Domain	Range	Description
skos:broader	▲	skos:Concept	skos:Concept	Parent concept of a concept
iso:broaderGeneric	▲	skos:Concept	skos:Concept	Parent in the case of Genus/Species relation
iso:broaderPartitive	▲	skos:Concept	skos:Concept	Parent in the case of Part/Whole relation
iso:broaderInstantial	▲	skos:Concept	skos:Concept	Parent in the case of Kind/Instance relation
skos:broaderTransitive	▲	skos:Concept	skos:Concept	Ancestor concepts (transitive version of broader)
iso:superOrdinate	▲	iso:ThesaurusArray	skos:Concept	Parent concept of array
skos:narrower	▼	skos:Concept	skos:Concept	Children concepts of a concept
skos:narrowerTransitive	▼	skos:Concept	skos:Concept	Descendant concepts (transitive version of narrower)
skos:member	▼	iso:ThesaurusArray	skos:Concept	Children concepts/arrays of array. See skos:member Structure for an illustration. skos:memberList is also used if the array is ordered, see skos:memberList Structure
iso:subordinateArray	▼	skos:Concept	iso:ThesaurusArray	Children arrays of a concept

We decided to remove all struck-out properties for the reasons described in [Reduced SKOS Inference](#)

2.3.2 GVP Hierarchical Relations

The standard relations have certain limitations:

- Different properties are used for different nodes in the hierarchy ([Subject Types](#)), which does not allow you to access the hierarchy uniformly
- iso:broaderGeneric, iso:broaderPartitive, iso:broaderInstantial (commonly known as BTG, BTP, BTI) are important distinctions of the broader relation. These apply to skos:Concept only, but GVP needs to use them for other Subject Types as well.
- These ISO properties are declared sub-properties of skos:broader, which itself is a sub-property of skos:broaderTransitive, so all these relations unconditionally contribute to skos:broaderTransitive. However, composing these relations is not appropriate in some cases, e.g. *Sofia BTP Bulgaria BTI country*, but *Sofia BTI country* is false: *Sofia BTI city* (or *inhabited place*), and there is no broader relation between *city* and *country*. So skos:broaderTransitive is not meaningful in these cases.

To overcome these limitations, we introduce a number of custom (GVP) relations, all of which are defined across all subject types (i.e. domain and range is gvp:Subject). The paper [Compositionality of ISO 25964 Hierarchical Relations](#) (V.Alexiev, J.Lindenthal, A.Isaac, May 2014, submitted) discusses the last point (meaningful closure) in detail:

- Introduces "Extended" relations (gvp:broaderGenericExtended, gvp:broaderPartitiveExtended, gvp:broaderInstantialExtended, denoted for brevity BTGE, BTPE, BTIE), meant to be meaningful closures of BTG, BTP, BTI.
- Proposes specific compositionality rules to derive the Extended relations as specific chains, see [BTG, BTP, BTI Inference](#).
- Defines gvp:broaderExtended as a disjunction of BTGE, BTPE, BTIE.
- Derives the ISO BTG, BTP, BTI relations from those BTGE, BTPE, BTIE that connect skos:Concepts directly, see [ISO Insert Queries](#) and [ISO Rules](#).

(**d** shows the direction up/down)

Relation	d	Name	Description
<code>gvp:broader</code>	▲	Parents	Each broader is also Preferred NonPreferred and Partitive Instantial Generic
<code>gvp:narrower</code>	▼	Children	Inverse of <code>gvp:broader</code>
<code>gvp:broaderPreferred</code>	▲	Preferred Parent	Main parent, e.g. <i>baking dishes</i> BTG <i>bakeware</i> or <i>Sofia</i> BTP <i>Bulgaria</i> .
<code>gvp:broaderNonPreferred</code>	▲	Non-Preferred Parents	Auxiliary parents, e.g. <i>baking dishes</i> BTG <i>dishes (vessels)</i> and <i>Embrun</i> BTP <i>Alpes Cottiae</i> . There may be several non-preferred parents.
<code>gvp:broaderGeneric</code>	▲	Parent (Generic)	BTG (Genus/Species, "is a") relation, e.g. <i>calcite</i> (AAT) BTG <i>mineral</i> (AAT)
<code>gvp:broaderPartitive</code>	▲	Parent (Partitive)	BTP (Part/Whole, "part of") relation, e.g. <i>Tuscany</i> (TGN) BTP <i>Italy</i> (TGN)
<code>gvp:broaderInstantial</code>	▲	Parent (Instantial)	BTI (Kind/Instance, "example of") relation, eg <i>Rembrandt van Rijn</i> (ULAN) BTI <i>Painter</i> (AAT)
<code>gvp:broaderGenericExtended</code>	▲	Ancestors (Generic)	Meaningful closure of <code>gvp:broaderGeneric</code>
<code>gvp:broaderPartitiveExtended</code>	▲	Ancestors (Partitive)	Meaningful closure of <code>gvp:broaderPartitive</code>
<code>gvp:broaderInstantialExtended</code>	▲	Ancestors (Instantial)	Meaningful closure of <code>gvp:broaderInstantial</code>
<code>gvp:broaderExtended</code>	▲	Appropriate ancestors	Meaningful closure of <code>gvp:broader</code> for query expansion. Use this, not <code>skos:broaderTransitive</code>
<code>gvp:narrowerExtended</code>	▼	Appropriate descendants	Meaningful closure of <code>gvp:narrower</code> for query expansion. Use this, not <code>skos:narrowerTransitive</code>
<code>gvp:broaderPreferredExtended</code>	▲	Preferred Ancestors	Meaningful closure of <code>gvp:broaderPreferred</code>

We decided to remove all struck-out properties for the reasons described in [Reduced SKOS Inference](#)

2.3.3 Hierarchy Structure

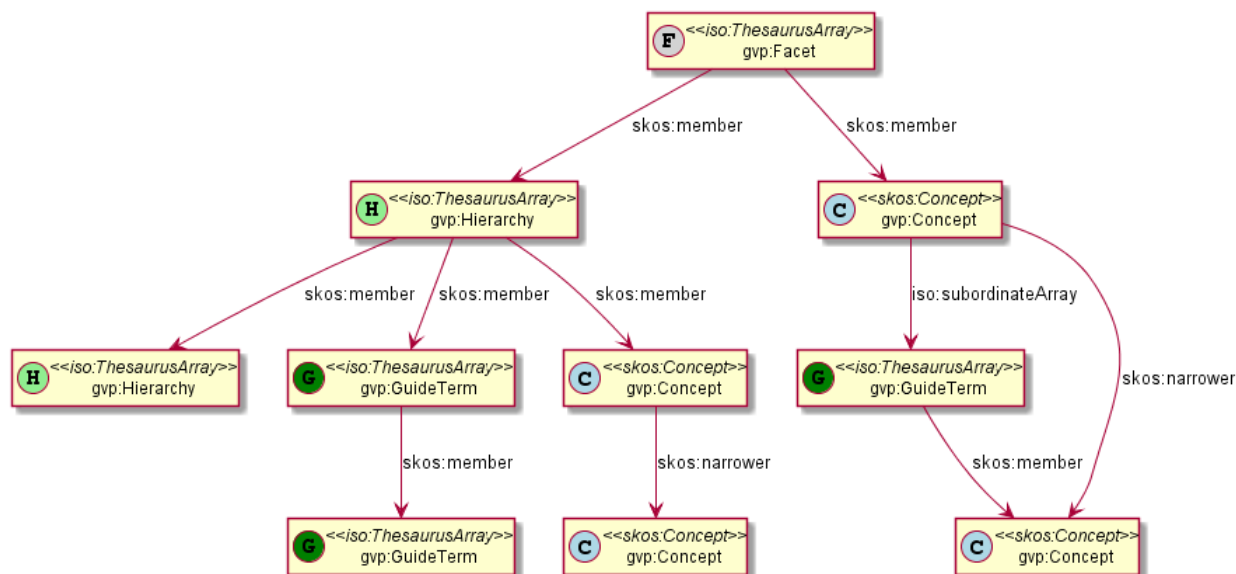
The structure of the subject hierarchy can be symbolized as **F>H>G&C**, i.e.

- Facets are above Hierarchies
- Hierarchies are above Guide Terms and Concepts
- Guide Terms and Concepts can be intermixed. There are many examples of G under C. (Note: during vocabulary evolution, G sometimes transitions to C)

The following table shows for each Subj1, what Subj2 can be nested under it (Facet cannot be nested under anything). In each cell we give an example of Subj2, and the linking [Standard Hierarchical Relations](#) (e.g. Concept→`iso:subordinateArray`→ GuideTerm).

Subj1\Subj2	Hierarchy	Guide Term	Concept
Facet	Living Organisms <code>skos:member</code>		agents (general) <code>skos:member</code>
Hierarchy	Visual Works (Hierarchy) <code>skos:member</code>	<organisms by activity> <code>skos:member</code>	visual works <code>skos:member</code>
Guide Term		<Early Western World> <code>skos:member</code>	Mediterranean (Early Western World) <code>skos:member</code>
Concept		<ancient Italian styles and periods> <code>iso:subordinateArray</code>	Old Hittite Kingdom <code>skos:narrower</code>

The following diagram illustrates the same nesting possibilities:

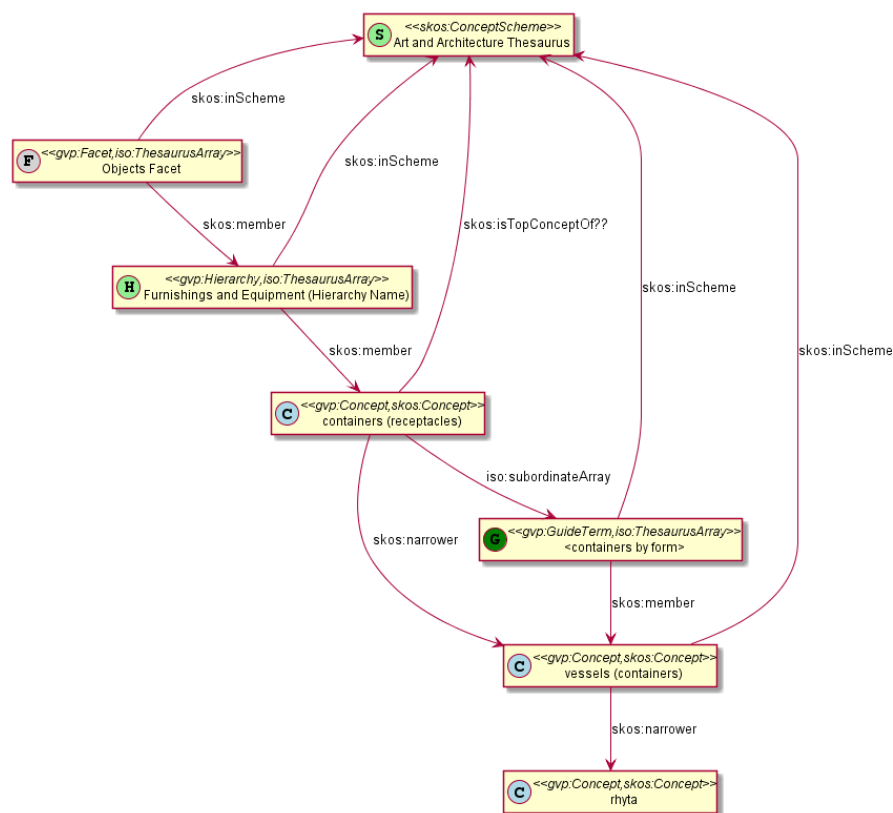


You can see that the standard representation is not uniform (uses different properties in the different cases). You may also notice the right-most link `skos:narrower`: although it's not in the original hierarchy, we insert it, so as to thread the Concepts through the hierarchy. We call this a "thread-through" `skos:narrower`.

Keep in mind that adjacent nodes are also connected uniformly by [GVP Hierarchical Relations](#) (e.g. `gvp:narrower` going down). Because `skos:narrower` connects only Concepts but "jumps levels" as in the example above, it is neither a sub-property nor a super-property of `gvp:narrower`.

2.3.4 Top Concepts

Consider this example from the AAT hierarchy. Legend: S=scheme, F=facet, H=hierarchy, G=guide term, C=concept. (Some levels are skipped for brevity, inverses and inferred properties are not shown). Every subject in the hierarchy has a link to the AAT ConceptScheme: `skos:inScheme aat`:



Consider the Concept "containers (receptacles)": it has a thread-through `skos:narrower` to "vessels (containers)". Furthermore, it is a Top Concept, because there are only Facets, Hierarchies and Guide Terms above it.

Many SKOS thesauri use `skos:topConceptOf` (a sub-property of `skos:inScheme`) to link such Concepts to the ConceptScheme. Even flat concept lists do that, e.g. LoC's MARC Relators are declared `topConceptOf` <http://id.loc.gov/vocabulary/relators> (see [HTML](#) and [RDF](#)).

In a previous version of AAT we did the same, but it has some counter-intuitive consequences. E.g. [aat:300054031](#) "drawing (metalworking)" is a top concept, although it's nested 8 levels deep:

- <metal forming processes and techniques>, <metalworking processes and techniques>, <metalworking and metalworking processes and techniques>, <processes and techniques by material>, <processes and techniques by specific type>, <processes and techniques>, Processes and Techniques, Activities Facet

The reason is that none of its ancestors is a concept: going up, there are 6 Guide Terms, then a Hierarchy, and finally a Facet. `skos:hasTopConcept` (the inverse of `skos:topConceptOf`) is meant as a navigation-enabler, to provide "entry points" in the hierarchy (see [SKOS Primer 2.5 Concept Schemes](#)). In AAT the Facets are such entry points, and it is dubious that top concepts would be useful for this purpose. Using `skos:hasTopConcept` will trick SKOS-consuming applications into displaying a number of disconnected "concept hierarchies", which don't really represent the AAT hierarchy. Picking the roots of "concept hierarchies" at random depths in the general AAT hierarchy feels awkward. There are 887 such hierarchies, the majority of which (633) are no hierarchies at all:

- concepts: 38146
- top concepts: 887 (2.3% of all concepts)
- top concepts without children: 633 (71% of the top concepts)

So we finally decided to omit `skos:topConceptOf` and `skos:hasTopConcept` altogether. In lieu of this and to allow RDF crawlers to get all of AAT, we declare all Facets as `void:rootResource` of the AAT dataset (see [Dynamic Descriptive Properties](#))

2.4 Sort Order

Usually, GVP Subjects and Terms are sorted alphabetically (for Subjects, by `gvp:prefLabelGVP`), but in some cases a specific ("forced") ordering is set, e.g. for many subjects in the Periods and Styles facet. Facets and Hierarchies don't have a useful ordering at present, so we ignore it.

[TGN Place Types](#) are sorted by a specific Sort Ordering that is meaningful to researchers. According to [TGN Editorial Guidelines](#):

- The place type in sequence number 1 must be the Preferred place type.
- The other place types are arranged in reverse chronological order, with Current place types placed before Historical ones.
- Within the subset of current or historical place types that date to the same period, the place types are arranged in order of importance.

To accommodate a maximum number of consumers, we map sorting in 3 ways (belt, suspenders, AND a piece of string!)

1. With a custom property `gvp:displayOrder` (`xsd:positiveInteger`). You can use that with "order by" in SPARQL.
2. OWLIM preserves the order of nodes as they are **first** inserted in the repository, and returns them in the same order in result sets. We have been careful to load, Subjects, Terms and Place Type relations in the desired order, so you should also get them in this order, if you don't specify an "order by" clause. Note: there is no guarantee of this behavior, and no W3C standard that mandates it.
3. Using `skos:OrderedCollection` and `iso:ThesaurusArray` for Guide Terms and Concepts (see the next section)

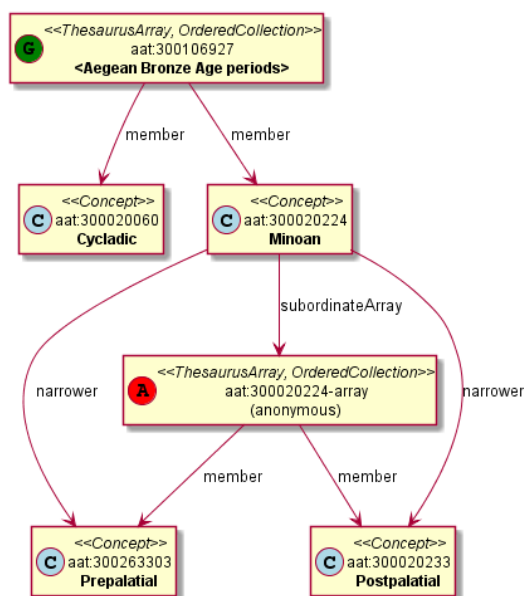
2.4.1 Sorting with Thesaurus Array

`skos:OrderedCollection` defines a standard way to order its children. In addition to `skos:member`, this uses `skos:memberList`, being an `rdf:List` (see section [SKOS member vs memberList](#) for details). `iso:ThesaurusArray` borrows the same paradigm. We illustrate the representation using two examples from the Periods and Styles facet:

- GuideTerm 300106927 <Aegean Bronze Age periods> and Concept 300020224 "Minoan"
- (Note: at present these don't have forced sort order, so instead explore 300018774 <Siberian periods>)

2.4.1.1 *skos:member* Structure

- The Guide Term (G) is represented as an `iso:ThesaurusArray` with `skosxl:Label` "<Aegean Bronze Age periods>"
- The Concept (C) "Minoan" is represented as `skos:Concept`. But it also has an `iso:subordinateArray` (A), that is an `iso:ThesaurusArray`, is *anonymous*, and serves only to hold the `skos:OrderedCollection`. *The anonymous array should not be displayed as a level in the hierarchy.*



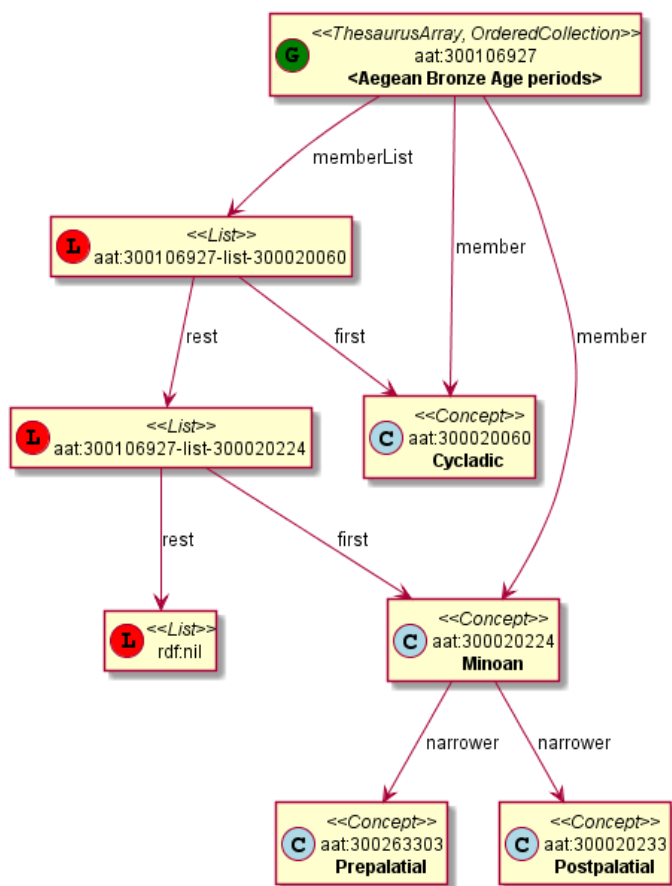
2.4.1.2 *skos:memberList* Structure

While the *skos:member* links establish collection membership (used both with unordered *skos:Collection* and ordered *skos:OrderedCollection*), additional *skos:memberList* structure provides the ordering of the members.

This uses the standard *rdf:List* construct. People often use blank nodes for *rdf:List* elements (and there is a special Turtle shortcut for this), but we use explicit URIs since this way it's easier to thread the list with the tooling used (R2RML).

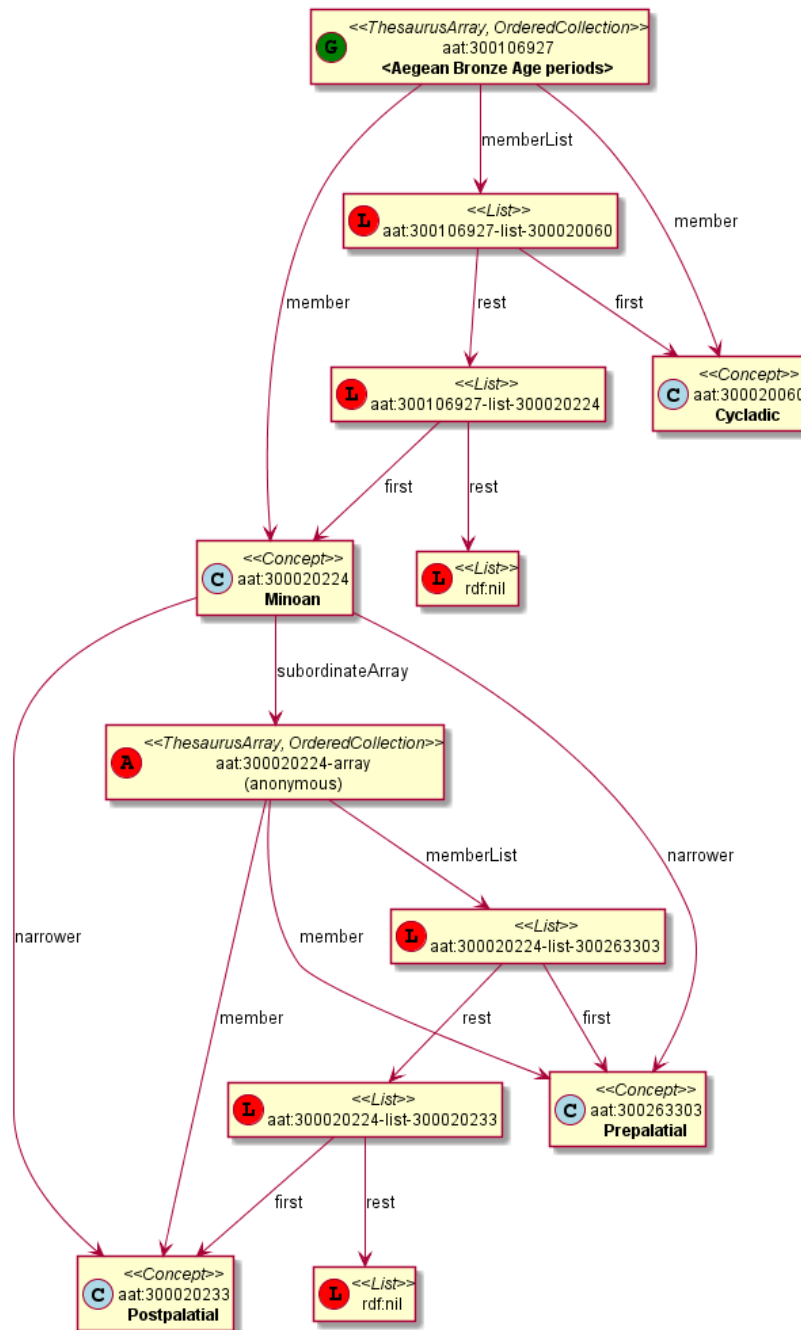
The list URIs have the form **{p}-list-{c}** where {p} is the parent ID (owner of the list) and {c} is the child ID corresponding to the current list element.

For pedagogical purposes, we first show only the list of (G) <Aegean Bronze Age periods>. See the next section for the list of (C) "Minoan" (i.e. the full representation).



2.4.1.3 Full Representation

Adding the skos:memberList of (C) "Minoan", we get the full representation (not for the faint of heart!)



2.5 Associative Relationships

GVP defines a plethora of associative relations between concepts:

- All associative relations are sub-properties of **skos:related**.
- Relations come in pairs of forward-inverse relation; symmetric relations are self-inverses.
- Every relation **instance** also has an inverse instance. i.e. if "A rel B" then there is relation "B inv(rel) A".
- We declare pairs as **owl:inverseOf** (and symmetric relations as owl:SymmetricProperty), but this won't infer new statements.

2.5.1 Relationships Table

The [Relationships Table](#) (in PDF) describes the associative relationships used for AAT. If you'd like it in Excel format, please contact us. Legend:

- fcode=code of forward relation, icode=code of inverse relation
- domain=source (of forward relation), range=target (of forward relation); i.e. the kind of Concepts that it can connect
- frel=forward relation name, irel=inverse relation name,
- fdef=forward relation definition, idef=inverse relation definition,
- fexample=example of forward relation, iexample=example of inverse relation.

2.5.2 Relationship Cross-Walk

The [Relationship Cross-Walk \(matrix\)](#) (PDF) shows applicability of relations to kinds of Concepts (domain/range). To use:

- Begin with the vertical column on the left,
- Find intersection of Concept in vertical row with another Concept in horizontal column,
- See what relations are applicable

2.5.3 Relationship Representation

All info from the [Relationships Table](#) is represented in RDF.

- Property URLs are generated as **gvp:<vocab><code>_<name>** where spaces in the name are replaced with "_".
- All are sub-properties of skos:related. (This was changed recently because there were a few relation instances that connected non-Concepts)
- rdfs:domain and rdfs:range are declared as skos:Concept, whereas the "application" domain & range are emitted as comments only.
- <code> is emitted as dc:identifier
- "<name> - <range>" is emitted as dc:title
- Examples are emitted as multiple skos:example. We think these add a lot of clarity to the meaning of relations.
- dct:description includes <domain> - <name> - <range> and the examples

```
gvp:aat2208_locus-setting_for a owl:ObjectProperty;
  rdfs:subPropertyOf skos:related;
  rdfs:domain skos:Concept; rdfs:range skos:Concept;
  # domain "locus/setting"; range "things";
  dc:identifier "2208";
  skos:prefLabel "aat2208_locus-setting_for";
  dc:title "locus/setting for - things";
  skos:example "glassworks (buildings) are the locus/setting for glassware",
    "caves are the locus/setting for cave paintings" ;
  dct:description ""locus/setting - [is] locus/setting for - things.
Example: glassworks (buildings) are the locus/setting for glassware;
caves are the locus/setting for cave paintings"" .
```

```
gvp:aat2209_use-located_in a owl:ObjectProperty;
  rdfs:subPropertyOf skos:related;
  rdfs:domain skos:Concept; rdfs:range skos:Concept;
  # domain "things"; range "locus/setting";
  dc:identifier "2209";
  skos:prefLabel "aat2209_use-located_in";
  dc:title "use/located in - locus/setting";
  skos:example "glassware is used/located in glassworks (buildings)",
    "cave paintings are located in caves" ;
  dct:description ""things - used/located in - locus/setting.
Example: glassware is used/located in glassworks (buildings); cave paintings are located in caves"" .
gvp:aat2208_locus-setting_for owl:inverseOf gvp:aat2209_use-located_in.
# or owl:SymmetricProperty if self-inverse
```

TGN also defines useful associative relations, which use their own prefix (e.g. gvp:aat2811_preceded relates styles/ periods/ cultures, while gvp:tgn3412_predecessor_of relates nations). For example:

```
gvp:tgn3411_successor_of a owl:ObjectProperty;
  rdfs:subPropertyOf skos:related;
  rdfs:domain skos:Concept; rdfs:range skos:Concept;
  # domain "nation"; range "nation";
  dc:identifier "3411";
  skos:prefLabel "tgn3411_successor_of";
  dc:title "successor of - nation";
  skos:example "Persia is the predecessor of Iran (nation)" ;
  dct:description ""nation - successor of - nation.
Example: Persia is the predecessor of Iran (nation)"" .

gvp:tgn3412_predecessor_of a owl:ObjectProperty;
  rdfs:subPropertyOf skos:related;
  rdfs:domain skos:Concept; rdfs:range skos:Concept;
  # domain "nation"; range "nation";
  dc:identifier "3412";
  skos:prefLabel "tgn3412_predecessor_of";
  dc:title "predecessor of - nation" ;
  dct:description ""nation - predecessor of - nation"" .
gvp:tgn3411_successor_of owl:inverseOf gvp:tgn3412_predecessor_of.
```

The dc:title allows you to construct nice displays. E.g. aat:300025419 "rope-makers" has a relation gvp:aat2292_work-live_in to "roperies". So its display can include:

Label:	rope-makers
Relations: <i>work/live in - locus/setting</i>	roperies

2.6 Obsolete Subject

GVP subjects may become obsolete as a result of editorial actions:

- Set as non-publishable (which basically means "deleted")
- Merged to another subject.

Obsolete concepts may have been used in client data. So in order not to leave such data hanging, we publish minimal information about them:

- **skos:prefLabel**: only the GVP preferred label (usually in English)
- **schema:endDate**: when it was obsoleted
- **dct:isReplacedBy**: merged to which subject

Currently AAT obsolete subjects are 4.4% of valid subjects, which shows a good rate of editorial actions, and the importance of this information. Examples:

```
aat:300123456 a gvp:ObsoleteSubject; # Was made non-publishable
  skos:prefLabel "Made up subject";
  skos:inScheme aat: ;
  schema:endDate "2012-12-31T12:34:56"^^xsd:dateTime.

aat:300386746 a gvp:ObsoleteSubject; # Was merged to a dominant Subject
  skos:prefLabel "Buncheong";
  skos:inScheme aat: ;
  dct:isReplacedBy aat:300018699; # Punch'ong
  schema:endDate "2012-12-31T12:34:56"^^xsd:dateTime.
```

Merged ObsoleteSubjects are mentioned by ID and name as "Recessive" in the "merge" [Revision History](#) action of the "Dominant" subject. We want the inverse relation from the Dominant to the Recessive subject (dct:replaces). DCT doesn't have such declaration, so we add it:

```
dct:isReplacedBy owl:inverseOf dct:replaces.
```

Some would say this is "namespace hijacking". We call it adding info that's missing from DCT.

2.7 Language

GRI has gathered information about a plethora of Languages (some 1800), both ancient and modern. Other cultural heritage institutions have asked GRI to standardize in this area. Although there are several sources of language information (Lingvo, ISO, etc), a lot of the GRI languages cannot be found there (see [GVP Language Tags](#) for details), e.g.:

- Liturgical Greek
- Chinese (transliterated Pinyin without tones)

GVP maintains language data in AAT, as concepts under [300389738](#) <languages and writing systems by specific example>.

- This data will be used uniformly across all GVP vocabularies
- A mapping to IANA language tags is under development. We have covered all languages used in AAT (about 105) and TGN (about 115 more).
- All AAT labels, but only 80% of TGN labels, have a language. See [Importance of the Vernacular Flag](#)
- All Scope Notes have language.

See the query [Languages and ISO Codes](#) to get all language data. See the query [Number of Terms per Language](#) to get language usage information.

2.7.1 IANA Language Tags

RDF literals use language tags as defined in the [IANA Language Subtag Registry](#). Its structure (described in [BCP47 sec 3.1](#)) is not easy to read. So we wrote a script [iana-lang-tags.pl](#) that gets the registry, parses it, and writes it to a tab-delimited file. Open this file in excel and search to your heart's content.

The registry includes almost 9000 registrations (broken down by Type and Scope):

- 7769 languages
- 227 extlangs, e.g. ar-auz (Uzbeki Arabic)
- 116 language collections, e.g. bh (Bihari languages)
- 62 macrolanguages, e.g. zh (Chinese), cr (Cree)
- 4 special languages, e.g. und (Undetermined)
- 162 scripts, eg Latn (Latin), Japn (Japanese)
- 301 regions, e.g. US (United States), 021 (Northern America)
- 61 variants
- 67 redundant
- 26 grandfathered

2.7.2 GVP Language Tags

Despite the richness of IANA tags, we had to define new tags, using several extension mechanisms:

- Private language, e.g.
 - **x-byzantin**-Latn for Byzantine Greek (transliterated)
 - **x-khasian** for Khasian
- Private language used in specific region, e.g.
 - **qqq-002** for African language
 - **qqq-142** for Asian language
 - **qqq-ET** for Ethiopian (not specified which: Boro/Borna, Karo, Male...)
- Private modifier, e.g.
 - grc-Latn-**x-liturgic** for Liturgical Greek
 - ber-Latn-**x-dialect** for Berber Dialects (transliterated)
 - fa-Latn-**x-middle** for Persian, Middle (transliterated)
 - zh-Latn-pinyin-**x-notone** for Chinese (transliterated Pinyin without tones)
 - **x-frisian**. IANA/ISO has codes for its predecessor Old Frisian and its dialects West, Saterland and North Frisian, but not for Frisian itself.

GVP has a language [300389645](#) "Undetermined" (gvp_lang:und). IANA has a corresponding tag **und** "Undetermined", but we decided NOT to emit it for terms: due to the Open World Assumption, unknown/undetermined info does not need to be emitted. So instead of the info on the left, we emit the one on the right:

<pre>aat:300312114 skos:altLabel "bukaġi"@und; skosxl:altLabel aat_term:1000408720-und. aat_term:1000408720-und skosxl:literalForm "bukaġi"@und; gvp:term "bukaġi"@und; dct:language aat:300389645, gvp_lang:und.</pre>	<pre>aat:300312114 skos:altLabel "bukaġi"; skosxl:altLabel aat_term:1000408720. aat_term:1000408720 skosxl:literalForm "bukaġi" ; gvp:term "bukaġi".</pre>
---	--

2.7.3 Language Tag Case

[Term](#) URLs have the language tag appended. But you may have noticed a discrepancy in the case used to spell them, e.g.

<pre>aat_term:1000024386-en-US xl:literalForm "currycombs"@en-us .</pre>
--

This is not an error:

- RFC 5646 (BCP47) section [2.1.1 Formatting of Language Tags](#) demands "At all times, language tags and their subtags, including private use and extensions, are to be treated as case insensitive".
- The SPARQL function [langMatches\(\)](#) treats them case-insensitively

Nevertheless, it's an unpleasant discrepancy, since RFC 5646 continues "consistent formatting and presentation of language tags will aid users. The format of subtags in the registry is RECOMMENDED as the form to use in language tags." In other words, RFC 5646 demands that implementations are case-insensitive, and recommends that they are case-preserving.

Conventionally, language tags are written in lower-case except:

- Script is capitalized, e.g. el-**Latn**
- Region is in uppercase, e.g. en-**US**

You should use [langMatches\(\)](#) to compare language tags (see [Find Terms by Language Tag](#)), or spell the tag in the exact case used in the repository (e.g. el-latn and en-us).

We undertook some steps to get RDF vendors to normalize case as recommended by RFC 5646, or at least preserve it:

- [Posted bug](#) against Perl's RDF::Trine::Node::Literal (Sep 2013), which is used in the R2RML tool that we use (RDF2RDB)
- Posted [lang_normalize routine](#) that can be reused by other vendors as well
- Perl [fix adopted and implemented](#) on Github (Jan 2014)

- Posted bug against Sesame (model and RIO): [SES-1999](#), [SES-1659](#) (Jan 2014), which is still open
- [Proposed change](#) to the:
 - [RDF 1.1 standard](#): "Lexical representations of language tags MAY be normalized, according to BCP47 section 2.1.1. "Formatting of Language Tags" (country codes in upper case, script codes capitalized, the rest in lower case). Language tags MAY also be normalized by converting all to lower case, but BCP47 normalization is preferred"
 - [SPARQL lang\(\)](#) function: "lang() MAY normalize the language tag as described in RDF 1.1 Concepts and Abstract Syntax sec 3.3 Literals. It is recommended that lang() normalizes the literal according to BCP47 section 2.1.1, and not by converting it all to lower case."

Jena appears to [store the lang tag as provided](#), which is better than storing as lowercase

2.7.4 Language Tags and Sources

Language data includes names in several languages, ISO language codes, IANA language tags. E.g. see [300389115](#) Portuguese:

- Language names (skos:prefLabel): "Portuguese (language)"@en, "portugais"@fr, "Portugiesisch"@de
- Language codes/tags (skos:altLabel): "por"@en, "pt"@en

You can find out where the tags come from by exploring the term's [Source](#):

- The skos:altLabel **"pt"**@en corresponds to the skosxl:altLabel aat_term:1000576998-en that has dct:source aat_source:2000075479, which is "ISO 639-1 Alpha-2 codes for names of languages"
- The skos:altLabel **"por"**@en corresponds to the skosxl:altLabel aat_term:1000576997-en that has dct:source aat_source:2000075493, which is "ISO 639-2 Alpha-3 codes for names of languages".

Notes:

- Previously we had an intermediate bibo:DocumentPart node that was dct:isPartOf aat_source:2000075479 (respectively aat_source:2000075493), with a constant bibo:locator saying "ISO 2-character". We considered this a parasitic node, so we have now removed it
- **"pt"** and **"por"** are not English words, so using @en for them is (strictly speaking) not correct. In the future we may introduce special lang tags @x-iso6392, @x-iso6393, @x-iana to mark the language tags with.
- Unfortunately the GVP language tags are not published currently (the query [Languages and ISO Codes](#) returns iso2 and iso3 codes, but not the GVP custom tags). Let us know if you need them published. In the meantime, the query [Language URLs](#) returns all GVP languages, and the tag is the last part of the URL.

2.7.5 Language Dual URLs

AAT languages with assigned language tag have dual URLs, e.g. for Portuguese:

- [aat:300389115](#): "systemic" URL of the language as a concept in the Languages hierarchy.
- [gvp_lang:pt](#): "logical" URL that includes the language tag is the last part of the URL. These are used as dct:language in Terms (skosxl:prefLabel, skosxl:altLabel) and Notes (skos:scopeNote). Coincidentally, the URLs of Terms also include the language tag, e.g.

```
aat_term:1000011071-pt
  skosxl:literalForm "asbestos"@pt;
  dct:language gvp_lang:pt.
```

The dual language URLs are declared owl:sameAs, e.g. <http://vocab.getty.edu/aat/300389115.ttl> has this triple:

```
aat:300389115 owl:sameAs gvp_lang:pt.
```

You can access the language data using either URL. The query [Language URLs](#) returns all GVP languages.

Why didn't we use established language URLs, e.g. from [Lexvo](#)? (For example, the [Lingvoj](#) site has started using Lexvo URLs in 2010, in a spirit of reuse and cooperation). We would be glad to, but many of the GVP languages are specific so we had to use custom tags. See [GVP Language Tags](#) for examples.

2.8 Term

GVP includes multilingual terms and rich information about them, including preferred status, sources, contributors, revision history, etc. We use SKOSXL (skosxl:Label) for the full information, and plain SKOS (literal labels) to allow SKOS-only clients to access the literals (see [SKOS and SKOS-XL](#)). Please note that each Term is owned by exactly one Subject (i.e. is dependent on the subject), whereas SKOS-XL potentially allows one xl:Label to be reused between Concepts (eventually in different roles).

Terms carry the following information:

- **dc:identifier**: numeric ID, also used in the term URL. See [Identifiers](#)
- **gvp:term**: the proper (own) label, e.g. "rhea"
- **gvp:qualifier**: serves to clarify and disambiguate terms with the same spelling but different meaning, e.g. "vessels" vs "species" (AAT) or "Republic of", "ancient", "historical" (TGN)
- **skosxl:literalForm**: concatenation of term and parenthesized qualifier, e.g.
 - "rhea (vessels)" meaning "rhyta" (a kind of drinking vessel) vs
 - "rhea (species)" meaning "Boehmeria nivea" (Chinese grass)
- **dct:language** see [Language](#). If the language is gvp_lang:<lang>, then the URL of the Term is aat_term:<identifier>-<lang>, and the language tag of the previous 3 properties is @<lang>
- **gvp:displayOrder**, see [Sort Order](#)
- [Historic Information](#) about historic applicability of the term
- Links to [Source](#)
 - Can be to a global source (bibo:Document), or to a local source (bibo:DocumentPart)
 - Can be plain dct:source; or sub-properties thereof (gvp:sourcePreferred, gvp:sourceNonPreferred, gvp:sourceAlternatePreferred), specifying whether the term is Preferred, NonPreferred or AlternatePreferred for the source
- Links to [Contributor](#).
 - Similar to Source links, these can be plain dct:contributor, or sub-properties thereof (gvp:contributorPreferred, gvp:contributorNonPreferred, gvp:contributorAlternatePreferred)

2.8.1 Term Characteristics

Terms have a number of enumerated characteristics. A lot of these are optional, i.e. could be missing.

- **gvp:termDisplay**: Use for Display (where natural order is preferred) or Indexing/lists (where inverted order is appropriate)?
- **gvp:termFlag**: Vernacular, Loan Term
- **gvp:termKind**: Abbreviation;
 - Common term, Chemical Name, Full term, Jargon or Slang, Neologism, Scientific or Technical term (AAT);
 - FIPS Code, ISO alpha-2 code, ISO alpha-3 code, ISO numeric-3 code, Official Name, Pseudonym, Provisional Name, Site Name, US Postal Service Code (TGN)
- **gvp:termPOS**: Noun, Plural Noun, Singular Noun, Both Singular and Plural, Past Participle, Verbal Noun/Gerund, Adjectival
- **gvp:termType**: Descriptor, Alternate Descriptor, Used for Term (AAT)

These are captured as Concepts in small subsidiary ConceptSchemes, with their own definitions and examples. You can see these with the query [Ontology Values](#).

Note: in the first release of AAT, the value URLs were under <http://vocab.getty.edu/aat>. But we have now moved them one level up, because many are also used for TGN.

2.8.2 Importance of the Vernacular Flag

Dealing with languages is a relatively new addition to the TGN, so 80% of TGN labels (1475319 of 1846080) don't have a language. Still, you can use the following observations:

- Flag Vernacular, which marks a label as using the local language. Most TGN places have at least one Vernacular label.

```
?label gvp:termFlag <http://vocab.getty.edu/term/flag/Vernacular>
```

- Most TGN English labels are marked as such
- While gvp:prefLabelGVP in AAT is usually in English, in TGN it is usually in the Vernacular.

The GVP site has a feature that allows you to switch between Vernacular Display and English Display, which relies on this. Let's look at a couple of examples from the Greek islands:

- [7011330](#): Dodekánisos is the prefLabelGVP, which is Vernacular in Greek (transliterated). Sporades (nomos) is the English preferred label, even though it's Historic.
- [7236186](#): Ágioi Theódoroi, Vrachonisída is the prefLabelGVP, which is Vernacular in Greek (transliterated). There is no English label, so Vernacular Display and English Display show the same. (Island Saint Teodoro is marked with Greek (transliterated), although it is in fact in English)
- [7001338](#): *Iónioi Nísoi* is the prefLabelGVP, which is Vernacular, even though it is not marked with a specific language. Ionian Islands is the English preferred label

See [Places, with English or GVP Label](#) for a query that selects the English label if present, else the prefLabelGVP.

2.9 Scope Note

All Concepts (but not all non-concept Subjects) in the AAT have definitions (scope notes) in the basic GVP languages: English, Spanish and Dutch (Chinese is upcoming). Currently most (about 98%) of TGN records do NOT have scope notes and those that do are in English.

Notes are represented with type gvp:ScopeNote and have similar (but simpler) information than Terms:

- **dc:identifier**: numeric ID, also used in the URL. See [Identifiers](#)
- **rdf:value**: the note itself (as per [SKOS Primer: 4.2 Advanced Documentation Features](#)), with language tag
- **dct:language** see [Language](#).
- **gvp:displayOrder**, see [Sort Order](#)
- [Historic Information](#) about applicability
- Links to [Source](#)
- Links to [Contributor](#).

(Note: in the previous release we used class xl:Label and property xl:literalForm to hold the note itself).

2.10 Identifiers

We map the database ID's of Subjects, Terms, Scope Notes, Sources and Contributors to dc:identifier. These are random numeric codes that are also used in the [GVP URLs](#) of these entities.

- They are guaranteed to stay permanent, so you can use them in your own data.
- If a Subject is merged to another, we emit it as [Obsolete Subject](#)

2.11 Notations

A notation is a code or number used to uniquely identify a concept within the scope of a given concept scheme. Unlike Terms, notations are not normally recognizable as a sequence of words in any natural language. DDC, UDC, STW and other well-known thesauri use notations. Example codes include "T58.5" or "303.4833".

We use notations for the following:

- Traditional GVP [Facet/Hierarchy Codes](#), e.g.

```
aat:300241490 skos:notation "V.R". # Components (Hierarchy Name)
```

- Note: on the GVP website, lower-level subjects (Guide Terms and Concepts) duplicate the same code as the higher level, so we don't include a notation for those
- Short character codes of [Term Characteristics](#), e.g.

```
<http://vocab.getty.edu/term/kind/ScientificOrTechnical> skos:notation "S".
```

2.12 Source

GVP tracks sources of Terms (labels), Subjects (concepts), and Scope Notes. Sources may be catalogs, encyclopedias, other publications, web sites, databases, etc. AAT and TGN sources are not unified, and so use different prefixes.

We represent sources as `bibo:Document` with the following info:

- **dc:identifier** (see [Identifiers](#))
- **bibo:shortTitle**: brief title
- **dct:title**: full title
- **skos:note**: bibliographic note

For example:

```
aat_source:2000030301 a bibo:Document;
  dc:identifier "2000030301";
  bibo:shortTitle "Chenhall, Revised Nomenclature (1988)";
  dct:title "Chenhall, Robert G. The Revised Nomenclature for Museum Cataloging: ... ".
aat_source:2000051089 a bibo:Document;
  dc:identifier "2000051089";
  bibo:shortTitle "AATA database (2002-)";
  dct:title "Getty Conservation Institute (GCI). database of AATA Online... 2002-. ".
aat_source:2000052946 a bibo:Document;
  dc:identifier "2000052946";
  bibo:shortTitle "Encyclopedia Britannica Online (2002-)";
  dct:title "Encyclopædia Britannica. ... http://www.eb.com/ (1 July 2002).".
```

Sources are applied to Subjects and Scope Notes using `dct:source`.

For terms, GVP may record whether the term is Preferred, NonPreferred or AlternatePreferred for the source, so we use `dct:source` or sub-properties thereof: `gvp:sourcePreferred`, `gvp:sourceNonPreferred`, `gvp:sourceAlternatePreferred`.

The multipart URLs like `aat_source:2000051089-term-1000198841` are explained in the next section:

```
# terms of "rhyta" in English, Greek, Spanish
aat_term:1000198841-en
  gvp:sourceNonPreferred aat_source:2000049728;
  dct:source aat_source:2000051089-term-1000198841.
aat_term:1000198841-el-Latn
  gvp:sourceNonPreferred aat_source:2000049728;
  dct:source aat_source:2000051089-term-1000198841.
aat_term:1000198841-es
  gvp:sourceNonPreferred aat_source:2000049728;
  dct:source aat_source:2000051089-term-1000198841.

# subject 300198841 (rhyta)
aat:300198841
  dct:source aat_source:2000030301-subject-300198841;
  dct:source aat_source:2000052378.

# notes in English and Dutch
aat_scopeNote:34904
  dct:source aat_source:2000046502.
aat_scopeNote:83378
  dct:source aat_source:2000051213.
```

2.12.1 Local Sources

When applied, a source may carry a locator. To attach this info, we need an intermediate node (Local Source) represented as `bibo:DocumentPart`:

- **dct:isPartOf**: points to the global (original) source

- **bibo:locator**: a page number, heading, database ID, or other accession information

The URL of local sources consists of the original source URI, followed by the ID of the thing being described (term, subject or note), e.g.:

```
aat_source:2000051089-term-1000198841 a bibo:DocumentPart;
  dct:isPartOf aat_source:2000051089;
  bibo:locator "128257 checked 26 January 2012".
aat_source:2000030301-subject-300198841 a bibo:DocumentPart;
  dct:isPartOf aat_source:2000030301;
  bibo:locator "horn, drinking".
```

Note: **bibo:locator** is defined as "A description (often numeric) that locates an item within a containing document or collection", which matches our usage.

Its domain is declared as **bibo:Document**. Consequently our **bibo:DocumentParts** are also inferred to be **bibo:Document**. Although unintended, this is ok, since the two classes are not disjoint (we think this is an omission in BIBO: it should allow **bibo:locator** to "locate a **DocumentPart** within its containing **Document**" as well).

2.13 Contributor

GVP tracks contributors of Terms (labels), Subjects (concepts), and Scope Notes. AAT and TGN contributors are not unified, and so use different prefixes.

We represent Contributors as **foaf:Agent** with the following info:

- **dc:identifier** (see [Identifiers](#))
- **foaf:nick**: abbreviation (e.g. CDBP-DIBAM)
- **foaf:name**: full name (e.g. Centro de Documentación de Bienes Patrimoniales (Dirección de Bibliotecas, Archivos y Museos; Santiago, Chile)

For example:

```
aat_contrib:10000000 a foaf:Agent;
  dc:identifier "10000000";
  foaf:nick "VP";
  foaf:name "Getty Vocabulary Program".
aat_contrib:10000131 a foaf:Agent;
  dc:identifier "10000131";
  foaf:nick "CDBP-DIBAM";
  foaf:name "Centro de Documentación de Bienes Patrimoniales ...".
aat_contrib:10000205 a foaf:Agent;
  dc:identifier "10000205";
  foaf:nick "Bureau AAT";
  foaf:name "Bureau AAT, RKD (Netherlands Institute for Art History; ...)".
```

Contributors are attached to Subjects and Notes using **dct:contributor**.

For terms, GVP may record whether the term is Preferred, NonPreferred or AlternatePreferred for the contributor, so we use **dct:contributor** or sub-properties thereof: **gvp:contributorPreferred**, **gvp:contributorNonPreferred**, **gvp:contributorAlternatePreferred** (similar to Sources), e.g.:

```
# term "rhyta"
aat_term:1000198841-en
  gvp:contributorNonPreferred aat_contrib:10000131;
  gvp:contributorPreferred aat_contrib:10000000.
aat_term:1000198841-el-Latn
  gvp:contributorNonPreferred aat_contrib:10000131;
  gvp:contributorPreferred aat_contrib:10000000.
aat_term:1000198841-es
  gvp:contributorNonPreferred aat_contrib:10000131;
  gvp:contributorPreferred aat_contrib:10000000.
```

```
# subject "rhyta"
aat:300198841
  dct:contributor aat_contrib:10000131;
  dct:contributor aat_contrib:10000000.

# notes in English and Dutch
aat_scopeNote:34904
  dct:contributor aat_contrib:10000000.
aat_scopeNote:83378
  dct:contributor aat_contrib:10000205.
```

2.14 Historic Information

GVP includes historic information about the validity of [Terms](#), [Hierarchical Relationships](#), [Associative Relationships](#), and [TGN Place Types](#). We use the following properties:

- **gvp:historicFlag**: with values current, historic, currentAndHistoric
- **schema:startDate**, **schema:endDate**: validity period
 - Literals are emitted in proper XSD date format and carry a type according to its precision, e.g.: "1940"^^xsd:gYear, "1940-06"^^xsd:gMonthYear, "1940-06-15"^^xsd:date
 - Years are spelled with least 4 digits. Years BC are expressed as negative, eg: "0100"^^xsd:gYear, "0010"^^xsd:gYear, "-0100"^^xsd:gYear, "-10000"^^xsd:gYear
 - Previously we used custom sub-properties of dct:valid (defined as "Date (often a range) of validity of a resource"). But some found it confusing to have two values for dct:valid, so we preferred to use schema: properties (which don't infer dct:valid).
- **rdfs:comment**: note on historic applicability.

2.14.1 Applying to Terms

Applying historic information to Terms represented as explicit nodes (skosxl:Label) is straightforward:

```
aat_term:1000002693-en
  a skosxl:Label;
  skosxl:literalForm "lambruscatura"@en ;
  gvp:historicFlag <http://vocab.getty.edu/historic/historic> ;
  schema:startDate "0900"^^xsd:gYear ;
  schema:endDate "1700"^^xsd:gYear ;
  rdfs:comment "Medieval term for wainscoting".
```

2.14.2 Applying to Relations and Place Types

To apply historic information to relations, we use the [RDF Reification](#) vocabulary. It allows to represent a relation instance (i.e. triple) as an rdf:Statement, and use rdf:subject, rdf:object and rdf:predicate to address its components.

We give these statements explicit URLs of the form *_rel:<subj1>-<type>-<subj2> where * is the vocabulary, <subj1> and <subj2> are the related subject IDs, and <type> is:

- "broader" for hierarchical relations
- the specific relation name for associative relations (see [Relationship Representation](#))
- "placeType" for TGN place types. These also carry gvp:displayOrder (see [Sort Order](#))

We give some examples: the first group from AAT, the second group from TGN:

```

aat_rel:300107346-broader-300020541 a rdf:Statement;
  rdf:subject      aat:300107346;          # Early Imperial
  rdf:predicate    gvp:broaderPreferred;
  rdf:object       aat:300020541;          # Imperial (Roman)
  rdfs:comment     "ca. 27 BCE-68 CE";
  schema:startDate "-0017"^^xsd:gYear;
  schema:endDate   "0068"^^xsd:gYear .

aat_rel:300020271-aat2812_followed-300020269 a rdf:Statement;
  rdf:subject      aat:300020271;          # Second Dynasty (Egyptian)
  rdf:predicate    gvp:aat2812_followed;
  rdf:object       aat:300020269;          # First Dynasty (Egyptian)
  rdfs:comment     "Second Dynasty began ca. 2775 BCE";
  schema:startDate "-2785"^^xsd:gYear;
  schema:endDate   "-2765"^^xsd:gYear .

tgn:7011179-placeType-300008347 a rdf:Statement;
  rdf:subject      tgn:7011179;           # Siena
  rdf:predicate    gvp:placeTypePreferred;
  rdf:object       aat:300008347;          # inhabited place
  rdfs:comment     "settled by Etruscans (flourished 6th century BCE)";
  schema:startDate "-0800"^^xsd:gYear;
  gvp:displayOrder "1"^^xsd:positiveInteger.

aat_rel:7011179-tgn3301_ally_of-7006072 a rdf:Statement;
  rdf:subject      aat:7011179;           # Siena
  rdf:predicate    gvp:tgn3301_ally_of;
  rdf:object       aat:7006072;           # Arezzo
  rdfs:comment     "Ghibelline allies during the 13th and 14th centuries";
  schema:startDate "1250"^^xsd:gYear;
  schema:endDate   "1400"^^xsd:gYear.

```

- In the case of hierarchical relations, historic information is asserted against gvp:broaderPreferred or gvp:broaderNonPreferred and is **not copied** to inferred relations.
- In the case of place types, historic information is asserted against gvp:placeTypePreferred or gvp:placeTypeNonPreferred and is **not copied** to inferred relations.
- In the case of associative relations, the same historic information is asserted against both the forward and inverse relations (e.g. aat2811_preceded and aat2812_followed) with the roles of rdf:subject and rdf:object swapped. For a symmetric relation (e.g. tgn3301_ally_of), the same info is asserted twice, with the roles of rdf:subject and rdf:object swapped. This allows you to fetch all relations of a subject by searching for rdf:subject, and not worrying about rdf:object (see query [Historic Information on Relations](#))

2.15 Revision History

GVP keeps extensive info about actions on Subjects and Sources: AAT has over 600k records as of Feb 2014. Although extensive, there is no guarantee the info is comprehensive, especially for very old actions, and for Publish events (issued). The various kinds of actions are listed below, together with approximate numbers for AAT (see [Count Revision Actions](#)). Actions on sub-entities of Subjects (terms, notes, associative relations) are emitted for the Subject.

what	dc:type	rdf:type	count	dc:description
Subject	created	prov:Create	45798	
Subject	updated	prov:Modify	105082	
Subject	term added	prov:Modify	293212	<term> (<term.id>)
Subject	term deleted	prov:Modify	15501	<term> (<term.id>) OR was <term>
Subject	note created	prov:Modify	10626	Language: <lang.name> (<lang.id>)
Subject	note updated	prov:Modify	20719	Language: <lang.name> (<lang.id>)
Subject	moved	prov:Modify	21588	Old Parent: <parent.term> (<parent.id>)
Subject	parent added	prov:Modify	3410	<parent.term> (<parent.id>)
Subject	relation added	prov:Modify	33321	<this.term> (<this.id>) 'relation.name' <related.term> (<related.id>)
Subject	relation deleted	prov:Modify	12512	<this.term> (<this.id>) 'relation.name' <related.term> (<related.id>)
Subject	merged	prov:Modify	soon	Dominant: <this.term> (<this.id>), Recessive: <rec.term> (<rec.id>)
Subject	issued	prov:Publish	2432	To be published to Website and LOD within 2 weeks
Source	created	prov:Create	35974	
Source	updated	prov:Modify	16306	
Source	merged	prov:Modify	423	Dominant: <this.name>, Recessive: <rec.name>

There are over 10M "term updated" actions that are not very interesting and are therefore elided.

2.15.1 Revision History Representation

We map revision actions to [PROV-O](#), using the [PROV-DC](#) mapping for inspiration. We use classes and properties from [prov-o.ttl](#) and [prov-dc-refinements.ttl](#). But because PROV has a high level of complexity (see the examples in section [PROV](#)), we use a rather simplified mapping.

Revision records use the *_rev: and *_source_rev: prefixes and have the following data:

- **rdf:type**: prov:Activity, and a specific type as listed in the table above: prov:Create, prov:Modify, or prov:Publish (these come from the PROV-DC mapping)
- **dc:type**: a literal as listed in the table above
- **prov:startedAtTime**: timestamp of the action (xsd:dateTime)
- **dc:description**: additional narrative about the action, such as which Recessive subject was merged, or what is the language of the note that was added.

Links between the entity and its actions:

- **skos:changeNote**: from entity to action. The [SKOS Advanced Documentation](#) pattern shows this for skos:Concept. For uniformity, we use the same property for any gvp:Subject, and also for Sources (foaf:Agents). The property doesn't define a domain, so that's permissible.
- **prov:wasGeneratedBy**: from entity to the prov:Create action
- **prov:used**: from prov:Modify or prov:Publish to the entity

Timestamps on the entity: we emit the action timestamps also as timestamps on the entity, using DCT properties:

- prov:Create → dct:created
- prov:Modify → dct:modified: there is one for each Modify action. If you prefer to have only the latest timestamp, let us know
- prov:Publish → dct:issued

2.15.2 Revision History for Subject

Let's say subject 300018699 was created by action 12345, modified by action 12346 (a term was added), and issued (published) by action 12347. We map it thus:

```

aat:300018699
  skos:changeNote aat_rev:12345, aat_rev:12346, aat_rev:12347;
  prov:wasGeneratedBy aat_rev:12345;
  dct:created "2014-01-02T01:02:03"^^xsd:dateTime;
  dct:modified "2014-01-03T01:02:03"^^xsd:dateTime;
  dct:issued "2014-01-04T01:02:03"^^xsd:dateTime.
aat_rev:12345 a prov:Activity, prov:Create;
  dc:type "created";
  prov:startedAtTime "2014-01-02T01:02:03"^^xsd:dateTime.
aat_rev:12346 a prov:Activity, prov:Modify;
  prov:used aat:300018699;
  dc:type "term added";
  dc:description "leggings, puttee (1000248060)";
  prov:startedAtTime "2014-01-03T01:02:03"^^xsd:dateTime.
aat_rev:12347 a prov:Activity, prov:Publish;
  prov:used aat:300018699;
  dc:type "issued";
  prov:startedAtTime "2014-01-04T01:02:03"^^xsd:dateTime.

```

2.15.3 Revision History for Source

Let's say source 2000053040 was created by action 12345 and modified by action 12346 (another source was merged); there are no Publishing actions for sources. (Note: these are different from the Subject actions described above, even if they have the same ID)

```

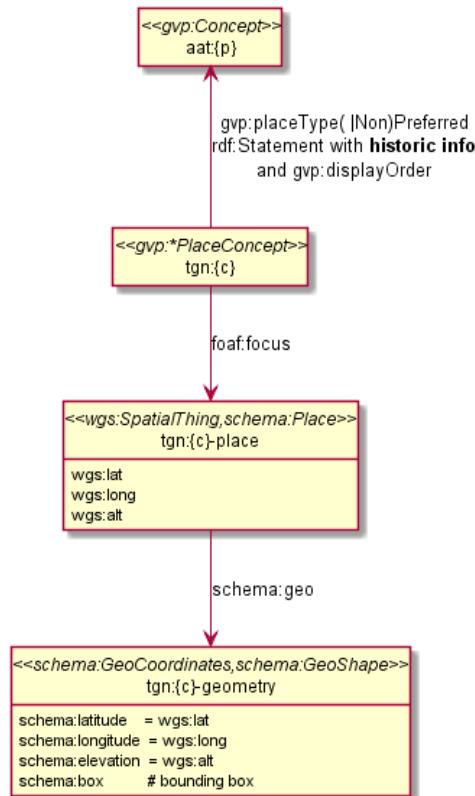
aat_source:2000053040
  skos:changeNote aat_source_rev:12345, aat_source_rev:12346;
  prov:wasGeneratedBy aat_source_rev:12345;
  dct:created "2014-01-02T01:02:03"^^xsd:dateTime;
  dct:modified "2014-01-03T01:02:03"^^xsd:dateTime.
aat_source_rev:12345 a prov:Activity, prov:Create;
  dc:type "created";
  prov:startedAtTime "2014-01-02T01:02:03"^^xsd:dateTime.
aat_source_rev:12346 a prov:Activity, prov:Modify;
  prov:used aat_source:2000053040;
  dc:type "merged";
  dc:description "Recessive: Magness, Archaeology of Qumran ... (2003) (2000076344)";
  prov:startedAtTime "2014-01-03T01:02:03"^^xsd:dateTime.

```

3 TGN Specifics

All of the information in the previous section applies equally to AAT and TGN. TGN has additional information, which is shown below and described in the following sub-sections.

3.1 TGN Overview



The additional information is attached to the gvp:*PlaceConcept **tgn:{c}**:

- The TGN concept has [TGN Place Types](#) that are AAT concepts. This relation carries gvp:displayOrder (see [Sort Order](#)) and [Historic Information](#)
- Following a [Concept vs Place Duality](#) principle, the PlaceConcept points to a separate node **tgn:{c}-place** using foaf:focus. It carries the WGS coordinate information, but the Schema coordinate information is in yet another node **tgn:{c}-geometry**.
- [Coordinate Information](#) consists of latitude, longitude, latitude, and (only for regions) bounding box

3.2 TGN Place Types

TGN includes rich information about Place Types:

- There are almost 1800 place types, ranging from "shantytown" to "undersea mountain chain".
- When applied to a particular place:
 - Place Types are Ordered (gvp:displayOrder)
 - Each place has one Preferred type and may have as many as 10 Non-Preferred types
 - They carry [Historic Info](#): Historic flag, Start Date, End Date, Comment (display date), see [Applying to Relations and Place Types](#)
- The average is 2.12 types per place (Jun 2014: 2671142 place type instances for 1259162 places).

E.g. the place types of **Machupicchu, Peru** are:

- deserted settlement (preferred, current). Start: 1430, End: 1550. Comment: building started ca. 1440; was inhabited until the Spanish conquest of Peru in 1532
- archaeological site (current). Start: 1911. Comment: rediscovered in 1911
- ruins (current)
- inhabited place (historic)
- Inca center (historic). Start: 1440, End: 1550

Types are attached to a place with `gvp:placeTypePreferred` or `gvp:placeTypeNonPreferred`. These are sub-properties of `gvp:placeType`, going from TGN (Place Concept) to AAT (Place Type).

- Place types are maintained in AAT and form BTG hierarchies.
- For example see the [aat:300008347](#) "inhabited places" hierarchy.
- See [Places by Direct and Hierarchical Type](#) for a comparison of querying by direct vs hierarchical type.

3.2.1 Why not broaderInstantial?

In a previous version (2.0) we mapped place types to `gvp:broaderInstantial`:

- BTG is appropriate to represent the type of something (`iso:broderInstantial` is defined as "instance - class relationship")
- It infers hierarchical types `gvp:broaderInstantialExtended`, standard properties `iso:broderInstantial`, `skos:broader` and `skos:broaderTransitive`
- It goes across thesauri (`skos:ConceptSchemes`). This is possible, because SKOS allows `skos:broaderMatch` (and by inference `skos:broader`) to go across thesauri. Conceptually it "grafts" TGN places under the respective AAT concepts.

But after experimentation with inferred data, we decided not to use `broaderInstantial` for place types, for the following reasons:

Improper `skos:broaderTransitive` inference

`skos:broaderTransitive` doesn't respect the BTG/BTP/BTI distinction, so it freely mixes BTG/BTP/BTI. This causes each place to "inherit" the types of all its parent places, e.g.:

n	statement	justification
1	Amsterdam Rijn Kanaal (canal) - <code>gvp:broaderPartitive</code> - Netherlands (nation)	explicit
2	Amsterdam Rijn Kanaal (canal) - <code>iso:broaderPartitive</code> - Netherlands (nation)	1, between <code>skos:Concepts</code>
3	Amsterdam Rijn Kanaal (canal) - <code>skos:broader</code> - Netherlands (nation)	2, <code>rdfs:subPropertyOf</code>
4	Netherlands (nation) - <code>gvp:broaderInstantial</code> - nations (AAT)	explicit
5	Netherlands (nation) - <code>gvp:broaderInstantial</code> - nations (AAT)	4, between <code>skos:Concepts</code>
6	Netherlands (nation) - <code>skos:broader</code> - nations (AAT)	5, <code>rdfs:subPropertyOf</code>
7	Amsterdam Rijn Kanaal (canal) - <code>skos:broaderTransitive</code> - nations (AAT)	3, 6, <code>owl:TransitiveProperty</code>

In its turn, Netherlands inherits all types of Europe and World; and Amsterdam Rijn Kanaal also inherits them.

This made `skos:broaderTransitive` that are very non-intuitive, and also generated a large number of useless statements.

The higher levels of the place type hierarchy are less useful

Consider the BTG hierarchies of "continents":

Objects Facet

.. Built Environment (Hierarchy Name)

.... Settlements and Landscapes

..... landscapes (environments)

..... **natural landscapes**

..... **landforms (terrestrial)**

..... **landmasses**

..... **continents**

Associated Concepts Facet

.. Associated Concepts
 scientific concepts
 physical sciences concepts
 earth sciences concepts
 earth features
 physical features
 **hypsographic features**
 **terrestrial features (natural)**
 **landforms (terrestrial)**
 **continents**

Arguably, the bold levels are useful while the higher levels are not really useful.

For example, the second bold hierarchy allows you to distinguish between features: hypsographic vs hydrographic vs undersea vs vegetation (e.g. forests, plantations). Compare with GeoNames feature classes:

8,592,982	Class	Description
303,814	A	Administrative Boundary Features (country, state, region,...)
1,742,469	H	Hydrographic Features (stream, lake, ,,,)
309,264	L	Area Features (parks,area, ,,,)
3,256,345	P	Populated Place Features (city, village,,,))
36,658	R	Road / Railroad Features (road, railroad)
1,812,606	S	Spot Features (spot, building, farm)
1,081,794	T	Hypsographic Features (mountain,hill,rock,,,)
13,955	U	Undersea Features (undersea)
30,784	V	Vegetation Features (forest,heath,,,)

"Physical features" is not useful, because its other children (sky and clouds) are not subject of the TGN

Few people would think of Europe as a physical science concept or a scientific concept

The non-preferred type hierarchies are less useful

Consider the preferred and non-preferred hierarchy of "inhabited places" (which includes cities, villages, etc):

Objects Facet

.. Built Environment (Hierarchy Name)

.... Settlements and Landscapes

..... inhabited places

Agents Facet

.. Organizations (Hierarchy Name)

.... organizations (groups)

..... administrative bodies

..... political administrative bodies

..... <political administrative bodies by general designation>

..... inhabited places

Arguably, the non-preferred hierarchy is less useful, because fewer people would think of a city as an organization, group or agent (this represents the corporate or governance nature of a city).

The AAT place type hierarchy is too young

The AAT place type hierarchy was created only a few months ago, and there is still no feedback which structures and distinctions are useful. Compare GeoNames features, which is a fixed 2-level mono-hierarchy. For example, below are some of GeoNames **Hydrographic Features**. They don't allow to distinguish characteristics such as:

State of matter (ice vs water)

Saltwater vs freshwater

Flowing vs stationary (riverine vs lakustrine vs wetlands). AAT has this distinction, see [<bodies of freshwater by biome>](#)

Natural vs man-made

Permanent vs intermittent

1,742,469	Feature	Name	Description
775,558	H.STM	stream	a body of running water moving to a lower level in a channel on land
251,139	H.LK	lake	a large inland body of standing water
125,251	H.STMI	intermittent stream	
82,555	H.RSV	reservoir(s)	an artificial pond or lake
80,225	H.WLL	well	a cylindrical hole, pit, or tunnel drilled or dug down to a depth from which water, oil, or gas can be pumped or brought to the surface
61,222	H.SPNG	spring(s)	a place where ground water flows naturally out of the ground
59,406	H.WAD	wadi	a valley or ravine, bounded by relatively steep banks, which in the rainy season becomes a watercourse; found primarily in North Africa and the Middle East
48,020	H.BAY	bay	a coastal indentation between two capes or headlands, larger than a cove but smaller than a gulf
29,703	H.CNL	canal	an artificial watercourse
21,740	H.PND	pond	a small standing waterbody
20,313	H.RSVT	water tank	a contained pool or tank of water at, below, or above ground level
15,736	H.COVE	cove(s)	a small coastal indentation, smaller than a bay
11,247	H.SWMP	swamp	a wetland dominated by tree vegetation
9,920	H.CHN	channel	the deepest part of a stream, bay, lagoon, or strait, through which the main current flows
9,099	H.RF	reef(s)	a surface-navigation hazard composed of consolidated material
8,338	H.SHOL	shoal(s)	a surface-navigation hazard composed of unconsolidated material
8,105	H.RVN	ravine(s)	a small, narrow, deep, steep-sided stream channel, smaller than a gorge
7,678	H.WTRH	waterhole(s)	a natural hole, hollow, or small depression that contains water, used by man and animals, especially in arid areas
7,195	H.INLT	inlet	a narrow waterway extending into the land, or connecting a bay or lagoon with a larger body of water
6,560	H.GLCR	glacier(s)	a mass of ice, usually at high latitudes or high elevations, with sufficient thickness to flow away from the source area in lobes, tongues, or masses

A poly-hierarchy like AAT has greater potential. However, more usage experience is required before that potential can be realized. We would like to receive feedback about what distinctions and structures of the place hierarchy you find useful.

3.3 Concept vs Place Duality

It is a somewhat established practice that a Concept should be treated separately from the Thing that it denotes (Person, Place, Monument, Historic event, etc).

- Concepts are the "business objects" of thesaurus management systems, and the daily business of editorial teams like GVP
- Things on the other hand exist (or have existed) independently in the real world.

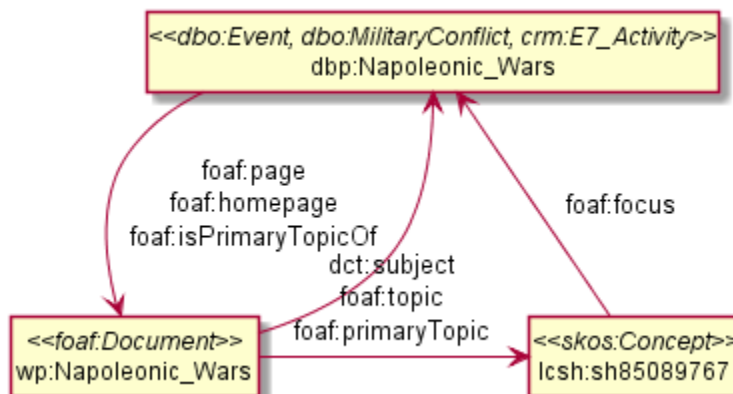
- Events can happen at Places (not Concepts); Events can be enacted by Agents (not Concepts), and cultural Objects can be created by Persons/Agents (not Concepts).
- Please consider "creation dates". The creation date of a cultural object (or the birth date of a person) is very different from dct:created of a Concept (which is the date when it was registered in a particular thesaurus management system)

A good explanation is provided in the blog post [Things & their conceptualisations: SKOS, foaf: focus & modelling choices](#) by Pete Johnston, Cambridge University Library, Sep 2011 (search for "Concept Schemes, SKOS and Document Metadata"). This post makes distinction between three nodes:

- A real-world (non-web) thing: http://dbpedia.org/resource/Napoleonic_Wars: an event taking place between 1800 and 1815, something with a duration in time, which occurred in physical locations, and in which human beings participated
- A web page (foaf:Document): http://en.wikipedia.org/wiki/Napoleonic_Wars: a Wikipedia page, a document, created and modified by Wikipedia contributors between 2002 and the present
- A concept (skos:Concept): <http://id.loc.gov/authorities/subjects/sh85089767>: a "conceptualisation of" the Napoleonic Wars, a social and technological artifact "designed to help interconnect", an "abstraction" created by the authors of LCSH for the purposes of classifying works

It is an established practice to use **foaf: focus** for the denotation link.

The 3 URLs are illustrated below, with some typically used types and relations:



You could also read the post [What Do URIs Mean Anyway?](#) by [Jenni Tennison](#), UK Open Data Institute, Jul 2011.

3.3.1 Cons of the Dual Approach

The cons of this approach include:

- More complexity, potential confusion which URL to use.
In your Cultural Heritage data you should use the Place or Agent URL (tgn:{m}-place, respectively ulan:{m}-agent).
- May need to duplicate information about each resource between the dual URLs, for example:
 - for TGN: skos:broadier and "place is part of" (eg crm:P88i_forms_part_of)
 - for ULAN: skos:pref/altLabel and foaf:name
- Harder to co-reference places, people etc across cultural heritage collections, which is an important use case for cross-collection search

3.3.2 Co-reference and Co-denotation

The usual way to co-reference instances of places, people etc across authority lists or collections is with **owl:sameAs**. E.g.

- http://dbpedia.org/resource/Leonardo_da_Vinci says it is owl:sameAs [freebase:Leonardo da Vinci](http://www.wikidata.org/entity/Q762), <http://www.wikidata.org/entity/Q762>, [http://yago-knowledge.org/resource/Leonardo da Vinci](http://yago-knowledge.org/resource/Leonardo_da_Vinci), etc
- <http://viaf.org/viaf/24604287> says it is owl:sameAs <http://www.idref.fr/085975915/id>, <http://data.bnf.fr/ark:/12148/cb11912491s#foaf:Person>, <http://d-nb.info/gnd/118640445>, <http://libris.kb.se/resource/auth/207435>, [http://dbpedia.org/resource/Leonardo da Vinci](http://dbpedia.org/resource/Leonardo_da_Vinci)

By standard OWL semantics, owl:sameAs causes all statements of two URLs (if loaded into the same repository) to be "smushed together". This is required to implement cross-collection search: when the data of several collections use different URLs for Leonardo, we need to make these URLs equivalent somehow.

However, one should use **skos:exactMatch** to say that two Concepts denote the same thing; if one used owl:sameAs, that would lose the individual perspectives of the two editorial teams that created the two Concepts (e.g. will copy dct:created between the two concepts, and cross-graft the two hierarchies together). There is no special semantics associated with skos:exactMatch (it is even weaker than skos:broadMatch), so one doesn't get co-referencing by skos:exactMatch alone.

The in-depth discussion [skos:exactMatch vs owl:sameAs](#) discusses the cons of the dual approach and proposes a "co-denotation axiom" at the end. Expanding on this, we could propose the following co-denotation axioms. ?cN are skos:Concept, ?tN are Things, we use [N3 Rules](#) notation, and following OWL semantics, **?t owl:sameAs ?t** always holds (a thing is equivalent to itself):

- Concepts denoting the same (or equivalent) thing are equivalent:

```
{?c1 foaf:focus ?t1. ?c2 foaf:focus ?t2. ?t1 owl:sameAs ?t2} => {?c1 skos:exactMatch ?c2}
```

- Equivalent concepts denote the same (or equivalent) thing:

```
{?c1 skos:exactMatch ?c2. ?c1 foaf:focus ?t} => {?c2 foaf:focus ?t}
{?c1 skos:exactMatch ?c2. ?c1 foaf:focus ?t1. ?c2 foaf:focus ?t2} => {?t1 owl:sameAs ?t2}
```

The last will buy us co-reference, but none of these axioms is adopted by the semantic web community (yet).

In the rest of this section we give examples and counter-examples from well-known libraries and authority lists.

3.3.3 VIAF: pro

The Virtual International Authority File (VIAF) aggregates authority info from about 20 national libraries. It uses the dual practice.

- Page about Leonardo: <http://viaf.org/viaf/24604287>
- RDF representation (Available from "Record views" at the bottom): <http://viaf.org/viaf/24604287/rdf.xml>

The entries from contributing institutions (including ULAN) are represented as skos:Concepts that have a foaf:focus link to the main entity (foaf:Person and rdaGr2:Person):

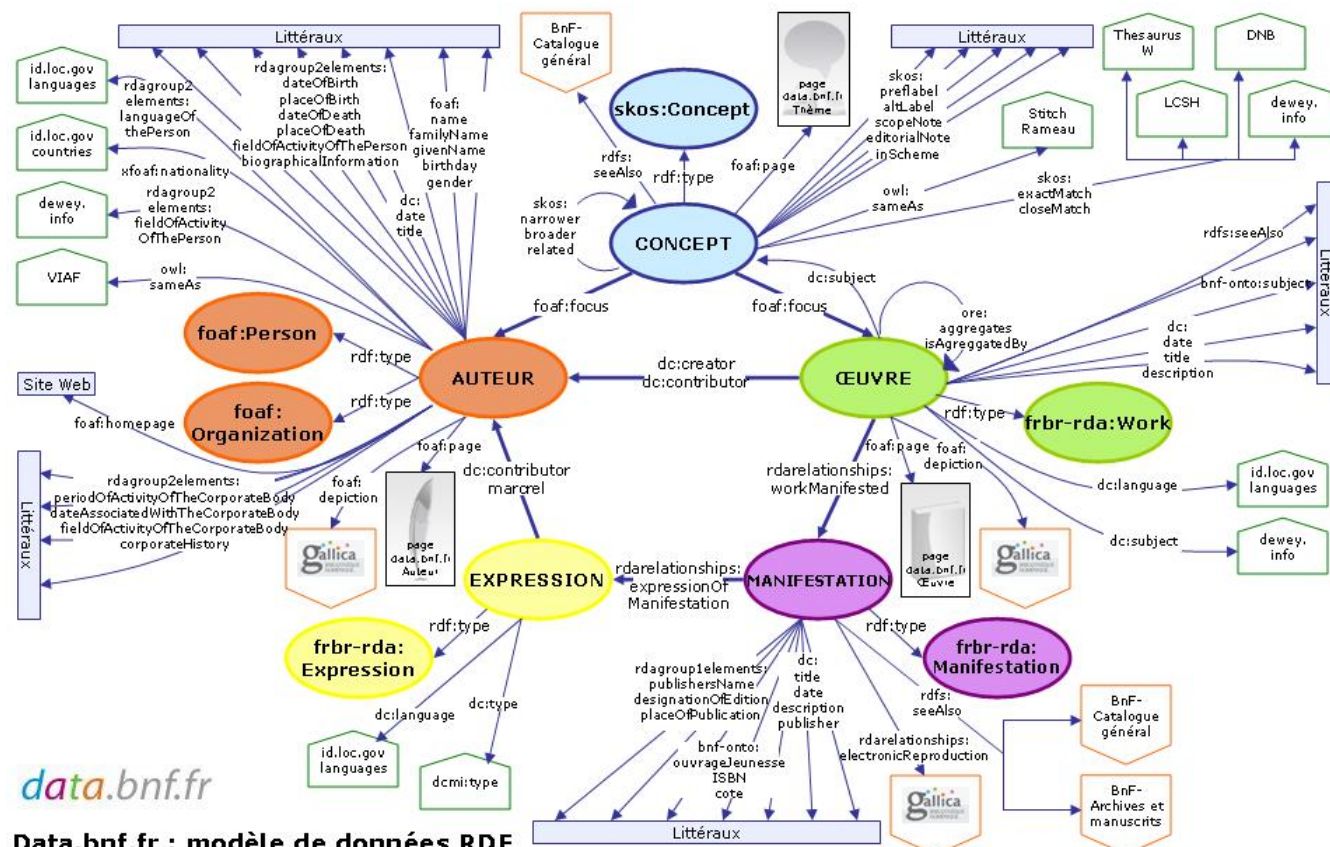
```
<http://viaf.org/viaf/sourceID/JPG%7C500010879#skos:Concept> a skos:Concept;
  foaf:focus <http://viaf.org/viaf/24604287>.
```

The foaf:Person has owl:sameAs links equating to URIs in other known sources.

The dual URLs are rendered as <person/> and <person/#skos:Concept>

3.3.4 FR BnF: pro

The National Library of France (BnF) uses the dual practice. In the middle of the BnF data model is `skos:Concept` that has `foaf:focus` to `AUTEUR` (`foaf:Agent`) and `OEUVRE` (`frbr-rda:Work`):



Data.bnf.fr : modèle de données RDF

Alignements externes	rdf: http://www.w3.org/1999/02-22/rdf-syntax-ns# xsf: http://www.w3.org/2000/D1/rdf-schema# owl: http://www.w3.org/2002/07/owl# skos: http://www.w3.org/2004/02/skos/core# dc: http://purl.org/dc/elements/1.1/	foaf: http://xmlns.com/foaf/0.1/ xfoaf: rdarelements: http://rdvocab.info/RDARelationshipsWEMI/ rdagroupElements: http://rdvocab.info/Element# rdagroup2Elements: http://rdvocab.info/Element#G2/	ore: http://www.openarchives.org/ore/terms/ foir-ids: http://purl.org/vocab/foir-core# marcat: http://d.ac.gov/vocabulary/relation/ bnf-anla: http://dsis.bnf.fr/anlaology/bnf-anla/
Sources de données BnF	Préfixes		


```
<http://libris.kb.se/resource/auth/207435#concept> a skos:Concept;
  foaf:focus <http://libris.kb.se/resource/auth/207435> .
  skos:exactMatch <http://viaf.org/viaf/24604287/#skos:Concept> .
<http://libris.kb.se/resource/auth/207435> a foaf:Person;
  owl:sameAs <http://dbpedia.org/resource/Leonardo_da_Vinci>,
    <http://viaf.org/viaf/24604287>,
    <http://id.loc.gov/authorities/names/n79034525>.
```

There is also information about creator and subject of various creative works:

```
<http://libris.kb.se/resource/bib/22919> dc:creator <http://libris.kb.se/resource/auth/207435> .
<http://libris.kb.se/resource/bib/23473> dc:subject <http://libris.kb.se/resource/auth/207435> .
<http://libris.kb.se/resource/bib/23473> dc:subject
<http://libris.kb.se/resource/auth/207435#concept> .
```

- Both Person and Concept are used for dc:subject (I agree that a Person can indeed be the subject of a creative work).
- Not only owl:sameAs to the VIAF person is made, but also skos:exactMatch to the VIAF concept
- Unfortunately, the owl:sameAs to <http://id.loc.gov/authorities/names/n79034525> breaks the scheme. LOC doesn't use the dual practice, so this makes a skos:Person owl:sameAs skos:Concept, which defeats the purpose

3.3.7 US LoC: cons

The Library of Congress Name Authority File (LCNAF) does **not** use this practice.

Consider the RDF for Leonardo: <http://id.loc.gov/authorities/names/n79034525.skos.rdf>: the URL is declared a skos:Concept, and there is no class for Person.

3.3.8 DE DNB: cons

The German National Library (DNB) does not use this practice. The reason is that the [GND ontology](#) does not use (is not aligned to) SKOS. Appropriate classes are aligned to FOAF though, e.g.

```
gnd:DifferentiatedPerson rdfs:subClassOf dnb:DistinguishedPerson.
dnb:DistinguishedPerson owl:equivalentClass foaf:Person.
```

Consider the RDF for Leonardo: <http://d-nb.info/118640445/about/rdf>. There is a single URL of type gnd:DifferentiatedPerson.

3.4 Coordinate Information

We represent TGN coordinates using two [Geographic Ontologies](#) (WGS and Schema.org). For example [tgn:3000034](#) Great Lakes Region:

```
tgn:3000034 a gvp:AdminPlaceConcept; # Great Lakes Region
  gvp:broaderPreferred tgn:1000001; gvp:broaderPartitive tgn:1000001; # North and Central America
  gvp:tgn3000_related_to tgn:7029370; # Great Lakes (lakes)
  foaf:focus tgn:3000034-place.
tgn:3000034-place a wgs:SpatialThing, schema:Place;
  wgs:lat "45.0000"; wgs:long "-85.0000"; wgs:alt "183.1840";
  schema:geo tgn:3000034-geometry.
tgn:3000034-geometry a schema:GeoCoordinates, schema:GeoShape;
  schema:latitude 45.0000; schema:longitude -85.0000; schema:elevation 183.1840;
  schema:box "-92.0160,43.1560 -92.0160,48.8120 -82.4910,48.8120 -82.4910,43.1560 -92.0160,43.1560".
```

- Like many other geographic ontologies, schema.org makes a difference between a Place and its geometry (GeoCoordinates and GeoShape). So we put the schema:latitude, schema:longitude, schema:elevation properties there
 - A few TGN places also have bounding box info. We put it in schema:box as a rectangular polygon of 5 space-separated points (where the first and last coincide), each represented as "long,lat".
- WGS is simpler: it has a class SpatialThing that carries wgs:lat, wgs:long and wgs:alt directly.

4 Additional Features

4.1 Inference

An important decision is what sort of inference level to use in the GVP SPARQL endpoint. Considerations:

- Advantage: the more powerful the inferencing, the fewer facts need to be stated explicitly and the easier it will be to develop and maintain the conversion.
- Disadvantage: users that download the files and don't have the same or higher level of inference will be at a disadvantage

We use an approach that uses the advantage while avoiding the disadvantage:

- The conversion process produces only basic facts: [Explicit Exports](#)
- Then we use OWLIM's powerful inference features (RDFS, OWL RL, OWL QL, custom rules) and INSERT queries to infer all required consequences and materialize them in the repository.
- Then we extract from OWLIM export files with all inferred facts: [Per-Entity Exports](#) and [Total Exports](#)

In this way you can use the [Total Exports](#) without need for additional inference, or use the [Explicit Exports](#) and provide the inferences described below.


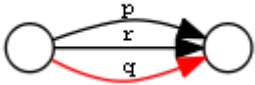
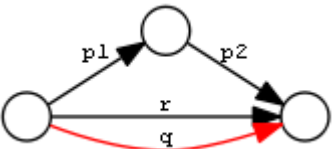
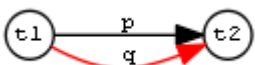
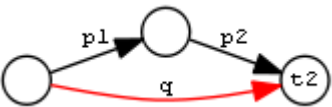
We also run INSERT queries to generate [Dynamic Descriptive Properties](#), but you don't need to do that: you can get the VOID file directly.

4.1.1 Extended Property Constructs

While OWL2 has very powerful class constructs, its property constructs are quite weak. In particular, OWL2 does not support conjunctive properties. For the inferences below, we found it useful to define several extensions. See [Extending OWL2 Property Constructs with OWLIM Rules](#) for comparison to OWL2, discussion and more constructs. Notation:

- **pN** are premises, **r** is a restriction (just another premise), **tN** are types, **q** is the conclusion, **t** is the axiom holding them together
- **p & r** is property conjunction (restriction): both properties connect the same nodes
- **[t1] p [t2]** is type restriction: the source node has type t1 and the target node has type t2 (shown inside the node)

We implement the constructs using [OWLIM Rules](#), and provide the implementations for reference below. We would be interested to hear about efficient implementations of these constructs using [SPIN Rules](#) over large datasets (not just in-memory databases).

Construct	Illustration	OWLIM implementation
PropChain $q \leq p1 / p2$		<pre> Id: ptop_PropChain t <ptop:premise1> p1 t <ptop:premise2> p2 t <ptop:conclusion> q t <rdf:type> <ptop:PropChain> x p1 y y p2 z ----- x q z </pre>
PropRestr $q \leq p \ \& \ r$		<pre> Id: ptop_PropRestr t <ptop:premise> p t <ptop:restriction> r t <ptop:conclusion> q t <rdf:type> <ptop:PropRestr> x p y x r y ----- x q y </pre>
PropChainRestr $q \leq (p1 / p2) \ \& \ r$		<pre> Id: ptop_PropChainRestr t <ptop:premise1> p1 t <ptop:premise2> p2 t <ptop:restriction> r t <ptop:conclusion> q t <rdf:type> <ptop:PropChainRestr> x p1 y y p2 z x r z ----- x q z </pre>
TypeRestr $q \leq [t1] \ p \ [t2]$		<pre> Id: ptop_TypeRestr t <ptop:premise> p t <ptop:type1> t1 t <ptop:type2> t2 t <ptop:conclusion> q t <rdf:type> <ptop:TypeRestr> x p y x <rdf:type> t1 [Cut] y <rdf:type> t2 [Cut] ----- x q y </pre>
PropChainType2 $q \leq p1 / p2[t2]$		<pre> Id: ptop_PropChainType2 t <ptop:premise1> p1 t <ptop:premise2> p2 t <ptop:type2> t2 t <ptop:conclusion> q t <rdf:type> <ptop:PropChainType2> x p1 y y p2 z z <rdf:type> t2 ----- x q z </pre>

For chains of length 2 (which are most chains seen in practice), PropChain allows a more efficient implementation than owl:propertyChainAxiom, since one doesn't have to unroll the rdf:List of owl:propertyChainAxiom, making intermediate nodes and statements.

It is also more efficient to implement transitivity with PropChain, instead of owl:TransitiveProperty:

- First, owl:TransitiveProperty is a special case of property chain: a self-chain:

```
?q a owl:TransitiveProperty <=> ?q owl:propertyChainAxiom (?q ?q)
```

- By itself, the above chain won't infer anything. Thus transitive properties are usually made on the basis of a step (basis) property p:

```
?p rdfs:subPropertyOf ?q.
```

- It is more efficient to use the step property in the chain, instead of making a self-chain; because the reasoner will try to grow the chain only at the end, instead of trying to combine any split of the chain:

```
?q owl:propertyChainAxiom (?q ?p).
```

- We include a rule that implements owl:TransitiveProperty as a self-chain:

```
Id: ptop_TransPropAsChain
q <rdf:type> <owl:TransitiveProperty>
-----
t <rdf:type> <ptop:PropChain>
t <ptop:premise1> q
t <ptop:premise2> q
t <ptop:conclusion> q
```

- But for high-volume transitive properties, we can redeclare them with the appropriate non-self-chain to gain a speed advantage.

The [Cut] in TypeRestr means that the inferencer won't try to apply the rule when one of the rdf:type statements is asserted, it will wait until all other premises (in particular "x p y") are asserted.

The constructs are used in specific places below.

4.1.2 Reduced SKOS Inference

The SKOS property hierarchy is shown below, together with an indication of property characteristics:

S=symmetric, I=inverse, T=transitive, D=disjoint:

skos:semanticRelation	rdfs:label
skos:related (S)	skos:prefLabel
skos:relatedMatch (S)	skos:altLabel
skos:broaderTransitive (T)	skos:hiddenLabel
skos:broader (I)	
skos:broadMatch (I)	skos:note
skos:narrowerTransitive (IT)	skos:changeNote
skos:narrower (I)	skos:definition
skos:narrowMatch (I)	skos:editorialNote
skos:mappingRelation	skos:example
skos:closeMatch (S)	skos:historyNote
skos:exactMatch (ST)	skos:scopeNote
skos:relatedMatch (S)	
skos:broadMatch (I)	
skos:narrowMatch (I)	

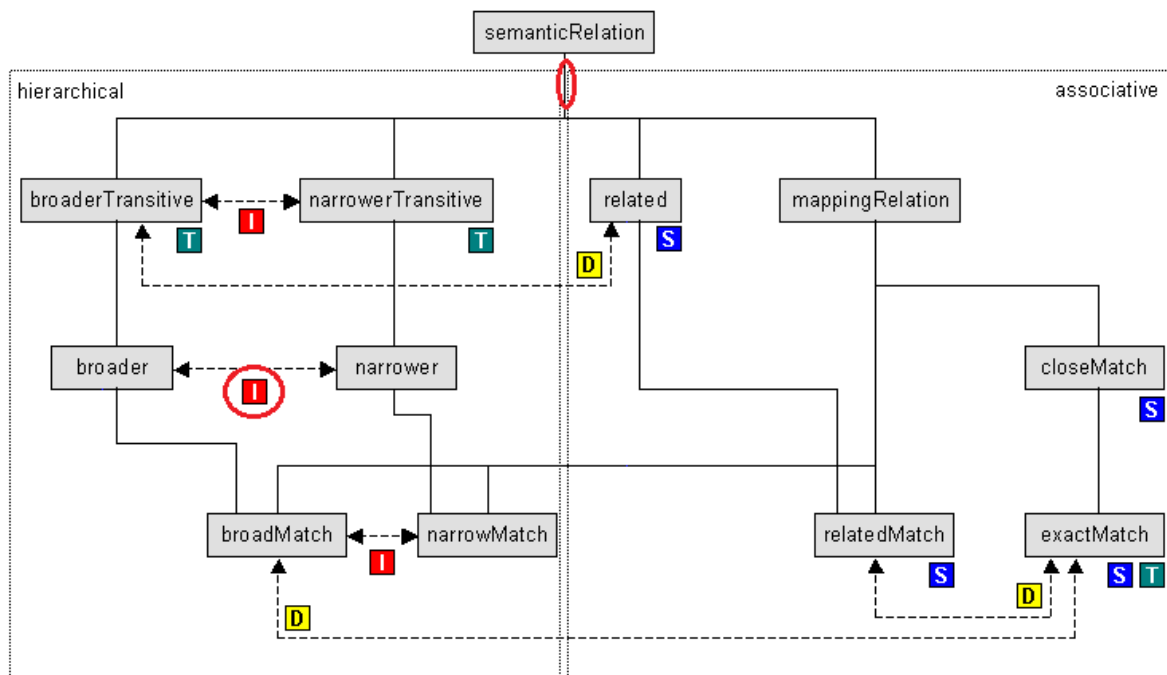
The following constructs are used in the SKOS ontology:

- rdfs:subPropertyOf**, e.g. to infer skos:broader → skos:broaderTransitive
- owl:TransitiveProperty**: to make the transitive closure, e.g. of skos:broaderTransitive
- owl:SymmetricProperty**: to infer "A skos:related B" → "B skos:related A"
- owl:inverseOf**: to infer e.g. skos:broader → skos:narrower and vice versa

However, we found this infers too many triples. E.g. tgn:1000001 North and Central America has 853k descendants. These were repeated as the following properties, which caused the [Per-Entity Exports](#) of top places (e.g. World (Facet), North and Central America, etc) to become huge (over 50Mb. So we decided to remove the struck-out properties as superfluous:

- gvp:broaderPartitiveExtended, gvp:broaderExtended, gvp:broaderPreferredExtended (GVP properties)
- skos:broaderTransitive, iso:broaderPartitive (Standard properties)
- ~~gvp:narrower, gvp:narrowerExtended, iso:narrowerPartitive, skos:narrowerTransitive, iso:subordinateArray~~ (Symmetric properties going down)
- ~~skos:semanticRelation~~ (twice): very unspecific relation

The following diagram shows where we break the inference for SKOS (red ovals):



SKOS properties relating concepts

XKOS specification, (c) Data Documentation Initiative 2013

D disjoint with

I inverse of

S symmetric

T transitive

We run the following [SPARQL Update](#) queries after loading the ontologies.

- Reduce inference for SKOS and ISO

```
delete where {?x rdfs:subPropertyOf skos:semanticRelation};
delete {?x owl:inverseOf ?y}
  where {?x owl:inverseOf ?y
    filter (?x in
      (skos:broader,skos:narrower,skos:broaderTransitive,skos:narrowerTransitive,iso:broaderGeneric,iso:narrowerGeneric,iso:broaderPartitive,iso:narrowerPartitive,iso:broaderInstantial,iso:narrowerInstantial,iso:subordinateArray,iso:superOrdinate))
    filter (?y in
      (skos:broader,skos:narrower,skos:broaderTransitive,skos:narrowerTransitive,iso:broaderGeneric,iso:narrowerGeneric,iso:broaderPartitive,iso:narrowerPartitive,iso:broaderInstantial,iso:narrowerInstantial,iso:subordinateArray,iso:superOrdinate))};
```

- Remove the inference from ISO properties BTG/BTP/BTI to skos:broader, because in [ISO Rules](#) we first infer skos:broader, and then ISO BTG/BTP/BTI:

```
delete data {iso:broaderGeneric      rdfs:subPropertyOf skos:broader};
delete data {iso:broaderPartitive   rdfs:subPropertyOf skos:broader};
delete data {iso:broaderInstantial  rdfs:subPropertyOf skos:broader};
```

You don't lose querying expressivity, because SPARQL queries can access properties in both directions:

instead of this	use that
?x gvp:narrower ?y	?y gvp:broader ?x
?x gvp:narrowerExtended ?y	?y gvp:broaderExtended ?x
?x skos:narrower ?y	?y skos:broader ?x
?x iso:narrowerGeneric ?y	?y iso:broaderGeneric ?x
?x iso:narrowerPartitive ?y	?y iso:broaderPartitive ?x
?x iso:narrowerInstantial ?y	?y iso:broaderInstantial ?x
?x iso:narrower ?y	?y iso:broader ?x
?x iso:subordinateArray ?y	?y iso:superOrdinate ?x

If you dislike these reductions, load the standard ontologies (omitting the reduction queries above) and the [Explicit Exports](#) to your own repository

4.1.3 SKOS member vs memberList

The [SKOS Reference, sec 9. "Concept Collections"](#) has a semantic constraint (S36) that's relevant to [Sorting with Thesaurus Array](#). S36 requires that every item in a skos:memberList should also have a direct link skos:member. The [Open Annotation specification, sec 4.3 "List"](#) suggests a way to infer the direct links from the list using owl:propertyChainAxiom.

However, we have chosen to emit explicit skos:member links, because we do that for unordered collections anyway, and it would not have been easier to do differently for ordered collections. So this inference rule is not needed.

4.1.4 SKOS-XL Inference

The SKOS recommendation requires every **skosxl:Label** to also be present as a simple SKOS literal label (see [Dumbing-Down SKOSXL labels to SKOS lexical labels](#)). This requirement S55 is not formally stated in the [SKOS-XL ontology](#), but can be implemented in two ways

SKOS-XL Property Chains

We state the following [OWL Property Chains](#). (Note: we must use property chains for [BTG, BTP, BTI Inference](#), so we also use them for this inference.)

```
skos:prefLabel owl:propertyChainAxiom (skosxl:prefLabel skosxl:literalForm).
```

```
skos:altLabel owl:propertyChainAxiom (skosxl:altLabel skosxl:literalForm).
```

```
skos:hiddenLabel owl:propertyChainAxiom (skosxl:hiddenLabel skosxl:literalForm).
```

SKOS-XL Insert Queries

If you don't have an OWL RL compliant repository/reasoner, you can implement the required inference by running the following INSERT queries after loading the [Explicit Exports](#) files:

```
insert {?x skos:prefLabel ?z} where {?x skosxl:prefLabel ?y. ?y skosxl:literalForm ?z};
```

```
insert {?x skos:altLabel ?z} where {?x skosxl:altLabel ?y. ?y skosxl:literalForm ?z};
```

```
insert {?x skos:hiddenLabel ?z} where {?x skosxl:hiddenLabel ?y. ?y skosxl:literalForm ?z};
```

4.1.5 BTG, BTP, BTI Inference

To infer "Extended" versions (meaningful closures) of the [GVP Hierarchical Relations](#), we use the rules defined in the paper [Compositionality of ISO 25964 Hierarchical Relations](#) (V.Alexiev, J.Lindenthal, A.Isaac, May 2014, submitted). We summarize the rules in the following table.

- Each cell expresses a chain of two relations, and whether they can infer a third one
- The row gives the **left** relation, the column gives the **right** relation, and the cell gives the **conclusion**
- BT*x means "BT* or BT*E". In is a reference to the particular inference, used in the next section

- For each positive conclusion, we give an example; and for some negative we give a counter-example

left/right	BTGx	BTPx	BTIx
BTGx	I2: BTGE: numerous examples	I3: BTP: <i>beak irons</i> BTG <i>anvil components</i> BTP < <i>anvils and anvil accessories</i> >	I4: no
BTPx	I5: BTPE: <i>anvil components</i> BTP < <i>anvils and anvil accessories</i> > BTG < <i>forging and metal-shaping tools</i> >	I6: BTP: <i>Sofia</i> BTP <i>Bulgaria</i> BTP <i>Europe</i>	I7: no: <i>Sofia</i> BTP <i>Bulgaria</i> BTI <i>country</i> , but <i>Sofia</i> is no <i>country</i>
BTIx	I8: BTIE: <i>Mt Athos</i> BTI <i>orthodox religious center</i> BTG <i>Christian religious center</i>	I9: no, see below	I10: no

These inferences can be expressed in 3 equivalent ways:

- Using [SPARQL property path](#) notation (as in the paper)

```
left / right => conclusion
```

- Using [N3 Rules](#):

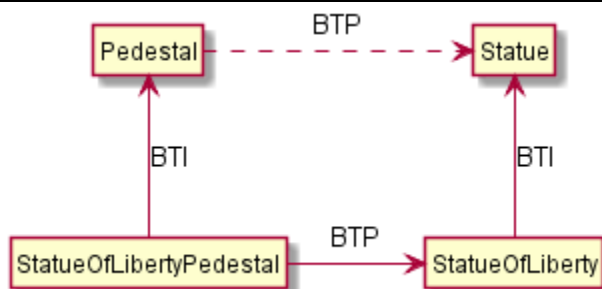
```
{?x left ?y. ?y right ?z} => {?x conclusion ?y}
```

- Using OWL property chains (as implemented in the next section)

```
conclusion owl:propertyChainAxiom (left right)
```

Regarding I9: Consider the example *Statue of Liberty pedestal* BTI *pedestals* BTP *statues*. The particular *Statue of Liberty pedestal* has no relation to *statues* in general (is neither an instance nor a part of *statues*). In the figure below we could infer the dashed relation (generalization of BTP from one instance thereof), i.e.

```
{?x BTI ?y. ?x BTP ?z. ?z BTI ?t} => {?y BTP ?t}
```



But in I9 we have only 3 nodes in sequence, not 4 nodes.

4.1.6 BTG, BTP, BTI Axioms

For these inferences we cannot use a finite number of INSERTs because the property chains involved are recursive. We need to use rule-based inference. OWLIM supports owl:propertyChainAxiom. First we feed BT* to BT*E:

```
# I1: BTG=>BTGE, BTP=>BTPE, BTI=>BTIE: basic inferences
gvp:broaderGeneric      rdfs:subPropertyOf gvp:broaderGenericExtended.
gvp:broaderPartitive    rdfs:subPropertyOf gvp:broaderPartitiveExtended.
gvp:broaderInstantial   rdfs:subPropertyOf gvp:broaderInstantialExtended.
```

These inferences are explained in the previous section:

```
# I2: BTGx/BTGx=>BTGE
gvp:broaderGenericExtended owl:propertyChainAxiom
  (gvp:broaderGenericExtended gvp:broaderGeneric).

# I3: BTGx/BTPx=>BTPE
gvp:broaderPartitiveExtended owl:propertyChainAxiom
  (gvp:broaderGeneric gvp:broaderPartitiveExtended).
# I5: BTPx/BTGx=>BTPE
gvp:broaderPartitiveExtended owl:propertyChainAxiom
  (gvp:broaderPartitiveExtended gvp:broaderGeneric).
# I6: BTPx/BTPx=>BTPE
gvp:broaderPartitiveExtended owl:propertyChainAxiom
  (gvp:broaderPartitiveExtended gvp:broaderPartitive).

# I8: BTIx/BTGx=>BTIE
gvp:broaderInstantialExtended owl:propertyChainAxiom
  (gvp:broaderInstantialExtended gvp:broaderGeneric).
```

This defines gvp:broaderExtended (BTE) as a disjunction of BTGE, BTPE, BTIE:

```
# I11: BTGE|BTPE|BTIE => BTE
gvp:broaderGenericExtended rdfs:subPropertyOf gvp:broaderExtended.
gvp:broaderPartitiveExtended rdfs:subPropertyOf gvp:broaderExtended.
gvp:broaderInstantialExtended rdfs:subPropertyOf gvp:broaderExtended.
```

4.1.7 broaderPreferredExtended Rules

We define gvp:broaderPreferredExtended as:

- Meaningful closure (appropriate extension) of gvp:broaderPreferred, or equivalently as
- Specialization of gvp:broaderExtended along broaderPreferred only.

```
gvp:broaderPreferredExtended <= gvp:broaderPreferred |
  (gvp:broaderPreferredExtended/gvp:broaderPreferred) & gvp:broaderExtended
```

The first line is a trivial rdfs:subPropertyOf. The second line fits the construct PropChainRestr:

```
gvp:Infer_broaderPreferredExtended a ptop:PropChainRestr;
  ptop:premise1 gvp:broaderPreferredExtended;
  ptop:premise2 gvp:broaderPreferred;
  ptop:restriction gvp:broaderExtended;
  ptop:conclusion gvp:broaderPreferredExtended.
```

The second line involves conjunction, so it cannot be implemented with OWL axioms. It also involves recursion, so it cannot be implemented with SPARQL INSERT, unless one is willing to run INSERTS many times, until nothing new is inferred. We would be interested to hear about efficient implementations using [SPIN Rules](#).

You can implement this property with OWL axioms and an insert query, at the expense of introducing an auxiliary property gvp:broaderPreferredTransitive (therefore more triples):

```
# Axioms
gvp:broaderPreferred rdfs:subPropertyOf gvp:broaderPreferredTransitive.
gvp:broaderPreferredTransitive a owl:TransitiveProperty.
# Insert Query
insert {?x gvp:broaderPreferredExtended ?y}
where {?x gvp:broaderPreferredTransitive ?y. ?x gvp:broaderExtended ?y}
```

4.1.8 ISO Insert Queries

We infer SKOS and ISO [Standard Hierarchical Relations](#) (skos:member and iso:superOrdinate; implementing [Hierarchy Structure](#) and [Sorting with Thesaurus Array](#) respectively). Run these queries after loading all Explicit statements:

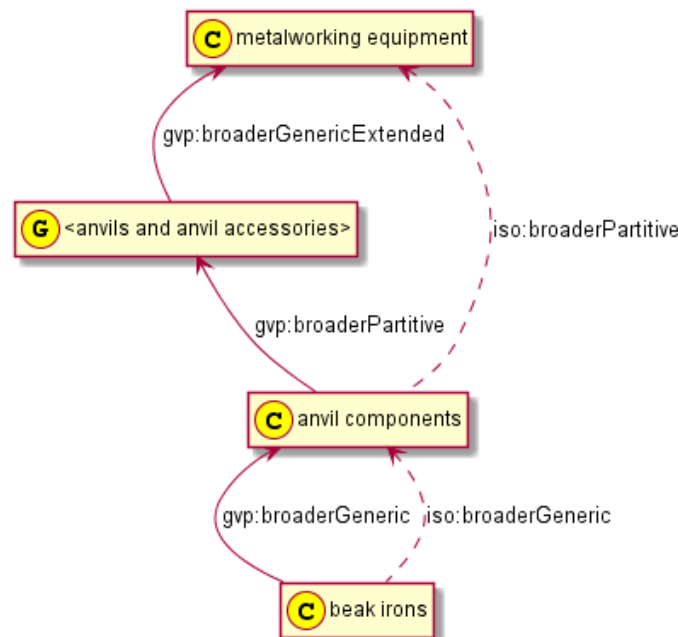
```
# Q1: add skos:member below each iso:ThesaurusArray
insert {?x skos:member ?y}
where {?y gvp:broader ?x. ?x a iso:ThesaurusArray};
# Q2: add iso:superOrdinate from iso:ThesaurusArray to skos:Concept
insert {?y iso:superOrdinate ?x}
where {?y gvp:broader ?x. ?x a skos:Concept. ?y a iso:ThesaurusArray};
```

The query below is not used anymore, since we decided not to emit [Top Concept](#) indication:

```
# Q3: add skos:topConceptOf for those Concepts having no broader Concept in the same scheme
insert {?x skos:topConceptOf ?scheme}
where {?x a skos:Concept; skos:inScheme ?scheme
  filter not exists {?x gvp:broader ?y. ?y a skos:Concept; skos:inScheme ?scheme}};
```

4.1.9 ISO Rules

We infer skos:broader and the ISO properties BTG/BTP/BTI as restrictions of GVP BTGE/BTPE/BTIE (the "Extended" relations implemented above), when the "Extended" relation connects two Concepts directly. Thus the SKOS and ISO properties "thread" the skos:Concept hierarchy as a subset of the complete gvp:Subject hierarchy. E.g.:



Unfortunately this cannot be implemented with an "insert where not exists" query, e.g.:

```
# Q4: is wrong because in addition to the indirect path through ?y, there may be a direct path
insert {?x iso:broaderGeneric ?z}
where {?x a skos:Concept. ?z a skos:Concept. ?x gvp:broaderGenericExtended ?z
  filter not exists {
    ?y a skos:Concept. ?x gvp:broaderGenericExtended ?y. ?y gvp:broaderGenericExtended ?z}};
```

There are directly related concepts that are also indirectly related, as you can verify with this query:

```
select * {
  ?x gvp:broader ?y. ?y gvp:broader ?z. ?x gvp:broader ?z.
  ?x gvp:prefLabelGVP [xl:literalForm ?xLab]. ?x a skos:Concept.
  ?y gvp:prefLabelGVP [xl:literalForm ?yLab]. filter not exists {?y a skos:Concept}.
  ?z gvp:prefLabelGVP [xl:literalForm ?zLab]. ?z a skos:Concept}
```

E.g. aat:300025276 mosaicists is related directly to aat:300025103 artists (visual artists), but also indirectly through aat:300386777 <artists by medium or work type>.

We first define an auxiliary property **gvp:broaderNonConcept**: a chain of gvp:broader from Concept to GuideTerms, without intervening Concept. Notes:

- We don't care about chains to Hierarchies or Facets, because there are no Concepts above them, see [Hierarchy Structure](#)
- Such chains are not limited to a specific length. E.g. here is a chain Concept-GuideTerm-GuideTerm-Concept:
hods - <plaster, concrete and mortar working equipment> - <equipment by material processed> - equipment.

So gvp:broaderNonConcept is defined with recursive rules:

```
gvp:broaderNonConcept <= [gvp:Concept] gvp:broader [gvp:GuideTerm] |
gvp:broaderNonConcept / gvp:broader [gvp:GuideTerm]
```

This fits the constructs TypeRestr and PropChainType2:

```
gvp:Infer_broaderNonConcept_TypeRestr a ptop:TypeRestr;
ptop:premise gvp:broader;
ptop:type1 skos:Concept;
ptop:type2 gvp:GuideTerm;
ptop:conclusion gvp:broaderNonConcept.
gvp:Infer_broaderNonConcept_PropChainType2 a ptop:PropChainType2;
ptop:premise1 gvp:broaderNonConcept;
ptop:premise2 gvp:broader;
ptop:type2 gvp:GuideTerm;
ptop:conclusion gvp:broaderNonConcept.
```

Now we infer **skos:broader** as either connecting two skos:Concept directly, or an extension of gvp:broaderNonConcept:

```
skos:broader <= [skos:Concept] gvp:broader [skos:Concept] |
gvp:broaderNonConcept / gvp:broader [skos:Concept]
```

This again fits the constructs TypeRestr and PropChainType2:

```
gvp:Infer_skosBroader_TypeRestr a ptop:TypeRestr;
ptop:premise gvp:broader;
ptop:type1 skos:Concept;
ptop:type2 skos:Concept;
ptop:conclusion skos:broader.
gvp:Infer_skosBroader_PropChainType2 a ptop:PropChainType2;
ptop:premise1 gvp:broaderNonConcept;
ptop:premise2 gvp:broader;
ptop:type2 skos:Concept;
ptop:conclusion skos:broader.
```

Finally, we infer **ISO BTG/BTP/BTI** as restrictions of GVP BTGE/BTPE/BTIE. We use the fact that skos:broader links pairs of directly connected Concepts.

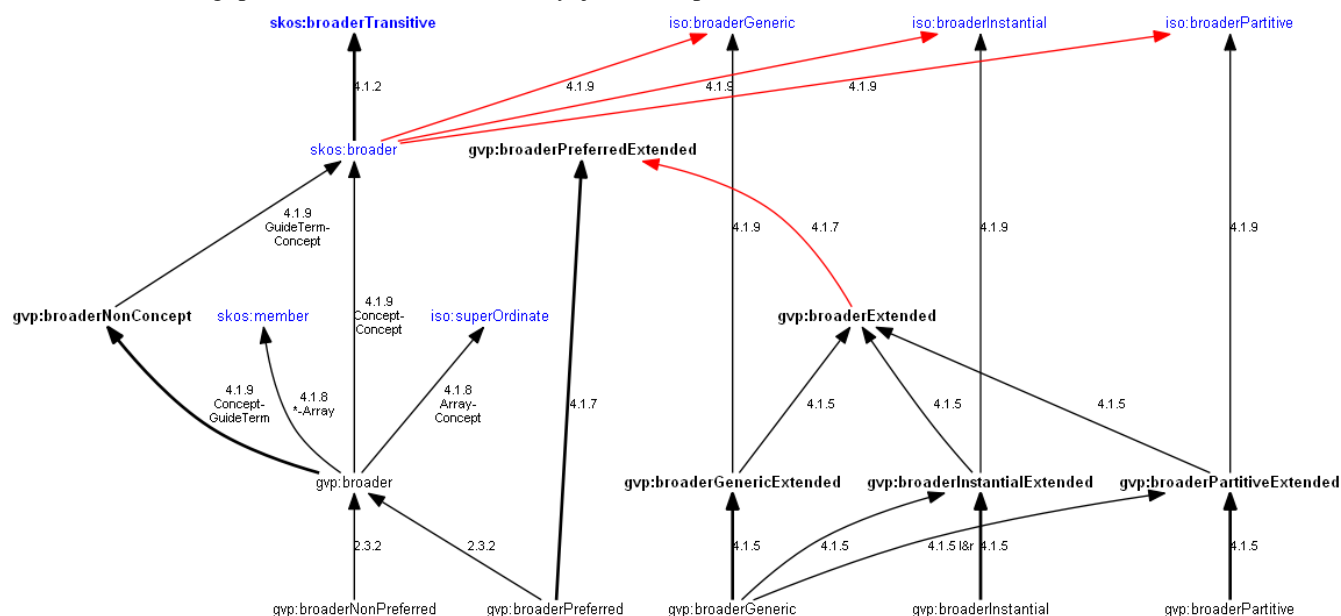
```
gvp:Infer_isoBroaderGeneric a ptop:PropRestr;
ptop:premise gvp:broaderGenericExtended;
ptop:restriction skos:broader;
ptop:conclusion iso:broaderGeneric.
gvp:Infer_isoBroaderPartitive a ptop:PropRestr;
ptop:premise gvp:broaderPartitiveExtended;
ptop:restriction skos:broader;
ptop:conclusion iso:broaderPartitive.
gvp:Infer_isoBroaderInstantial a ptop:PropRestr;
ptop:premise gvp:broaderInstantialExtended;
ptop:restriction skos:broader;
ptop:conclusion iso:broaderInstantial.
```

4.1.10 Hierarchical Relations Inference

The following diagram shows the inference paths between hierarchical relations. Legend:

- The numbers refer to sections in this document (e.g. 4.1.5), or specific queries/rules within (e.g. 4.1.8 Q1)

- Standard relations (2.3.1) are shown in blue, GVP relations (2.3.2) in black
- Bold font shows a "closure" relation: transitive or "proper closure" (see 4.1.4). Bold arrow connects the relation being closed (iterated)
- Red arrow shows a restriction relation: the target relation is restricted by the source relation
- "l&r" means that the step property (broaderGeneric) can be to the left or right of the closure (broaderPartitiveExtended); in other property chains the step property can be only to the right
- "Concept-Array" means the target relation is a restriction of the source relation over immediately connected nodes of type `skos:Concept` and `iso:ThesaurusArray`. "Array-*" is a restriction from `iso:ThesaurusArray` to any immediately connected node. "Concept-Concept" is a restriction over two immediately connected `skos:Concepts`.
- Note: the source relations `gvp:broader(|Non)Preferred` and `gvp:broader(Generic|Partitive|Instantial)` are always used together (a total of 6 combinations), and `gvp:broader` can be inferred from any one of them. The diagram shows inference from `gvp:broader(|Non)Preferred` only, just to keep it tidier.



4.1.11 FTS Insert Queries

OWLIM includes a [Full-Text Search extension](#). We use it to provide [Full Text Search](#) in the UI (also see [Full Text Search Query](#)). To create the two FTS indexes, use the following queries. (`luc:setParam` is a space-separated list of the full URLs of properties to include in indexing):

```
PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
INSERT DATA {
  luc:includePredicates luc:setParam "http://www.w3.org/2008/05/skos-xl#prefLabel
http://vocab.getty.edu/ontology#term http://www.w3.org/2008/05/skos-xl#altLabel
http://purl.org/dc/elements/1.1/identifier" .
  luc:index luc:setParam "uri" .
  luc:moleculeSize luc:setParam "3"};
INSERT DATA { luc:term luc:createIndex "true" };
```



```
INSERT DATA {
  luc:includePredicates luc:setParam "http://www.w3.org/2004/02/skos/core#prefLabel
http://www.w3.org/2004/02/skos/core#altLabel http://www.w3.org/2004/02/skos/core#scopeNote
http://www.w3.org/2008/05/skos-xl#literalForm http://www.w3.org/1999/02/22-rdf-syntax-ns#value
http://purl.org/dc/elements/1.1/identifier" .
  luc:index luc:setParam "uri" .
  luc:moleculeSize luc:setParam "2"};
INSERT DATA { luc:text luc:createIndex "true" };
```

If you don't use OWLIM, you'd need to use the repository-specific FTS extensions of the repository that you use.

4.1.12 OntoGeo Insert Query

OWLIM provides some [Geo-spatial Extensions](#), see example of their use in [TGN-Specific Queries](#). To enable them, we need to create the OntoGeo geospatial index. This is done with a query like this:

```
PREFIX ontogeo: <http://www.ontotext.com/owlim/geo#>
INSERT DATA { _:b1 ontogeo:createIndex _:b2. }
```

4.2 Alignment

A key potential benefit of LOD is the ability to create and exploit linkages between datasets, e.g. alignments. AAT provides a few alignments, and GVP hopes that external contributors (such as [VUA's Amalgame project](#)) will provide more.

4.2.1 LCSH Alignment

We generate some (about 300) alignments to LCSH based on explicit mention of LCSH ID in bibo:locator of a [Local Source](#):

- When the source dct:isPartOf aat_source:2000046735 (LC Subject Authority Headings) and bibo:locator includes a number of >=8 digits, OR
- When bibo:locator consists of a prefix "sh", followed by a number of >=8 digits

This check is performed for sources at both Subject and Term level. The alignment is emitted for the AAT subject.

Take for example aat:300008736 "waterfalls" (see <http://vocab.getty.edu/aat/300008736.ttl>):

```
aat:300008736 a gvp:Concept ;
  gvp:prefLabelGVP aat_term:1000008736-en ;
  gvp:prefLabelLoC aat_term:1000008736-en .
aat_term:1000008736-en a skosxl:Label
  gvp:sourcePreferred aat_source:2000046735-term-1000008736 .
aat_source:2000046735-term-1000008736 a bibo:DocumentPart ;
  dct:isPartOf aat_source:2000046735 ;
  bibo:locator "sh 85145720" .
aat_source:2000046735 a bibo:Document ;
  bibo:shortTitle "LC Subject Authority Headings [online] (2002-)" .
```

It has a Term (prefLabel) that has a Local Source (bibo:DocumentPart) that is both part of LCSH, and has a bibo:locator matching the "sh" ID pattern. From this we infer the alignment:

```
aat:300008736
  skos:exactMatch <http://id.loc.gov/authorities/subjects/sh85145720>.
```


4.2.2 AATNed Alignment

AATNed is the project producing the Dutch translation of AAT. It started publishing LOD earlier than AAT, so some institutions (e.g. the Rijksmuseum) have already started using their URLs (e.g. <http://service.aat-ned.nl/skos/300024521>). AATNed has made the decision to merge into AAT, so the new GVP URLs should be used (e.g. <http://vocab.getty.edu/aat/300024521>). The IDs correspond, so there is no need to provide alignment links

```
<http://service.aat-ned.nl/skos/300024521>
dct:isReplacedBy <http://vocab.getty.edu/aat/300024521>
```

4.3 Forest UI

Forest is a UI framework for creating semantic applications by Ontotext. GVP uses a customized version of Forest that provides the following features (the red numbers on the screen-shots are explained in the text below).


Getty Vocabularies: LOD
SPARQL

(2)

SPARQL Query

Query:

```

1 # All Info About a Subject
2 CONSTRUCT {
3   ?s ?p1 ?o1. # subject
4   ?t ?p3 ?o3. # term/note
5   ?ss ?p4 ?o4. # subject local source
6   ?ts ?p6 ?o6. # term/note local source
7   ?st ?p7 ?o7. # statement about relations of subject
8   ?ar ?p8 ?o8. # anonymous array of subject
9   ?l1 ?p9 ?o9. # list element of subject
10  ?l2 ?p0 ?o0. # list element of anonymous array
11 } WHERE {
12   BIND (aat:300107346 as ?s)
13   { ?s ?p1 ?o1 }
14   UNION { ?s dct:source ?ss. ?ss a bibo:DocumentPart. ?ss ?p4 ?o4 }
15   UNION { ?s skos:scopelabel ?skosx1:preflabel ?skosx1:altlabel ?t.
16

```

☒ Include inferred (4)
 ☐ Expand results over equivalent URIs (5)

Sample queries:
Append predefined namespaces: (6)

SPARQL Select template, Concept by full label, without giving any language tag, Top-level Concepts, Descendants of a Given Parent, Subjects by Contributor Abbrev, Subjects by Contributor Id, Subject Preferred Label, Subjects in Order, Subject by Term (FTS), Subject by Text (FTS), Subject by Text (more info), All Info About a Subject (7) Historic Info on Relations

- [Full Text Search](#) (1)
- SPARQL query endpoint (2), supporting SPARQL 1.1
 - Accessible both through REST URL, and through an interactive form with syntax highlighting and auto-indent (3)
 - You can specify whether to return only Explicit triples, or also Inferred triples (4)
 - You can specify whether to expand results across owl:sameAs and return owl:sameAs assertions. (This applies to [Language Dual URLs](#)) (5)
 - All [External Prefixes](#) and [GVP Prefixes](#) used by the representation are predefined in the repository, so you don't need to add them. There is a function to append a predefined namespace (6), but you won't need to use it, except to examine these namespaces
- [Sample Queries](#), accessible through links below the SPARQL query box (7)

Results for # [Subject by Term](#) (8) ... (44) (9)

(10) Download SPARQL Results in: [JSON](#) | [XML](#)

Subject	Term	Parents	ScopeNote	Type
aat:300310116	vessel stands@en	<stands by function>, stands (support furniture), ... Objects Facet	Refers generally to objects designed specifically to support vessels such as Roman kraters or Chinese bronze ritual vessels.@en	Concept
aat:300041950	vessel rattles@en	rattles, indirectly struck idiophones, ... Objects Facet	Rattles enclosing small objects that strike against each other or the walls of the rattles when they are shaken.@en	Concept
aat:300195368	concussion vessels@en	directly struck idiophones, struck idiophones, ... Objects Facet	Concussion idiophones made of hollowed-out objects.@en	Concept

- Editing of the previous query (8).
Click on the query link "Results for..." and **not** on the SPARQL link in the header
- Number displayed and total number of results (9).

- Returning query results in HTML, RDF/XML, Turtle, NTriples, JSON (10).
RDF/XML and Turtle are available only for CONSTRUCT queries
- Query result pagination (for HTML)
- [Semantic Resolution](#) of resources (GVP URLs), including content negotiation

vessel stands (12)

Source: <http://vocab.getty.edu/aat/300310116>

Subject (31)

Predicate

Object

All (13)

(15) Website

(17) Hierarchy

(11) Download in: JSON | RDF | N3/Turtle | N-Triples

(14) Inference

Explicit only

Statements in which the resource exists as a subject.

Predicate	Object
rdf:type	gvp:Concept
rdfs:seeAlso	http://www.getty.edu/vow/AATFullDisplay?find=&logic=AND&note=&subjectid=300310116 (16)
dcterms:contributor	aat_contrib:10000000
skos:scopeNote	aat_scopeNote:56940
skos:inScheme	aat
skos:prefLabel	vessel stands@en
skos:altLabel	vessel stand@en
skos:changeNote	aat_rev:5001917793 , aat_rev:5001917794 , aat_rev:5001917795 , aat_rev:5001917797 , aat_rev:5001917798 , aat_rev:5001917799 , aat_rev:5001917800 , aat_rev:5002044453 (18)
xl:prefLabel	aat_term:1000394534-en

Resource representations in HTML, RDF/XML, Turtle, NTriples, JSON (11).

- Label of the resource. For subjects this is usually but not always the first prefLabel (12)
- Tabs to show triples where the resource plays different roles (subject, predicate, object) or all triples (13)
- Selector whether to include Explicit, Inferred or all triples (14)
- For the semantic formats (RDF/XML, Turtle, NTriples, JSON), all triples are returned
- For the independent entities (Subjects, Sources, Contributors), the semantic formats include all triples of all owned objects as well (see [Per-Entity Exports](#))
- Link to the Subject page on GVP's website (15).
This is the same link as rdfs:seeAlso (16) but it would take an extra click if you use that.
- A link to the GVP page showing the Subject's position in the Hierarchy (17)
- Conversely, the GVP site has links to download each of the semantic formats.
- Clickable links to explore all related resources (18)

4.4 Full Text Search

You can search for Subjects using the Full Text Search box. Two indexes are provided that can be selected with a drop-down:

- **Brief** (luc:term): includes all terms and subject ID.
- **Full** (luc:text): includes all terms, qualifiers, subject ID, and scope notes.

This search uses the Lucene FTS engine that is built into OWLIM.

The search results include the following columns: subject ID (with link), GVP-preferred term (gvp:prefLabelGVP), abbreviated parent string, abbreviated scope note, and subject type).

- Result pagination is provided
- All language representations are searched, but the results are returned always in English.

The following pre-processing of the query phrase is performed:

- Characters that are not letter/digit are replaced with a space.
- Words are wild-carded with *

- A conjunction (AND) is added between the words
 - For Chinese hieroglyphs, no analysis is performed and the entered hieroglyphs must match exactly.
- For programmatic querying, use the predicates luc:text and luc:term in SPARQL (see [Full Text Search Query](#)).

4.5 Descriptive Information

Notice: this section describes AAT only, not yet TGN. TGN info will be added in a future release.

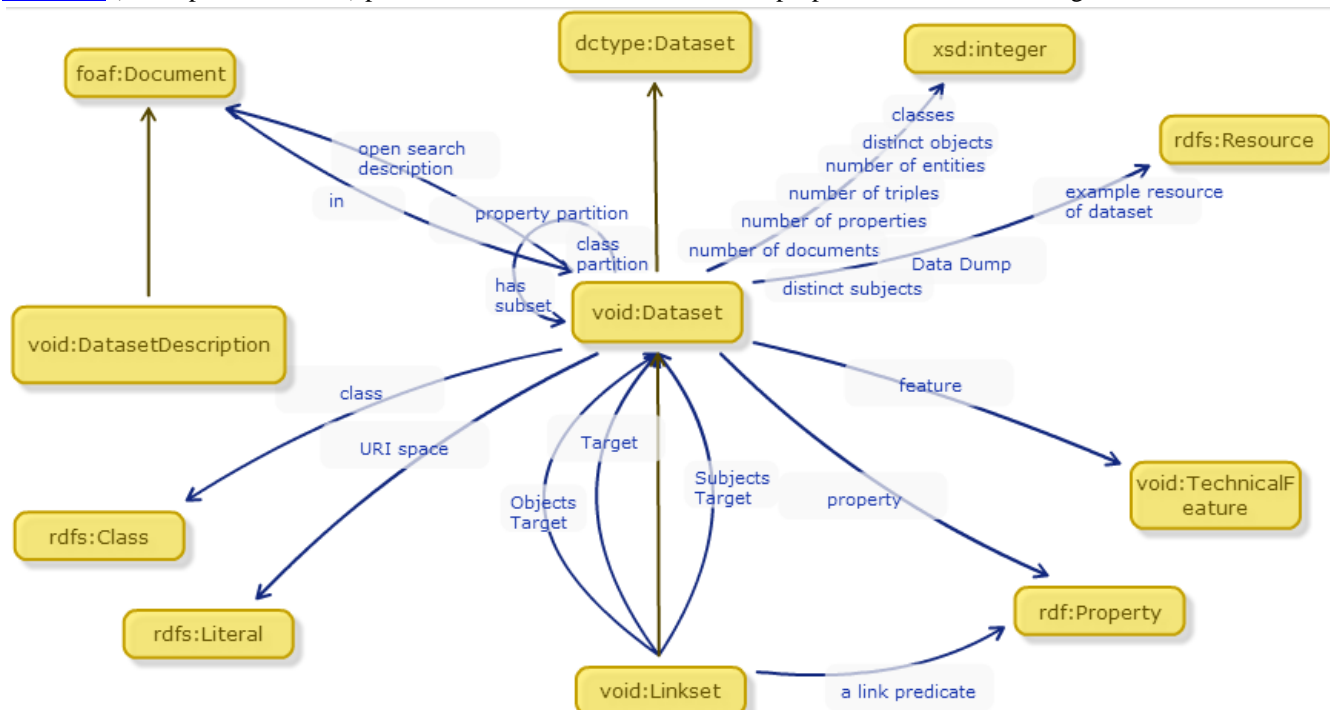
Machine-readable descriptive information is crucial to allow semantic agents to discover, register, crawl, analyze and summarize datasets. It provides the backbone of information for LOD registries such as the DataHub (<http://datahub.io>). Yet, the creators of the famous LOD cloud diagram (<http://lod-cloud.net>) report that many datasets are missing basic descriptive and licensing information, which makes it harder for people and agents to consume LOD.

We provide comprehensive info about the AAT dataset, ontology and concept scheme.

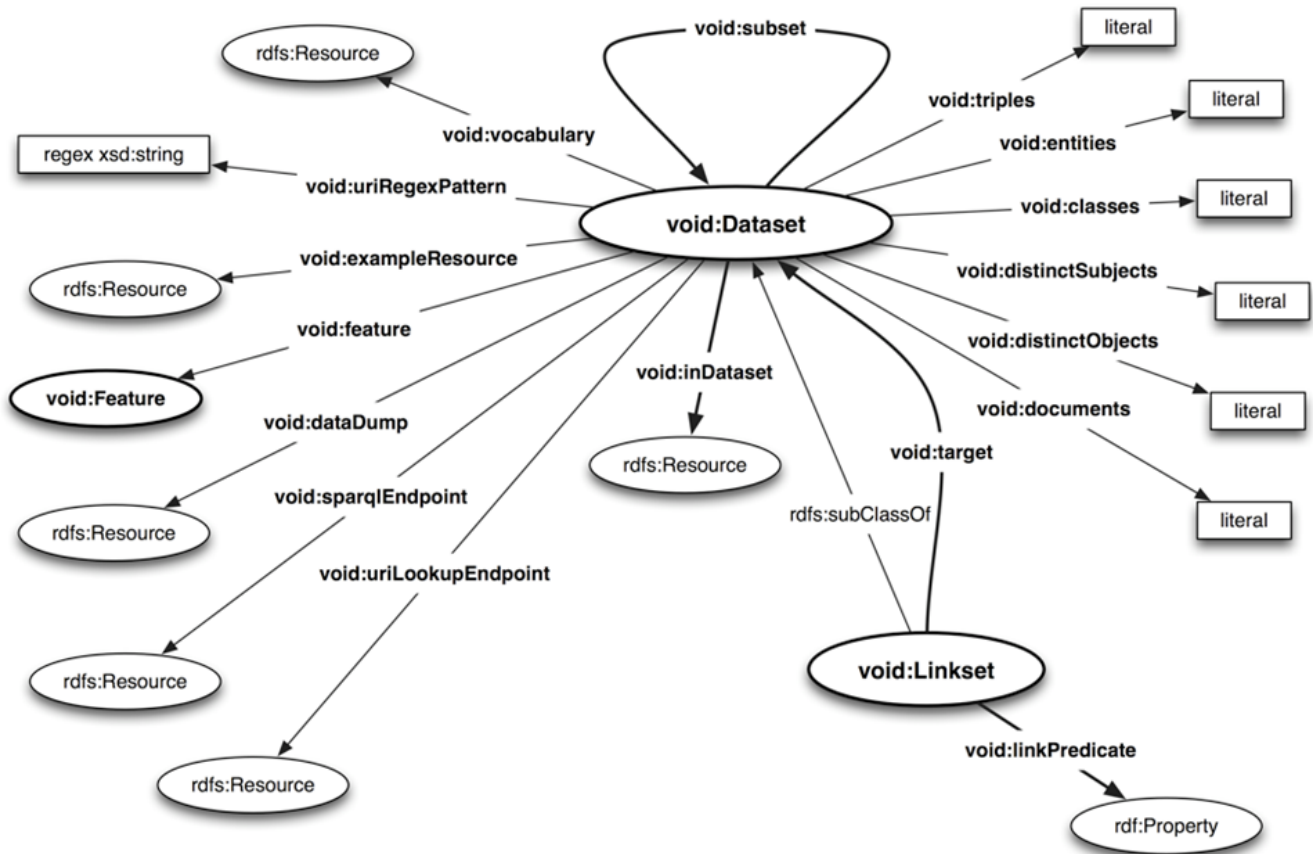
- We used this [useful list](#) of metadata-description vocabularies from LOV
- We use properties from most of the vocabularies listed in [Descriptive Prefixes](#). Why so many? Because there is overlap between the different vocabularies, yet each has something extra to say
- The basic ontologies are VOID, DCAT (not to be confused with DCT!), ADMS, CC. They are described in the subsections below.
- We also use the ubiquitous DC, DCT; and a few properties from DCTYPE, VANN, VOAG, WDRS, WV.
- The AAT dataset is already registered at the DataHub: <http://datahub.io/dataset/getty-aat>
- We hope that our descriptive info fulfills the [Guidelines for Collecting Metadata on Linked Datasets in the Data Hub](#), and intend to validate this with <http://validator.lod-cloud.net/>

4.5.1 VOID

Vocabulary of Interlinked Datasets (VOID) is the main ontology for describing RDF datasets. The VOID specification [Describing Linked Datasets with the VoID Vocabulary](#) was published by W3C on 3 March 2011. The [VoID vocabulary definition](#) (namespace document) provides a reference of all classes and properties, and the following domain model:



The slideshare presentation [VOID: Metadata for RDF Datasets](#) (Richard Cyganiak, May 14, 2012, p.13) provides a more lucid domain model:

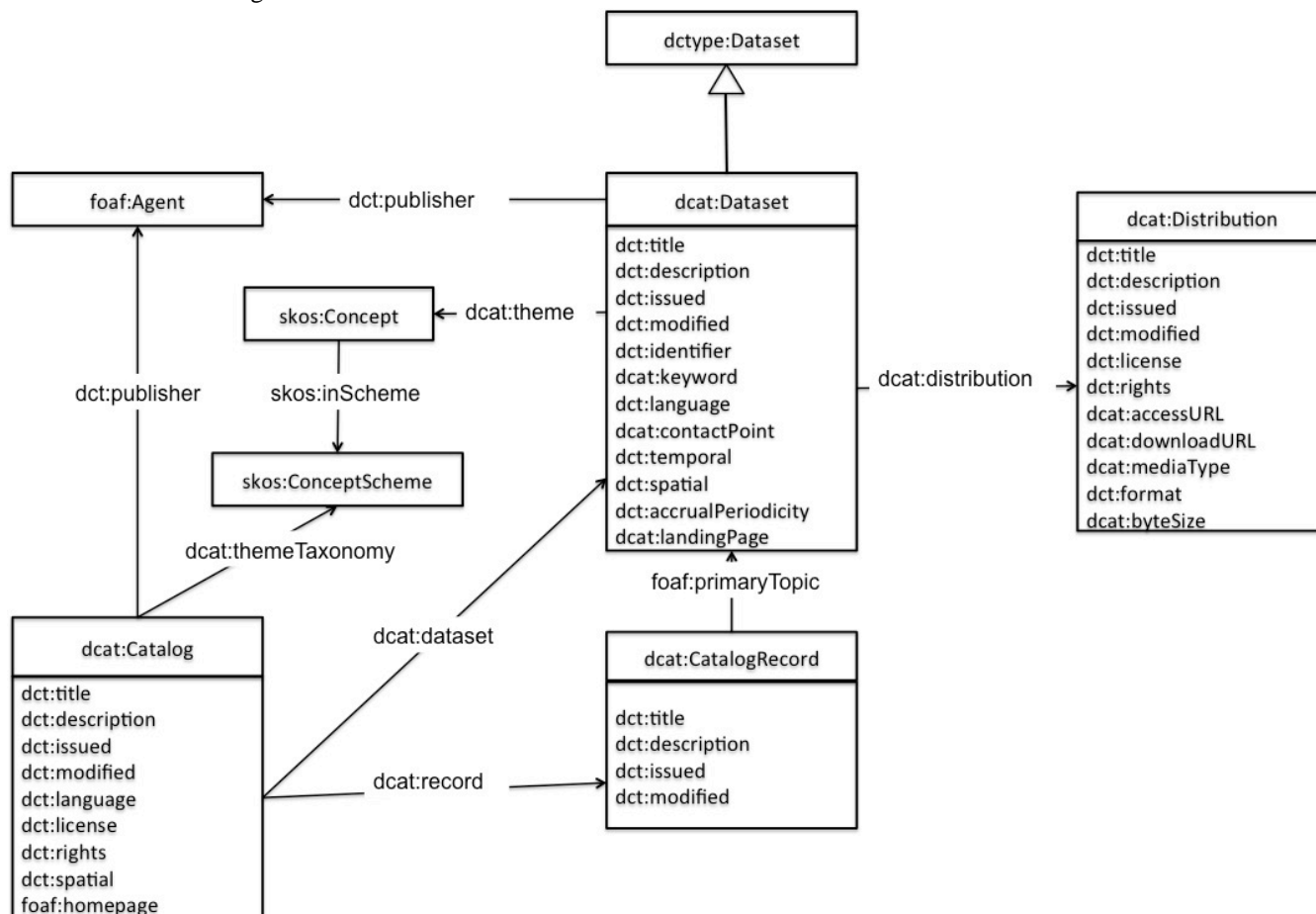


VOID covers a number of areas:

- Descriptive info (who, when, what) using DC, DCT
- Structural info, interlinking the dataset, its description, data dumps, SPARQL endpoint, used ontology, etc
- Access URLs and mechanisms, e.g. RDF dumps and SPARQL endpoint
- Vocabularies, properties and classes used
- Statistics about size (number of triples), including per property/class
- Resource URI patterns

4.5.2 DCAT

The [Data Catalog Vocabulary](#) (DCAT) is a W3C Recommendation published on 16 January 2014. It is used to describe datasets and data catalogs.

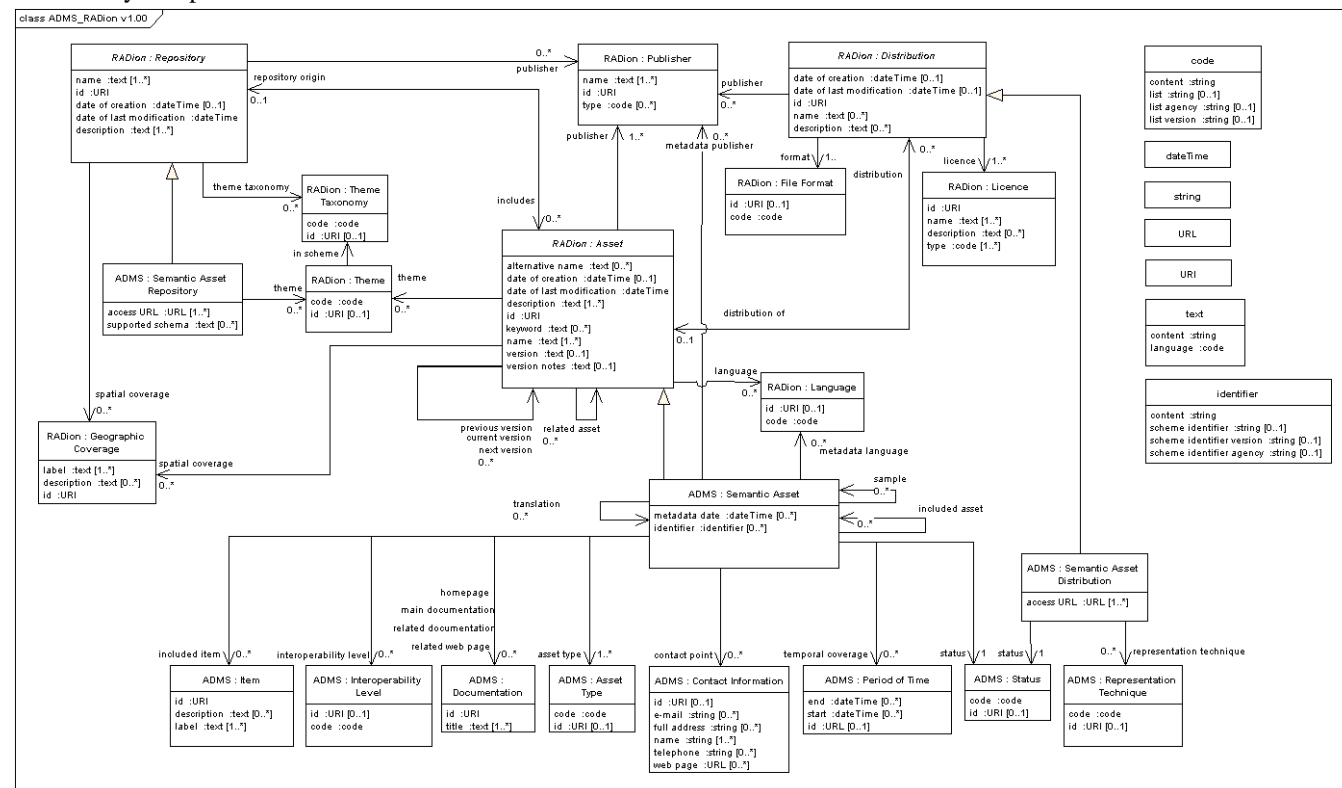


4.5.3 ADMS

The Asset Description Metadata Schema (ADMS) is a metadata vocabulary created by the Interoperability Solutions for European Public Administrations (ISA) Programme of the European Commission.

- While DCAT describes data sets, ADMS describes reusable metadata (e.g. xml schemata, generic data models) and reference data (e.g. code lists, taxonomies, dictionaries, vocabularies).
- While DCAT is focused on data catalogs, ADMS is focused on the assets within a catalog

Version 1.00 was released on 18 April 2012. The [ADMS Conceptual Model](#) is based on an earlier ontology called RADION and is fairly complex:



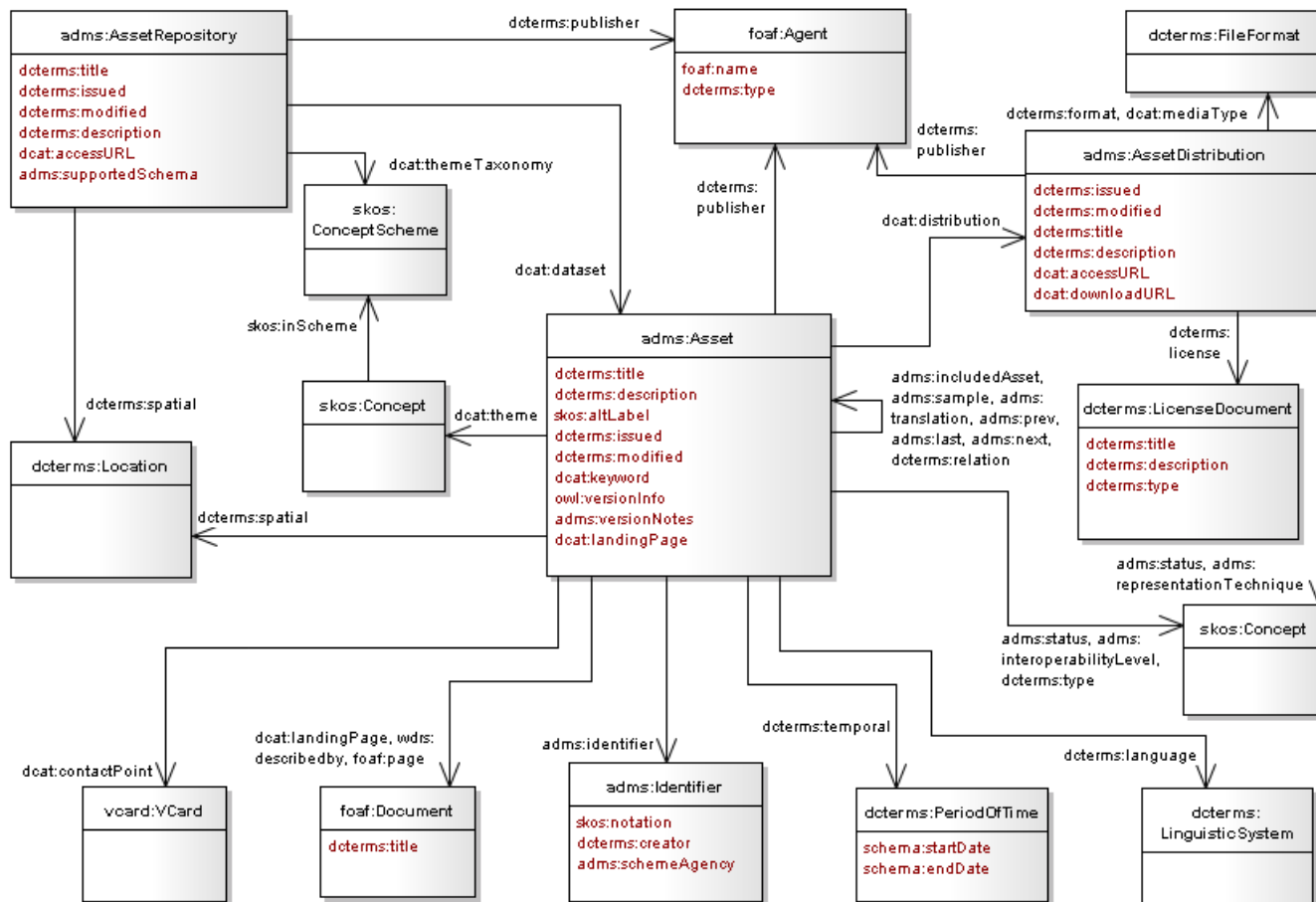
A [spreadsheet template](#) is provided, so a user can fill your dataset's metadata, and then Google Refine can create appropriate RDF.

The [EU Open Data Portal](#) uses a metadata vocabulary (EC-ODP) that's quite close to ADMS. For example, see the [description](#) of EuroVoc (EU's multilingual thesaurus), and the corresponding [RDF file](#)

4.5.3.1 W3C ADMS

After version 1.00, ADMS was contributed to W3C's [Government Linked Data Working Group](#) for further development. W3C streamlined the ADMS and converted it to a DCAT profile. This version of ADMS re-uses and subclasses DCAT and other standard vocabularies (e.g. SKOS, DCT) wherever possible and therefore defines a minimal set of classes and properties of its own. The ADMS specification was published on 1 August 2013 as a W3C note.

The domain model was simplified significantly. We work with this streamlined W3C version.

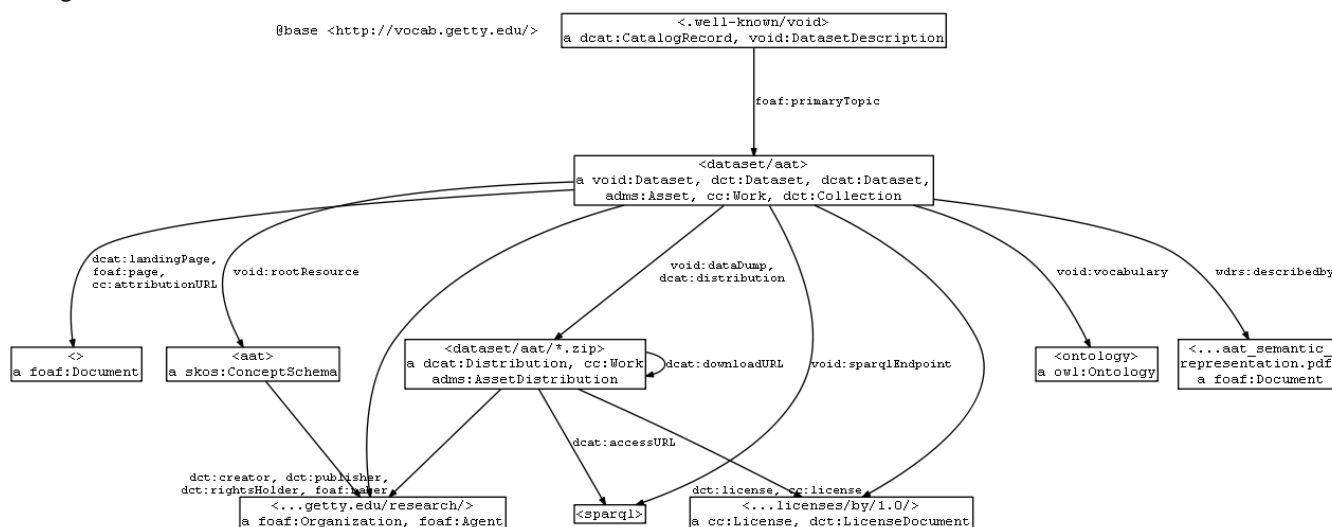


4.5.4 Descriptive Entities

We describe the following entities. Luckily, the domain models of VOID, DCAT and ADMS are fairly well aligned, so the entities can be assigned a consistent set of classes. Then the entities are interlinked with [Descriptive Relations](#), more info is attached as [Descriptive Properties](#), and more is computed as [Dynamic Descriptive Properties](#).

entity	URL	classes
Descriptor	http://vocab.getty.edu/.well-known/void	void:DatasetDescription, dcat:CatalogRecord
AAT dataset	http://vocab.getty.edu/dataset/aat	void:Dataset, dct:Dataset, dcat:Dataset, adms:Asset, cc:Work, dct:Collection
Home page	http://vocab.getty.edu/	foaf:Document
AAT thesaurus	http://vocab.getty.edu/aat/	skos:ConceptSchema
Explicit Exports	http://vocab.getty.edu/dataset/aat/explicit.zip	dcat:Distribution, adms:AssetDistribution, cc:Work
Total Exports	http://vocab.getty.edu/dataset/aat/full.zip	
GVP ontology	http://vocab.getty.edu/ontology	owl:Ontology
Documentation	http://www.getty.edu/research/tools/vocabularies/loa/aat_semantic_representation.pdf	foaf:Document
Creator/publisher	http://www.getty.edu/research/	foaf:Organization, foaf:Agent
SPARQL endpoint	http://vocab.getty.edu/sparql	
License	http://opendatacommons.org/licenses/by/1.0/	cc:License, dct:LicenseDocument

A diagram of the entities and their relations follows:



4.5.5 Descriptive Relations

The main descriptive entities are linked with the following relations:

subjects	relations	objects
http://vocab.getty.edu/.well-known/void	foaf:primaryTopic	http://vocab.getty.edu/dataset/aat
http://vocab.getty.edu/dataset/aat	dcat:landingPage, foaf:homepage, cc:attributionURL	http://vocab.getty.edu/
http://vocab.getty.edu/dataset/aat	wdrs:describedby	http://www.getty.edu/research/tools/vocabularies/lod/aat_semantic_representation.pdf
http://vocab.getty.edu/dataset/aat	void:vocabulary	http://vocab.getty.edu/ontology ¹
http://vocab.getty.edu/dataset/aat	void:rootResource	http://vocab.getty.edu/aat/ ²
http://vocab.getty.edu/dataset/aat	void:sparqlEndpoint	http://vocab.getty.edu/sparql
http://vocab.getty.edu/dataset/aat	void:dataDump, dcat:distribution	http://vocab.getty.edu/dataset/aat/explicit.zip http://vocab.getty.edu/dataset/aat/full.zip
http://vocab.getty.edu/aat/ http://vocab.getty.edu/dataset/aat/explicit.zip http://vocab.getty.edu/dataset/aat/full.zip http://vocab.getty.edu/dataset/aat	dct:creator, dct:publisher, dct:rightsHolder, foaf:maker	http://www.getty.edu/research/
http://vocab.getty.edu/dataset/aat/explicit.zip	dcat:downloadURL	http://vocab.getty.edu/dataset/aat/explicit.zip
http://vocab.getty.edu/dataset/aat/full.zip		http://vocab.getty.edu/dataset/aat/full.zip
http://vocab.getty.edu/dataset/aat/explicit.zip http://vocab.getty.edu/dataset/aat/full.zip	dcat:accessURL	http://vocab.getty.edu/sparql

4.5.6 Descriptive Properties

We use the following descriptive properties and values.

- We use appropriate [ADMS SKOS concepts](http://purl.org/adms/) from namespace <http://purl.org/adms/> (Turtle). Their URLs are self-describing.
- The distribution of properties amongst entities is dictated by the domain models above (and see the actual VOID descriptor)

entity	property	values
descriptor	dct:title	"AAT description document (VOID file)"
descriptor	dct:created	"2014-03-20"^^xsd:date
descriptor	dc:format	"meta/void"
thesaurus	dct:title	"Art & Architecture Thesaurus (AAT) ®"
ontology	rdfs:label	"Getty Vocabulary Program ontology"
ontology	vann:preferredNamespacePrefix	"gvp"

¹ And the other [External Ontologies](#) used by GVP

² And all gvp:Facets, see next section

entity	property	values
ontology	vann:preferredNamespaceUri	"http://vocab.getty.edu/ontology#"
ontology	dc:format	"meta/rdf-schema"
publisher	foaf:name, rdfs:label	"Getty Research Institute"
dataset	dcat:contactPoint	[vcard:email <mailto:VocabLOD@getty.edu>]
publisher	dct:type	http://purl.org/adms/publishertype/NonProfitOrganisation
dataset	dct:title	"AAT Linked Open Data (LOD) Dataset"
dataset, ontology, exports	dct:created	"2014-02-20"^^xsd:date
dataset	dcat:keyword	"Thesauri"
dataset	dcat:theme, dct:subject	aat:300026677, http://id.loc.gov/authorities/subjects/sh85134827 , http://dbpedia.org/resource/Thesaurus
dataset	void:exampleResource, adms:sample	aat:300264092, # Objects Facet aat:300264551, # Furnishings and Equipment (Hierarchy Name) aat:300197200, # <containers by function or context> aat:300198841. # Rhyta
dataset	dct:language	gvp_lang:en, gvp_lang:nl, gvp_lang:es, gvp_lang:zh ³
dataset	voag:frequencyOfChange, dct:accrualPeriodicity	voag:BiWeekly ⁴
dataset	dct:source	http://www.getty.edu/research/tools/vocabularies/aat/
dataset	dct:type	http://purl.org/adms/assettype/Thesaurus
dataset	adms:interoperabilityLevel	http://purl.org/adms/interoperabilitylevel/Semantic ⁵
dataset	adms:representationTechnique	http://purl.org/adms/representationtechnique/SKOS
dataset	adms:status	http://purl.org/adms/status/Completed
dataset	vann:preferredNamespacePrefix	"aat"
dataset	vann:preferredNamespaceUri	"http://vocab.getty.edu/aat/"
dataset	void:uriSpace	"http://vocab.getty.edu/aat/"
dataset, ontology	owl:versionInfo	"1.0"
dataset	void:feature	fmt:N-Triples, fmt:RDF_XML, fmt:Turtle, fmt:SPARQL_Results_XML, fmt:SPARQL_Results_JSON
exports	dc:format	"application/n-triples", "application/zip"
	dcat:mediaType, dct:format	http://provenanceweb.org/format/mime/application/n-triples , http://provenanceweb.org/format/mime/application/zip (*)
explicit	dct:title	"Explicit AAT statements"

³ Only the main AAT languages. Chinese (gvp_lang:zh) covers about 25% of all subjects)

⁴ Frequency of data update and adding new Subjects. SDMX provides <http://purl.org/linked-data/sdmx/2009/code#freq-W> but that is "Weekly"

⁵ See [European Interoperability Framework \(EIF\)](#)

entity	property	values
exports	dct:description	"NTriples zip, file size is approximate. First load ontologies, then files in indicated order"
	dcat:byteSize	"75000000"^^xsd:decimal
total exports	dct:title	"Total AAT statements"
	dct:description	"NTriples zip, file size is approximate"
	dcat:byteSize	"80000000"^^xsd:decimal

(*) Notes about dcat:mediaType, dct:format:

- Some VOID examples use types from <http://mediatypes.appspot.com>, e.g. <http://purl.org/NET/mediatypes/application/n-triples>
- For some reason "application/n-triples" is not in the [IANA Media Types Registry](http://iana.org/media-types), although it's [given in the N-Triples spec](#). Consequently, neither <http://provenanceweb.org/format/mime/application/n-triples> nor <http://purl.org/NET/mediatypes/application/n-triples> are defined.
- We'd like to say void:feature fmt:N-Triples, but this property has domain void:Dataset, while the exports have type dcat:Distribution (object of void:dataDump).

4.5.7 License Info

Unless you are a lawyer, you may not care about licensing info. However, this info is important if you want to ensure you play by the rules, and some software uses it to filter to datasets satisfying certain open data criteria.

subject	property	object
http://vocab.getty.edu/dataset/aat http://vocab.getty.edu/dataset/aat/explicit.zip http://vocab.getty.edu/dataset/aat/full.zip	dct:license, cc:license	http://opendatacommons.org/licenses/by/1.0/
http://vocab.getty.edu/dataset/aat	dct:rights	"Copyright © 2000 The J. Paul Getty Trust. Made available under the ODC Attribution License"
	cc:attributionName	"Contains information from Art & Architecture Thesaurus (AAT)® which is made available under the ODC Attribution License"
	vv:norms	http://www.opendatacommons.org/norms/odc-by-sa/
	vv:declaration	"In circumstances where providing the full attribution statement is not technically feasible, the use of canonical AAT URIs is adequate to satisfy Section 4.3 of the ODC Attribution License"
http://opendatacommons.org/licenses/by/1.0/	dct:type	http://purl.org/adms/licencetype/Attribution
	cc:requires	cc:Attribution

4.5.8 VOID Subsets

As described in [Per-Entity Exports](#), each independent entity (Subject, Source, Contributor) is available in several semantic formats. We express this as void:subsets with uriRegexPattern and the corresponding format:

```
<http://vocab.getty.edu/dataset/aat> void:subset
  <http://vocab.getty.edu/dataset/aat/subjects/rdf>.
<http://vocab.getty.edu/dataset/aat/subjects/rdf>
  dct:title "AAT Subjects as RDF/XML";
  void:uriRegexPattern "^http://vocab.getty.edu/aat/\\d+.rdf$";
  void:feature fmt:RDF_XML.
```

and similarly for:

- Sources and Contributors
- The other formats (fmt:N-Triples, fmt:Turtle, fmt:SPARQL_Results_JSON).

4.5.9 VOID Linksets

We also describe the [AAT to LCSH Alignment](#) following VOID section [5 Describing linksets](#).

```
<http://vocab.getty.edu/dataset/aat/alignment/lcsh> a void:Linkset;
  void:target
    <http://vocab.getty.edu/dataset/aat>,
    [a void:Dataset;
      dct:title "Library of Congress Subject Headings";
      foaf:homepage <http://id.loc.gov/authorities/subjects>;
      void:linkPredicate skos:exactMatch.
    ]
  <http://vocab.getty.edu/dataset/aat> void:subset <http://vocab.getty.edu/dataset/aat/alignment/lcsh>.
```

- The alignment is a void:Linkset that connects two void:targets: AAT and LCSH using skos:exactMatch.
- We were not able to find a URL for the LCSH dataset: <http://id.loc.gov/authorities/subjects> provides a number of Alternate Formats, but these are "data dumps" or "distributions", not datasets
- So we describe it through its foaf:homepage. This is almost as good, as explained in VOID section [2.1 Web page links](#): "As foaf:homepage is an [Inverse Functional Property](#), different descriptions of a dataset provided in different places on the Web can be automatically connected or "smushed" if they use the same homepage URI"
- The linkset is provided only as part of the AAT dataset and not separately (let us know if you want this changed). This is expressed by the last statement. Please note that the void:subset example in section [5.2 Linksets as part of larger datasets](#) uses the wrong direction, see [VOID Issue 105](#)

We also count the void:triples in the linkset, see the end of next section.

4.5.10 Dynamic Descriptive Properties

Some of the properties need to be computed dynamically with every update (regeneration) of the dataset. We insert them in named graph <http://vocab.getty.edu/.well-known/void>:

- Set **dct:modified**, **dct:issued** of dataset, exports, descriptor to the dateTime of regeneration (now()). It's safe to assume that some data is updated on every generation; and publication will happen on the same day

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat>          dct:modified ?date; dct:issued ?date.
  <http://vocab.getty.edu/dataset/aat/explicit.zip>  dct:modified ?date; dct:issued ?date.
  <http://vocab.getty.edu/dataset/aat/full.zip>      dct:modified ?date; dct:issued ?date.
  <http://vocab.getty.edu/.well-known/void>        dct:modified ?date; dct:issued ?date.
}} where {bind (now() as ?date)}
```

- Declare all gvp:Facets as **void:rootResource** of the dataset, To let LOD crawlers find all statements in a top-down fashion. This is in lieu of declaring the [Top Concepts](#) of AAT

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat> void:rootResource ?facet
}} where {?facet a gvp:Facet}
```

The rest provide [VOID statistics](#) (counts) about the dataset. More numbers than you can shake a stick at!

- Count total number of triples

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat> void:triples ?count
}} where {select (count(*) as ?count) {?x ?p ?y}}
```

- Count number of main entities. We consider only gvp:Subject, not the subsidiary entities

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat> void:entities ?count
}} where {select (count(*) as ?count) {?x a gvp:Subject}}
```

- Count number of distinct classes

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat> void:classes ?count
}} where {select (count(*) as ?count)
  {select distinct ?class {?x a ?class. filter(!isBlank(?class))}}}
```

- Count number of distinct properties

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat> void:properties ?count
}} where {select (count(*) as ?count) {select distinct ?prop {?x ?prop ?y}}}
```

- Count number of distinct subjects

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat> void:distinctSubjects ?count
}} where {select (count(*) as ?count) {select distinct ?subj {?subj ?p ?y}}}
```

- Count number of distinct objects

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat> void:distinctObjects ?count
}} where {select (count(*) as ?count) {select distinct ?obj {?x ?p ?obj}}}
```

- Count number of entities per class ([class partition](#)). In contrast to void:entities for the whole dataset, here we consider all entities.

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat> void:classPartition
  [void:class ?class; void:entities ?count]
}} where {select ?class (count(*) as ?count)
  {?x a ?class. filter(!isBlank(?class))} group by ?class}
```

- Count number of triples per property ([property partition](#))

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat> void:propertyPartition
  [void:property ?prop; void:triples ?count]
}} where {select ?prop (count(*) as ?count) {?x ?prop ?y} group by ?prop}
```

- Count number of triples in Linkset (see previous section). We count only one direction ?x skos:exactMatch ?y but not the other direction ?y ?x nor the trivial statements ?x ?x and ?y ?y:

```
insert {graph <http://vocab.getty.edu/.well-known/void> {
  <http://vocab.getty.edu/dataset/aat/alignment/lcsh> void:triples ?count
}} where {select (count(*) as ?count) {?x skos:exactMatch ?y filter(str(?x) < str(?y))}}
```

4.5.11 VOID Deployment

The full descriptive info is available at the following URLs (VOID file in Turtle format):

- <http://vocab.getty.edu/.well-known/void>, following the well-known URI specified in [VOID spec section 7.2](#).
 - As you see above, this URL is used to tie the descriptor to the other entities
 - This URL is also linked from the GVP LOD home page
 - This URL is implemented as a HTTP 302 redirect to the following URL

- <http://vocab.getty.edu/void.ttl>, following the practice in [VOID spec section 6.2](#).

In addition, the descriptive info is available

- In the repository, in its own named graph <http://vocab.getty.edu/.well-known/void>.

This follows the suggestion "query the endpoint itself in case it indexes its own VoID description" in section 2.1 of [SPARQL Web-Querying Infrastructure: Ready for Action?](#) (ISWC 2013) by the creators of the [SPARQL Endpoint Status](#) service. It is also queried by [RKBExplorer VOID storage](#). The following key relations are used to discover the dataset and endpoint:


```
<http://vocab.getty.edu/.well-known/void>
  foaf:primaryTopic <http://vocab.getty.edu/dataset>.
<http://vocab.getty.edu/dataset>
  void:sparqlEndpoint <http://vocab.getty.edu/sparql>.
```

You can query the descriptive info using SPARQL, as shown in [Counting and Descriptive Info](#)

4.5.12 Possible Future Additions

We are considering the following additional descriptive info for future versions, and welcome your suggestions:

- Cover Vocabulary of a Friend (VOAF) metadata and register AAT at the [Linked Open Vocabularies](#) (LOV) site
- Provide [SPARQL 1.1 Service Description](#) of the endpoint, and [tie up the VOID with it](#)
- Provide VOID deployment through the endpoint URL, as per [SPARQL 1.1 Service Description section 2](#)
- void:openSearchDescription to describe the [Full Text Search](#) capabilities, following the [Open Search 1.1 Specification](#)
- We might consider VAEM, VDPP and VOAG, but we are not convinced they have significant penetration

4.6 Export Files

We provide export files (data dumps) in several configurations and formats.

4.6.1 Explicit Exports

These are statements in NTriples format, generated from GVP's database using R2RML.

- First load the required [External Ontologies](#) (SKOS, SKOS-XL, ISO 25964): links are provided in that section
- Then load the [GVP Ontology](#) from <http://vocab.getty.edu/ontology.rdf>
- Then load the export files:
 - <http://vocab.getty.edu/dataset/aat/explicit.zip>: 71 Mb zipped, 1428 Mb unzipped
 - <http://vocab.getty.edu/dataset/tgn/explicit.zip>: **TODO** Mb zipped, Mb unzipped

AAT	TGN
AATOut_1Subjects.nt	TGNOut_1Subjects.nt
AATOut_2Terms.nt	TGNOut_2Terms.nt
AATOut_AssociativeRels.nt	TGNOut_AssociativeRels.nt
AATOut_ContribRels.nt	TGNOut_ContribRels.nt
AATOut_Contribs.nt	TGNOut_Contribs.nt
AATOut_HierarchicalRels.nt	TGNOut_Coordinates.nt
AATOut_Lang_sameAs.nt	TGNOut_HierarchicalRels.nt
AATOut_LCSHAlignment.nt	TGNOut_ObsoleteSubjects.nt
AATOut_Notations.nt	TGNOut_OrderedCollections.nt
AATOut_ObsoleteSubjects.nt	TGNOut_PlaceMap.nt
AATOut_OrderedCollections.nt	TGNOut_PlaceTypes.nt
AATOut_RevisionHistory.nt	TGNOut_RevisionHistory.nt
AATOut_RevisionHistorySource.nt	TGNOut_RevisionHistorySource.nt
AATOut_ScopeNotes.nt	TGNOut_ScopeNotes.nt
AATOut_SemanticLinks.nt	TGNOut_SemanticLinks.nt
AATOut_SourceRels.nt	TGNOut_SourceRels.nt
AATOut_Sources.nt	TGNOut_Sources.nt

- The files inside the zips are named after the different parts of the semantic representation.
- Load them in alphabetical order. OWLIM preserves the order of nodes as they are **first** inserted in the repository, and the first two files are sorted by the required [Sort Order](#)
- Ensure the required [Inference](#) (since these are explicit statements only)

The above is the actual process we use to load the GVP repository (SPARQL endpoint) with fresh data every 2 weeks.

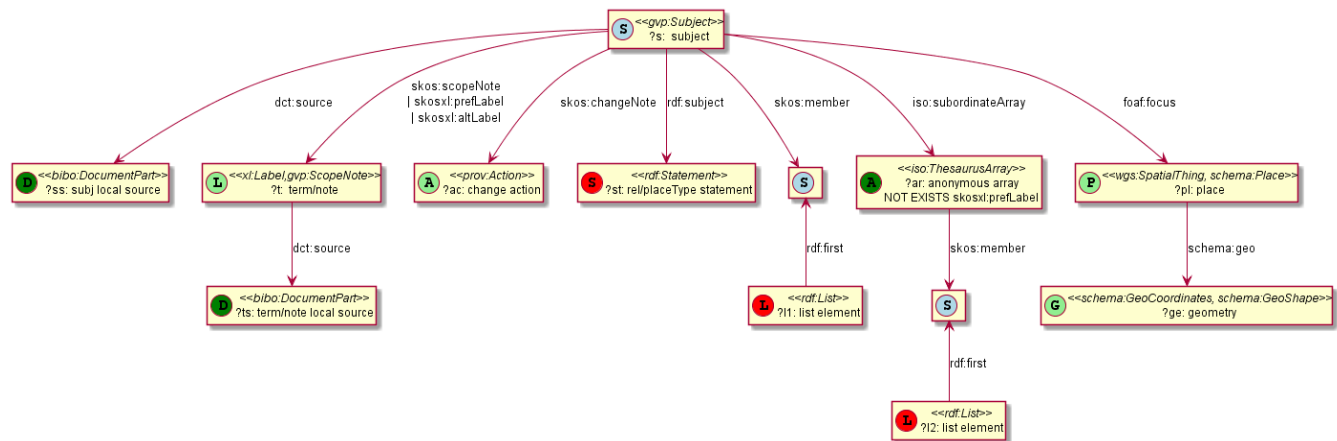
4.6.2 Per-Entity Exports

The Forest page for each resolvable URL provides downloadable semantic representations for that entity in RDF/XML, Turtle, NTriples, JSON formats. The same formats are available through content negotiation, and through direct URLs including file extension, as described in [Semantic Resolution](#).

For the independent entities (Subjects, Sources, Contributors), the semantic formats include all triples (explicit and inferred) of all owned objects. This information is fetched with complex CONSTRUCT queries and cached for better performance. There is no zip with all these files since they are over 80k. Use [Total Exports](#) instead, but if you want such a zip, please contact us.

The [Construct Subject](#) query is shown below and includes:

- All direct triples of the Subject (explicit and inferred) and the other nodes described below
- Local sources (bibo:DocumentPart)
- Terms (skosxl:Label) and scope notes (gvp:ScopeNote) and their local sources
- Change Notes (prov:Activity)
- rdf:Statements describing a relation where the subject plays the role of rdf:subject
- rdf:List nodes of skos:member
- schema:Place and schema:Geo* nodes for TGN subjects that have them



4.6.3 Total Exports

This file includes all statements (explicit and inferred) of all independent entities. It's a concatenation of the [Per-Entity Exports](#) in NTriples format.

Because it includes all required [Inference](#), you can load it to any repository (even one without RDFS reasoning). But it **does not** include gvp:narrowerExtended, skos:narrowerTransitive and skos:semanticRelation because this makes the export files very large (see [All Data For Subject](#)). If you want these properties, you can add them with INSERT queries.

You still want to load the ontologies to get descriptions of properties, associative relations, etc.

- First load the [External Ontologies](#) (SKOS, SKOS-XL, ISO 25964): links are provided in that section
- Then load the [GVP Ontology](#) from <http://vocab.getty.edu/ontology.rdf>
- Then load the export files:
 - <http://vocab.getty.edu/dataset/aat/full.zip>: 120 Mb zipped, 1871 Mb unzipped
 - <http://vocab.getty.edu/dataset/tgn/full.zip>: 340 Mb zipped, 5300 Mb unzipped

AAT	TGN
AATOut_Full.nt : subjects	TGNOut_Full.nt : subjects, places
AATOut_Contribs.nt : contributors	TGNOut_Contribs.nt : contributors
AATOut_Sources.nt : sources	TGNOut_Sources.nt : sources

The first two files are very large and may require some special data loading tool for your repository.

Please note that we have not tried out this process yet. If you encounter any problems, please contact us.

5 Sample Queries

In this section we provide sample queries for various tasks with the AAT LOD. We strive to have the same queries below the SPARQL search box in the [Forest UI](#), but there might be some discrepancies.

If you write an interesting query, please contribute it!

5.1 Finding Subjects

5.1.1 Top-level Subjects

The top-level Subjects of AAT are gvp:Facets, so the query is easy:

```
select * {?f a gvp:Facet; skos:inScheme aat: }
```

The same holds of TGN (there's only two: World and Extraterrestrial Places):

```
select * {?f a gvp:Facet; skos:inScheme tgn: }
```

5.1.2 Descendants of a Given Parent

Let's look for AAT descendants of 300194567 "drinking vessels". This finds "rhyta" and other interesting records.

```
select * {?x gvp:broaderExtended aat:300194567; skos:inScheme aat: }
```

It's a good habit to filter by vocabulary (skos:inScheme) when you want only concepts from that vocabulary. For this particular parent it will make no difference, but

5.1.3 Subjects by Contributor Id

You can easily find subjects contributed by a particular Contributor if you know the id. E.g. GCI is aat_contrib:10000088:

```
select * {
  ?x a gvp:Subject; dct:contributor aat_contrib:10000088}
```

Please note that the different vocabularies use different namespaces for sources and contributors. CGI has a different URL as TGN contributor: tgn_contrib:10000088. See next section for accessing contributions across vocabularies.

5.1.4 Subjects by Contributor Abbrev

If you know the abbreviation of a Contributor but not the id, you can find all contributions across vocabularies (assuming that the same abbreviation was used consistently in all vocabularies), e.g.:

```
select * {
  ?x a gvp:Subject; dct:contributor ?contrib.
  ?contrib foaf:nick "GCI"}
```

If you want to find only contributions to a particular vocabulary, filter by skos:inScheme:

```
select * {
  ?x a gvp:Subject; dct:contributor ?contrib; skos:inScheme aat: .
  ?contrib foaf:nick "GCI"}
```

5.1.5 Preferred Ancestors

Fetch all preferred ancestors of 300226882 "baking dishes" very efficiently (no traversal). We also fetch each parent: this can be used to reconstruct the hierarchy in memory.

```
select * {
  aat:300226882 gvp:broaderPreferredExtended ?parent.
  OPTIONAL {?parent gvp:broaderPreferred ?grandParent}}
```

5.1.6 Full Text Search Query

This is the query used for the [Full Text Search](#).

```
SELECT ?Subject ?Term ?Parents ?ScopeNote ?Type {
  ?Subject luc:term "fishing* AND vessel*"; a ?typ.
  ?typ rdfs:subClassOf gvp:Subject; rdfs:label ?Type.
  optional {?Subject gvp:prefLabelGVP [skosxl:literalForm ?Term]}
  optional {?Subject gvp:parentStringAbbrev ?Parents}
  optional {?Subject skos:scopeNote [dct:language gvp_lang:en; rdf:value ?ScopeNote]}
```

- You should preprocess the query phrase for proper results, as **highlighted above** and described in section [Full Text Search](#).
- If you want to remove punctuation from a user-entered string, it's important to use Unicode properties to decide what is not a letter/digit, lest you nuke Greek, Chinese hieroglyphs, and accented characters (e.g. Spanish, Dutch, Chinese transliterations).
- Use a powerful regex handler and think `\P{L}` and `\P{Nd}`. Don't forget apostrophe.
- If you use this e.g. for an auto-completion application, be kind to our service and wait until the user has typed 3-4 chars and has waited a bit, before firing a query

5.1.7 Find Person Occupations by broaderExtended

A user was searching for "president" occupations by luc:term (FTS) and further restrictions by

- FILTER regex(?parentStringAbbrev, "Agents Facet")
- FILTER langMatches(lang(?label), "en")

But there's a much faster way to restrict to a part of the hierarchy: gvp:broaderExtended. Let's explore the data:

1. Look at e.g. aat:300025470 "presidents" with inference="Explicit and Implicit":
<http://vocab.getty.edu/aat/300025470?inference=all>
2. Notice it has many gvp:broaderExtended:
aat:300024978, aat:300024979, aat:300024980, aat:300025426, aat:300025427, aat:300025432, aat:300264089
3. How to pick the best root? You don't want to explore all them broader one by one.
4. Click on the [Hierarchy](#) tab, which sends you to the Getty site.
(The semantic site does not have a decent hierarchical display yet)
5. aat:300024979 "people (agents)" looks ok.
6. But Guide Term aat:300024980 <Persons by Occupation> is exactly what we need!
7. Use gvp:prefLabelGVP (instead of lang "en") because some concepts may NOT have an "en" prefLabel
8. Don't forget a wildcard in luc:term, or you'll miss "first ladies" = "presidents' wives"
9. Consider NOT ordering by ?label, since luc:term natively returns them in relevance order:
 - By ?label: first ladies, presidents, vice-presidents
 - By Lucene relevance: presidents, vice-presidents, first ladies

The rewritten query is short, nice and efficient:

```
select * {
  ?c a gvp:Concept;
  gvp:broaderExtended aat:300024980 ; # <Persons by Occupation>
  gvp:prefLabelGVP/xl:literalForm ?label ;
  luc:term "president*"}

```

You may also try with "engraver*"

5.1.8 Find Person Occupations by Double FTS

For the query above we had to find out the ID of the guide term <Persons by Occupation> "by hand". But we can use FTS (e.g. "occupation*") to find that subject. If you use that FTS query alone, you'll find many "occupation*" (namely, 82), but if you also search by the desired target term, you find only the same 3 results as above:

```
select * {
  ?concept a gvp:Concept; gvp:broaderExtended ?broader.
  ?concept luc:term "president*".
  ?broader luc:term "occupation*".
  ?concept gvp:prefLabelGVP [xl:literalForm ?concept_label].
  ?broader gvp:prefLabelGVP [xl:literalForm ?broader_label]}
```

Note that we had to use blank nodes "[...]" instead of property paths "/" because of a SPARQL parser bug ([SES-2024](#)). Searching for a target term, plus terms of some of its gvp:broaderExtended ancestors, is a powerful technique that we call "Double FTS"

5.1.9 Find Quartz Timepieces by Double FTS

For another example of "Double FTS", let's find Quartz Timepieces:

```
select * {
  ?concept a gvp:Concept; gvp:broaderExtended ?broader.
  ?concept luc:term "quartz*".
  ?broader luc:term "time*".
  ?concept gvp:prefLabelGVP [xl:literalForm ?concept_label].
  ?broader gvp:prefLabelGVP [xl:literalForm ?broader_label]}
```

There's only one matching ?concept ([aat:300225923](#) "quartz clocks"), which however is returned twice because it has two ?broader ("timepieces" vs <timepieces by form>). If you want it just once, try "select distinct ?concept ?concept_label".

5.1.10 Find Subject by Exact English PrefLabel

The FTS queries above are perfect for auto-completion services, i.e. for returning many potential matches when the user enters a keyword (perhaps partial). But if you are working on co-referencing (e.g. with OpenRefine reconciliation), you may need something simpler: match by exact label:

```
select * {?subj gvp:prefLabelGVP/xl:literalForm "rhyta"@en}
```

Note that prefLabelGVP is usually in the plural!

For AAT, it's important to specify the language tag, else the query will return nothing (for TGN, 80% of the terms don't have a language). Most prefLabelGVP are in English (but not all, see next section)

5.1.11 Find Subject by Language-Independent PrefLabels

Most AAT prefLabelGVP are in English but not all. In AAT, 5% are in a different language:

```
select * {?subj gvp:prefLabelGVP/xl:literalForm ?label. filter(lang(?label) != "en")}
```

Of these exceptions, almost all are in en-us "American English" (curiously, even the acronym "[IEEE 802.11](#)"@en-us is marked so). And there's one in Spanish: [aat:300181273](#) "troffers"@es (a kind of recessed lamps).

In TGN, 80% don't have a language. For this reason, you may prefer to search by string without providing the language tag:

```
select distinct ?subj {?subj skos:prefLabel ?label. filter(str(?label)="rhyta")}
```

- We use str() to strip off the lang tag
- We search for all prefLabels (not just prefLabelGVP)
- We use the shortcut skos:prefLabel instead of having to go through xl:prefLabel/xl:literalForm
- We have to add "distinct" because a subject may have the same string in several languages, e.g. "rhyta"@en and "rhyta"@el-Latn (the latter is Greek transliterated to Latin)

Important: This query is quite inefficient, since there is no functional index for str(...). Use exact literal (previous section) or FTS ([Full Text Search Query](#)).

5.1.12 Find Subject by Any Label

Since GVP prefers the Plural form as Descriptor (prefLabel), the previous query finds nothing for "rhyton".

To widen the search to cover all labels, we add skos:altLabel to the query. We use SPARQL 1.1 Property Path notation "|":

```
select distinct ?subj {?subj skos:prefLabel|skos:altLabel ?label. filter(str(?label)="rhyton")}
```

5.1.13 Find Terms by Language Tag

Say you want to find all Berber terms transliterated to Latin (no? I do this every day!):

```
select * {
  ?subj xl:prefLabel|xl:altLabel [xl:literalForm ?term]
  filter(langMatches(lang(?term),"ber-Latn"))}
```

- Here we use query features we've seen before: a property path alternative "|", and a blank node since we don't care about the xl:Label but only about its literal form.
- We also use a new feature: langMatches(). It lets us find not only terms in Berber (e.g. "Zemmour"@ber-latn), but also in Berber Dialects (e.g. "Ait Youssi"@ber-latn-x-dialect).

You could try to use substr(lang()) or strstarts(lang()), but the comparison must be case-insensitive (see [Language Tag Case](#)). langMatches() is easier to use and is dedicated to this purpose.

5.1.14 Find Ordered Subjects

Find AAT subjects whose children have "forced" (explicit) ordering.

- All subjects have gvp:displayOrder, so we check for non-trivial one (>1)
- We get the label using an anonymous node notation "[.]"

```
select ?coll ?label {
  ?coll gvp:prefLabelGVP [skosxl:literalForm ?label]; skos:inScheme aat: .
  filter(exists {?coll gvp:narrower [gvp:displayOrder ?order] filter(?order>1)})}
```

5.1.15 Find Ordered Hierarchies

In this section we do something similar as the previous one, but illustrate some different approaches:

- We look for hierarchies (gvp:Hierarchy) that are ordered by explicit type (skos:OrderedCollection). The semantic representation uses skos:OrderedCollection for exactly those subjects whose children have non-trivial "forced" ordering
- We get the label using SPARQL Property Path syntax "/"
- We use the shorter prefix "xl:" instead of "skosxl:": both of these prefixes are defined in the repository, and mean the same thing.

```
select * {?x a gvp:Hierarchy, skos:OrderedCollection; gvp:prefLabelGVP/xl:literalForm ?label}
```

5.1.16 Get Subjects in Order

Some subjects e.g. 300018774 "Siberian periods" have children laid out in a particular order:

```
select * {
  aat:300018774 gvp:narrower ?x.
  optional {?x gvp:displayOrder ?ord}
  ?x gvp:prefLabelGVP [xl:literalForm ?l]
} order by ?ord
```

Because OWLIM preserves the order in which resources are first inserted (see [Sort Order](#) item 2), subjects and terms are returned in the desired order, even if you don't use the custom field gvp:displayOrder in your query:

```
select * {
  aat:300018774 gvp:narrower ?x.
  ?x gvp:prefLabelGVP [xl:literalForm ?l]}
```

5.1.17 Find Contributors by Vocabulary

To get subjects that belong to a specific vocabulary, we can use skos:inScheme:

```
select * {?subject skos:inScheme aat:}
```

Sources and contributors are also split by vocabulary, but don't have a property to link them to that vocabulary. So we have to use [Named Graphs](#):

```
select * {graph <http://vocab.getty.edu/dataset/aat>
  {?contrib a foaf:Agent; foaf:nick ?abbrev}}
```

5.1.18 Find Sources by Vocabulary

Let's find all TGN sources. We again use [Named Graphs](#):

```
select * {graph <http://vocab.getty.edu/dataset/tgn>
  {?source a bibo:Document; bibo:shortTitle ?name}}
```

If you read [Number of Entities \(Dynamic\)](#), you may wonder how come we didn't have to filter out [Local Sources](#), which have explicit type `bibo:DocumentPart` and inferred type `bibo:Document`. The answer is that OWLIM puts all inferred statements in the empty graph (a peculiarity of OWLIM), so the named graph above doesn't include the inferred type statements.

5.2 Getting Information

5.2.1 Subject Preferred Label

For each of the Find queries, you can get the preferred label in addition to the URI by adding this fragment:

```
?x gvp:prefLabelGVP [skosxl:literalForm ?label]
```

- Here we reach to the `skosxl:Label` preferred by GVP (each Subject has exactly one). Since we don't care about the `skosxl:Label` node (only about the label stored there), we use a blank node in the query.

Eg when we add this fragment to "Subjects by Contributor Abbrev", we get this query:

```
select * {
  ?x a gvp:Subject; dct:contributor ?contrib;
    gvp:prefLabelGVP [skosxl:literalForm ?label].
  ?contrib foaf:nick "GCI"}
```

5.2.2 All Data for Terms of Subject

The following query returns all data about the [Terms](#) of a given subject. It can be used to make displays as on Getty's site, e.g. see [tgn:7001393](#) "Athens" or [aat:300000590](#) "miniature golf courses".

```
select ?l ?lab ?lang ?pref ?historic ?display ?pos ?type ?kind ?flag ?start ?end ?comment {
  values ?s {tgn:7001393}
  values ?pred {xl:prefLabel xl:altLabel}
  ?s ?pred ?l.
  bind (if(exists{?s gvp:prefLabelGVP ?l},"pref GVP",if(?pred=xl:prefLabel,"pref","")) as ?pref)
  ?l xl:literalForm ?lab.
  optional {?l dct:language [gvp:prefLabelGVP [xl:literalForm ?lang]]}
  optional {?l gvp:displayOrder ?ord}
  optional {?l gvp:historicFlag [skos:prefLabel ?historic]}
  optional {?l gvp:termDisplay [skos:prefLabel ?display]}
  optional {?l gvp:termPOS [skos:prefLabel ?pos]}
  optional {?l gvp:termType [skos:prefLabel ?type]}
  optional {?l gvp:termKind [skos:prefLabel ?kind]}
  optional {?l gvp:termFlag [skos:prefLabel ?flag]}
  optional {?l schema:startDate ?start}
  optional {?l schema:endDate ?end}
  optional {?l rdfs:comment ?comment}
} order by ?ord
```

5.2.3 Preferred and Vernacular Terms

Find concepts having alternate labels in the Vernacular, and list them together with the preferred label.

(Please note that there are some preferred labels that are also in the Vernacular.)

```
select ?c ?preferred ?vernacular {
  ?c xl:altLabel ?t; gvp:prefLabelGVP/xl:literalForm ?preferred.
  ?t gvp:termFlag <http://vocab.getty.edu/term/flag/Vernacular>; xl:literalForm ?vernacular}
```

5.2.4 Scientific Names by Language

"Scientific name" is the zoology/botany taxon name (genus, species, etc) and comes from Latin. However, all such terms are also emitted in English. I wondered what is the distribution of scientific names across languages:


```
select ?lang (count(*) as ?c) {
  ?t gvp:termKind <http://vocab.getty.edu/term/kind/ScientificOrTechnical>; dct:language ?l.
  ?l gvp:prefLabelGVP/xl:literalForm ?lang
} group by ?lang
```

The result for AAT is:

lang	c
English (language)@en	1403
Latin (language)@en	1351
Spanish (language)@en	63
Italian (language)@en	1
French (language)@en	1

5.2.5 Scientific Names not in English and Latin

Then I wondered which are the odd scientific names (neither English nor Latin):

```
select ?c ?lab {
  ?c xl:prefLabel|xl:altLabel ?t.
  ?t gvp:termKind <http://vocab.getty.edu/term/kind/ScientificOrTechnical>;
    xl:literalForm ?lab.
  filter (lang(?lab) not in ("en", "la"))}
```

5.2.6 All Data For Subject

This complex query is used to get all info about a [Subject](#) and its owned sub-objects (see [Per-Entity Exports](#)):

```
CONSTRUCT {
  ?s ?p1 ?o1. # subject
  ?ac ?p2 ?o2. # change action
  ?t ?p3 ?o3. # term/note
  ?ss ?p4 ?o4. # subject local source
  ?ts ?p6 ?o6. # term/note local source
  ?st ?p7 ?o7. # statement about relations/placeTypes
  ?ar ?p8 ?o8. # anonymous array of subject
  ?l1 ?p9 ?o9. # list element of subject
  ?l2 ?pA ?oA. # list element of anonymous array
  ?p1 ?pB ?oB. # place
  ?ge ?pC ?oC. # geometry
} WHERE {
  BIND (tgn:1000193 as ?s)
  {?s ?p1 ?o1 FILTER(!isBlank(?o1) &&
    !(?p1 in (gvp:narrowerExtended, skos:narrowerTransitive, skos:semanticRelation)))}
  UNION {?s skos:changeNote ?ac. ?ac ?p2 ?o2}
  UNION {?s dct:source ?ss. ?ss a bibo:DocumentPart. ?ss ?p4 ?o4}
  UNION {?s skos:scopeNote|skosxl:prefLabel|skosxl:altLabel ?t.
    {?t ?p3 ?o3 FILTER(!isBlank(?o3))}
    UNION {?t dct:source ?ts. ?ts a bibo:DocumentPart. ?ts ?p6 ?o6}}
  UNION {?st rdf:subject ?s. ?st ?p7 ?o7}
  UNION {?s skos:member/^rdf:first ?l1. ?l1 ?p9 ?o9}
  UNION {?s iso:subordinateArray ?ar FILTER NOT EXISTS {?ar skosxl:prefLabel ?t1}.
    {?ar ?p8 ?o8}
    UNION {?ar skos:member/^rdf:first ?l2. ?l2 ?pA ?oA}}
  UNION {?s foaf:focus ?p1.
    {?p1 ?pB ?oB}
    UNION {?p1 schema:geo ?ge. ?ge ?pC ?oC}}
```

Blank nodes (owl:Restriction types) are possible for Subject, Term and Note (?o1 and ?o3), so we filter them out.

We also filter out downward-looking properties (gvp:narrowerExtended, skos:narrowerTransitive) and skos:semanticRelation, otherwise these cause very large [Per-Entity Exports](#) for subjects high-up in the hierarchies. Consider that aat:300386699 "facets (controlled vocabulary)" > World > almost all of TGN.

5.2.7 Historic Information on Relations

Here is an example query to fetch all relations of aat:300020271, together with optional [Historic Information](#).

```
SELECT ?concept1 ?rel ?concept2 ?startDate ?endDate ?comment ?hist {
  ?concept1 ?rel ?concept2. FILTER(?concept1=aat:300020271 || ?concept2=aat:300020271)
  ?concept1 a gvp:Subject. ?concept2 a gvp:Subject.
  OPTIONAL {
    ?statement rdf:subject ?concept1; rdf:predicate ?rel; rdf:object ?concept2.
    OPTIONAL {?statement schema:startDate ?startDate}.
    OPTIONAL {?statement schema:endDate ?endDate}.
    OPTIONAL {?statement rdfs:comment ?comment}.
    OPTIONAL {?statement gvp:historicFlag ?hist}}}
```

We show some of the results below. Extensive [SKOS Inference](#) is involved, so we omit a lot of the inferences). Please note that the historic info is available only on the explicitly stated relation (e.g. gvp:aat2812_followed), not on inferred relations (e.g. skos:related)

concept1	Rel	concept2	startDate	endDate	comment	hist
aat:300020271	gvp:aat2812_followed	aat:300020269	-2785	-2765	Second Dynasty began ca. 2775 BCE	-
aat:300020269	gvp:aat2811_preceded	aat:300020271	-2785	-2765	Second Dynasty began ca. 2775 BCE	-
aat:300020271	gvp:broaderPreferred	aat:300020265	-	-	-	-
aat:300020271	skos:broader	aat:300020265	-	-	-	-

5.2.8 Historic Information of Terms

Fetch all terms with **some** [Historic Information](#), together with that information:

```
select * {
  ?term a xl:Label; xl:literalForm ?literal.
  optional {?term gvp:historicFlag [skos:prefLabel ?historic]}
  optional {?term schema:startDate ?start}
  optional {?term schema:endDate ?end}
  optional {?term rdfs:comment ?comment}
  filter (bound(?historic) || bound(?start) || bound(?end) || bound(?comment))}
```

The filter picks only terms that have at least one of the Historic Information properties.

The way the GVP database is structured, terms for the same concept and having the same xl:literalForm (differing only by language) are managed as one and carry the same historic info. You'll see such duplication, e.g. for "opuscules"@en and "opuscules"@fr. We can eliminate the duplicates like this (from 15k down to 4k across AAT and TGN):

```
select ?term ?literal ?historic ?start ?end ?comment {
  ?term a xl:Label; xl:literalForm ?literal.
  optional {?term gvp:historicFlag [skos:prefLabel ?historic]}
  optional {?term schema:startDate ?start}
  optional {?term schema:endDate ?end}
  optional {?term rdfs:comment ?comment}
  filter (bound(?historic) || bound(?start) || bound(?end) || bound(?comment))
  filter not exists {
    ?c xl:prefLabel|xl:altLabel ?term, ?term1.
    ?term1 xl:literalForm ?literal1
    filter (str(?literal1) = str(?literal) && ?term1 < ?term)}}}
```

Some explanations about the last filter:

- We look for another ?term1 of the same concept ?c, having the same ?literal1 as ?term
- If ?term1 has a "smaller" URL string then "filter not exists" rejects the ?term

As a result, we pick only one of the terms having the same literal; the criterion "least URL" is arbitrary but fast

5.2.9 Preferred Terms for Contributors

Terms that are preferred for specific contributors. We want only the labels, so we use blank nodes [...] to disregard the URIs. You can similarly use gvp:contributorNonPreferred and gvp:contributorAlternatePreferred.

```
select * {[xl:literalForm ?term] gvp:contributorPreferred [foaf:nick ?contrib]}
```

Here we use blank nodes both before and after the property:

- The first blank node ignores the term's URL and returns only the term
- The second blank node ignores the contributor's URL and returns only the contributor abbreviation

This returns a mix of terms (labels) from all vocabularies. To filter by vocabulary, use something like this:

```
select * {
  [] skos:inScheme aat: ;
  xl:prefLabel|xl:altLabel [
    xl:literalForm ?term;
    gvp:contributorPreferred [foaf:nick ?contrib]
  ]
}
```

We use the SPARQL 1.1 Property Paths notation "|" and a lot of blank nodes. Are you confused by all these blank nodes yet? Let's rewrite by using explicit variables, and a subset of them in the select:

```
select ?term ?contrib {
  ?concept skos:inScheme aat: ;
  xl:prefLabel|xl:altLabel ?t.
  ?t xl:literalForm ?term;
  gvp:contributorPreferred ?c.
  ?c foaf:nick ?contrib}
}
```

This is much more understandable, though some might find it less elegant.

5.2.10 Preferred Terms for Sources

Terms that are preferred for specific sources. We want only the labels, so we use blank nodes [...] to disregard the URIs. You can similarly use gvp:sourceNonPreferred and gvp:sourceAlternatePreferred.

```
select * {[xl:literalForm ?term] gvp:sourcePreferred [bibo:shortTitle ?source]}
```

To filter by vocabulary, we can use this:

```
select ?term ?source {
  ?concept skos:inScheme aat: ;
  xl:prefLabel|xl:altLabel ?t.
  ?t xl:literalForm ?term;
  gvp:sourcePreferred ?s.
  ?s bibo:shortTitle ?source}
}
```

5.2.11 Concepts Related by Particular Associative Relation

While investigating the precise meaning of the [Associative Relationship](#) 2100 "distinguished from", we tried this query. It retrieves related concepts, their labels and scope notes (in English).

- We use blank nodes "[...]" instead of property paths "/" because of a SPARQL parser bug ([SES-2024](#))
- We filter by the string representation of the two concept URIs because this associative relation is symmetric, and we don't want to get it both forward and inverse in the result set.

```
select * {
  ?c1 gvp:aat2100_distinguished_from ?c2. filter (str(?c1) < str(?c2))
  ?c1 gvp:prefLabelGVP [xl:literalForm ?l1];
    skos:scopeNote [rdf:value ?n1; dct:language gvp_lang:en].
  ?c2 gvp:prefLabelGVP [xl:literalForm ?l2];
    skos:scopeNote [rdf:value ?n2; dct:language gvp_lang:en]}
```

We don't need to filter by skos:inScheme because gvp:aat2100_distinguished_from can link only AAT concepts, nothing else.

5.2.12 Languages and ISO Codes

We use the Guide Term [300389738](#) <languages and writing systems by specific example> and the two specific sources described at [Language Tags and Sources](#) to return all languages together with their ISO2 and ISO3 language codes (where assigned):

```
select ?lang ?name ?iso2 ?iso3 {
  ?lang gvp:broader aat:300389738; gvp:prefLabelGVP/skosxl:literalForm ?name.
  optional {?lang skosxl:altLabel [skosxl:literalForm ?iso2; dct:source aat_source:2000075479]}
  optional {?lang skosxl:altLabel [skosxl:literalForm ?iso3; dct:source aat_source:2000075493]}
```

Some observations on the ISO codes:

- Only the "more important" languages have alpha-2 codes (e.g. Abkhaz has one, but Abaza doesn't)
- Out of 1883 languages, 1330 (70%) have an alpha-3 code. The rest are more exotic languages, variants (e.g. transliterated; liturgical; medieval), or modifications (e.g. GVP uses Frisian. ISO has defined codes for its predecessor Old Frisian and its dialects West, Saterland and North Frisian, but not for Frisian itself).

5.2.13 Language URLs

Find all "logical" language URLs ([Language Dual URLs](#)):

```
select * {?x owl:sameAs ?y FILTER(str(?x) > str(?y))} order by str(?x)
```

You **must** check "Expand results over equivalent URIs" in the SPARQL UI. Or use this [direct query link](#).

5.3 TGN-Specific Queries

5.3.1 Places by Type

Remember that place types are AAT concepts. To find places by type, we could locate the needed AAT concept and use it. But it's easier to use the label of that concept; remember that you have to specify the language! E.g. looking for "republics", we find 180:

```
select * {
  ?c gvp:prefLabelGVP [xl:literalForm ?lab];
    gvp:placeType [rdfs:label "republics"@en]}
```

Because AAT provides labels in plural (skos:prefLabel) and singular (skos:altLabel) and rdfs:label includes both, we can get away with being a little less precise and providing the type name in singular (same results):

```
select * {
  ?c gvp:prefLabelGVP [xl:literalForm ?lab];
    gvp:placeType [rdfs:label "republic"@en]}
```

5.3.2 Places, with English or GVP Label

For AAT we usually return the label preferred by GVP (gvp:prefLabelGVP/xl:literalForm, which is most often in English). Let's try the same for TGN (e.g. looking for "republics"), we get 180 results:

```
select * {
  ?c gvp:prefLabelGVP [xl:literalForm ?lab];
    gvp:placeType [rdfs:label "republics"@en]}
```

In TGN, **prefLabelGVP is in the Vernacular** language (e.g. "Afghānestān"@prs-latn, that's Pashtu transliterated to Latin). You may prefer the English label. Let's try this, but we get only 151 results:

```
select * {
  ?c xl:prefLabel [xl:literalForm ?lab; dct:language gvp_lang:en];
  gvp:placeType [rdfs:label "republics"@en]}
```

The reason is that **not all** TGN places **have an English label**. We can use the `coalesce()` function to pick the English label if present, else the `prefLabelGVP`:

```
select ?c (coalesce(?labEn,?labGVP) as ?lab) {
  ?c gvp:placeType [rdfs:label "republics"@en]
  optional {?c xl:prefLabel [xl:literalForm ?labEn; dct:language gvp_lang:en]}
  optional {?c gvp:prefLabelGVP [xl:literalForm ?labGVP]}
```

5.3.3 Places by Direct and Hierarchical Type

Let's try to find all countries. But what exactly is a country? AAT has a number of related concepts: dig a bit in the hierarchy of [300232420](#) "sovereign states". The scope notes can explain the difference:

- [300387506](#) "countries (sovereign states)": independent states, or regions once independent and still distinct in race, language, institutions, or historical memories.
Example: "England, Scotland, and Northern Ireland are countries in the nation of the United Kingdom"
- [300128207](#) "nations": Sovereign states typically originating with a large group of people associated with a particular territory and possessing distinct ethnic, historical, or cultural characteristics
- [300232420](#) "sovereign states": Political units, such as nations, that exercise and are recognized internationally as possessing sovereignty.
Includes city-states like the Vatican and empires like the Roman Empire, the Russian Empire and the Kievan Rus

So "countries" is too narrow. Let's first search for `placeTypePreferred` "nations", we get 195:

```
select * {
  ?c gvp:prefLabelGVP [xl:literalForm ?lab];
  gvp:placeTypePreferred [rdfs:label "nations"@en]}
```

Here we're looking for an English `xl:prefLabel`; `gvp:prefLabelGVP` is usually in the.

There are some "nations" having that in `placeTypeNonPreferred` (e.g. Armenia, Czechoslovakia, Cyprus, Soviet Union), 16:

```
select * {
  ?c gvp:prefLabelGVP [xl:literalForm ?lab];
  gvp:placeTypeNonPreferred [rdfs:label "nations"@en]}
```

We get both by searching in `placeType` (a generalization of `placeTypePreferred` and `placeTypeNonPreferred`), 211:

```
select * {
  ?c gvp:prefLabelGVP [xl:literalForm ?lab];
  gvp:placeType [rdfs:label "nations"@en]}
```

We get more results if we search for either "direct type" or "hierarchical type" (using SPARQL 1.1 Property Paths), 255:

```
select distinct * {
  ?c gvp:prefLabelGVP [xl:literalForm ?lab];
  gvp:placeType|(gvp:placeType/gvp:broaderGenericExtended) [rdfs:label "nations"@en]}
```

This includes sub-types of "nations", such as republics, island nations, etc.

Did you notice the "select **distinct**"? Without it you'll get 413 results, which includes 158 duplicates. The reason is that **some but not all** TGN records have both "nations" and a sub-type thereof as direct `placeTypes`. For example:

- `tgn:1000046` Bolivia has direct type: "republics" and its super-type "nations", but
- `tgn:7024573` Greenland has direct type "island nations" without its super-type "nations"

You can find all TGN records with a sub-type of "nations" but not "nations" like this, 44:

```
select * {
  ?c gvp:prefLabelGVP [xl:literalForm ?lab];
  gvp:placeType/gvp:broaderGenericExtended [rdfs:label "nations"@en].
  filter not exists {?c gvp:placeType [rdfs:label "nations"@en]}}
```

Finally, we get most results if we search for the hierarchical type "sovereign states", which is a super-type of "nations", 320.

```
select distinct * {
  ?c gvp:prefLabelGVP [xl:literalForm ?lab];
  gvp:placeType|(gvp:placeType/gvp:broaderGenericExtended) [rdfs:label "sovereign states"@en]}
```

You can get 3 lessons from this section:

- To find places by type, find the most general applicable type in the AAT hierarchy (this takes some browsing and data examination)
- Use gvp:placeType, which includes both placeTypePreferred (usually Current) and placeTypeNonPreferred (usually Historic)
- Search for either direct or hierarchical type: gvp:placeType|(gvp:placeType/gvp:broaderGenericExtended)

5.3.4 Breakdown of Sovereign States by Type

Given the series of increasing numbers in the previous section, let's see the breakdown of "sovereign states" by placeTypePreferred:

```
select ?type (count (distinct ?x) as ?c) {
  ?x gvp:placeType|(gvp:placeType/gvp:broaderGenericExtended) [rdfs:label "sovereign states"@en];
  gvp:placeTypePreferred [gvp:prefLabelGVP [xl:literalForm ?type]]
} group by ?type order by desc(?c)
```

The result is:

type	c
nations@en	195
former primary political entities@en	39
republics@en	24
historical regions@en	23
inhabited places@en	11
countries (soveriegn states)@en	7
general regions@en	6
former administrative divisions@en	2
deserted settlements@en	2
states (political divisions)@en	2
dependent states@en	2
regions (administrative divisions)@en	2

5.3.5 Inhabited Places That Were Sovereign States

Something interesting in the previous table: there are 11 inhabited places (cities/villages) that used to be sovereign states. Let's find them, and also output the parentString:

```
select * {
  ?c gvp:prefLabelGVP [xl:literalForm ?lab]; gvp:parentString ?parents;
  gvp:placeType|(gvp:placeType/gvp:broaderGenericExtended) [rdfs:label "sovereign states"@en];
  gvp:placeTypePreferred [rdfs:label "inhabited places"@en]}
```

5.3.6 Places by Type and Parent Place

Often we need to search for places of certain kind, within a certain locality (parent place).

- For type, we use all lessons from [Places by Direct and Hierarchical Type](#), and therefore the last query form in that section.
- For parent place, we use gvp:broaderPartitiveExtended, which finds all descendants of a given place. We also name the parent place by (any one of its) label(s). This works only if the place label is unique, which is true for the examples below.

Let's find archaeological sites in Egypt; there are 66:

```
select distinct * {
  ?place skos:inScheme tgn: ;
  gvp:placeType| (gvp:placeType/gvp:broaderGenericExtended) [rdfs:label "archaeological sites"@en];
  gvp:broaderPartitiveExtended [rdfs:label "Egypt"@en];
  gvp:prefLabelGVP [xl:literalForm ?name];
  gvp:parentString ?parents}
```

5.3.7 Places by Type, with placeTypePreferred

Let's find all "[riverine bodies of water](#)" in Germany (this includes rivers, streams, brooks, watercourses, waterfalls, etc). There are 7424, of which 301 are rivers. You may also want to print the placeTypePreferred, to distinguish between Aar (river) and Aar (stream), both in Hesse, Germany:

```
select distinct * {
  ?place skos:inScheme tgn: ;
  gvp:placeTypePreferred [gvp:prefLabelGVP [xl:literalForm ?type]];
  gvp:placeType| (gvp:placeType/gvp:broaderGenericExtended) [rdfs:label "riverine bodies of water"@en];
  gvp:broaderPartitiveExtended [rdfs:label "Germany"@en];
  gvp:prefLabelGVP [xl:literalForm ?name];
  gvp:parentString ?parents}
```

Notes:

- If you forget the "distinct", you'll get 7673 results, of which 249 duplicates. The likely reason is that these 249 places have two types, both sub-types of "riverine bodies of water" ([Places by Direct and Hierarchical Type](#) includes a similar analysis regarding "nations" and sub-types thereof).
- This query is inspired by the [sample queries at this XML-based service](#).

5.3.8 Places by Triple FTS

Place names offer great ambiguity, since when people migrate they often carry old place names to new places; and some places or features are named after others. Just try a couple of searches on the [Getty TGN Site](#) to be convinced:

- London: finds 142 places, amongst them:
 - About 20 boroughs of Greater London, since their Official names are "London Borough of ..."
 - About 20 places in the US (3 in Ohio alone!), Canada, Jamaica, etc
 - Hydrographic features, such as 10 creeks, branches, ditches, outflow canals, the Bay of London, the London Basin
 - London Bridge: there are a railway station, a ridge, a stream and a creek named that way!
 - Green Park, since one of its labels has a qualifier "London" to disambiguate it from other such parks (how ironic this increases the ambiguity for the name "London"!)
 - London Road Station, located in Manchester
- San Francisco: finds 683 places in the US, Mexico, Spain (including Tenerife and Majorka), Chile, even Antarctica
 - Even if you limit to the United States, there are 65 matches
 - Even if you use the nickname Frisco and limit to the United States, there are 61 matches
 - So it's quite hard to find **the** San Francisco, California

Therefore mechanisms to add specificity to TGN searches are very desirable. This section describes such a mechanism.

TGN places form a containment hierarchy (gvp:broaderPartitiveExtended) and have types with a useful hierarchy (gvp:placeType/gvp:broaderGenericExtended, see [TGN Place Types](#)). Extending the approach from [Find Person Occupations by Double FTS](#), we get powerful ways to find places by using any combination of words from the labels of:

- The place: we limit to skos:inScheme tgn:
- Its types and super-types: we limit to skos:inScheme aat:
- Its parent places: we limit to skos:inScheme tgn:

- Unlike searching for AAT concepts, we **don't** use wildcards in luc:term, because there are many useful short words that increase specificity (e.g. "CA" is the USPS code for California and an ISO code for Canada).

For example the following query is looking for places called "Athos" having type "religious" (center), located in "Greece".

```
select distinct * {
  ?p   skos:inScheme tgn;; luc:term "athos";      gvp:prefLabelGVP [xl:literalForm ?pLab].
  ?t1  skos:inScheme aat;; luc:term "religious";  gvp:prefLabelGVP [xl:literalForm ?t1Lab].
  ?pp1 skos:inScheme tgn;; luc:term "greece";     gvp:prefLabelGVP [xl:literalForm ?ppLab].
  ?p   gvp:placeType|(gvp:placeType/gvp:broaderGenericExtended) ?t1.
  ?p   gvp:broaderPartitiveExtended ?pp1.
}
```

This will find Mount Athos, which is a religious center, inhabited place and mountain in Greece.

To try the searches in the following tables, you need to edit the query in an obvious way.

There are many other ways to find Mount Athos:

p	t1	t2	pp	Result
Athos	religious		Greece	Mount Athos
mount	religious			Mount Athos (Greece), Mount Aracadi (Greece), Mount Zion (Israel), Mount Grace Priory (England), Convent of Mount Sinai (Egypt)
	religious	mountain		Mount Athos. Despite their names, all the others are not actually mountains
Athos	religious			Mount Athos. If you don't specify "religious" you'll get a bunch of administrative regions
	religious		China	Religious centers in China (e.g. Lhasa in Tibet)
	Inca			Centers related to the Inca culture (e.g. Machupicchu): 9
	under*			Underwater sites, undersea features; but also underground stations

You need to be mindful of performance for those queries, especially if you don't specify a search keyword for ?p, because then OWLIM needs to look for gvp:broader*Extended of all combinations of the other variables, and these can be a lot.

Limiting to skos:inScheme aat:, there are 50 ?t1 matching "religious" and 30 ?t2 matching "mountain". That's 150 combinations, and looking for all their gvp:broader*Extended is already a lot of work.

If you don't limit ?t2 to skos:inScheme aat:, there are 27k TGN concepts matching "mountain". Together with ?t1, that makes 1.3M combinations, and looking for all their gvp:broader*Extended takes forever.

Furthermore, If you want to check for two types ?t1 and ?t2, you may have to use the direct type only and not the hierarchical type, since that slows down the query too much.

5.3.9 Places by FTS Parents

We modify the query to include two parent places in sequence.

p	t1	pp1	pp2	Result
Sofia		Mexico		Sofia, New Mexico, US
	ranch	Mexico		Ranches in Mexico: 4762
Sofia		Bulgaria		Sofia (city) and Sofiya-Grad (administrative division)
San Francisco		CA	US	San Francisco, California. We use the USPS code of California
Frisco		CA	US	San Francisco, California. We also use its nickname

E.g. for the last row:

```
select * {
  ?p skos:inScheme tgn;; luc:term "frisco"; gvp:prefLabelGVP [xl:literalForm ?pLab].
  ?pp1 skos:inScheme tgn;; luc:term "CA"; gvp:prefLabelGVP [xl:literalForm ?pp1Lab].
  ?pp2 skos:inScheme tgn;; luc:term "US"; gvp:prefLabelGVP [xl:literalForm ?pp2Lab].
  ?p gvp:broaderPartitiveExtended ?pp1.
  ?pp1 gvp:broaderPartitiveExtended ?pp2}
```

These searches are considerably more powerful than the Getty website search:

- It searches for words that are parts of the corresponding label
- It allows hierarchical type search (direct type or its super-types)
- It allows conjunctive type or super-place search (e.g. "religious" and "mountain" together)
- It allows any super-places, not just nations

5.3.10 Capitals by Type

TGN includes several place types that indicate capitals: national capitals, state capitals, etc. Following the approach from the previous section, we can use these to find quickly the capitals of various entities, e.g.:

t	pp1	pp2	Result
capital	Bulgaria		Sofia (city), the national capital of Bulgaria.
capital	CA	US	Sacramento, the state capital of California. We use the USPS code of California

```
select distinct * {
  ?p skos:inScheme tgn;; gvp:prefLabelGVP [xl:literalForm ?pLab].
  ?t skos:inScheme aat;; luc:term "capital"; gvp:prefLabelGVP [xl:literalForm ?tLab].
  ?pp1 skos:inScheme tgn;; luc:term "CA"; gvp:prefLabelGVP [xl:literalForm ?pp1Lab].
  ?pp2 skos:inScheme tgn;; luc:term "US"; gvp:prefLabelGVP [xl:literalForm ?pp2Lab].
  ?p gvp:placeType|(gvp:placeType/gvp:broaderGenericExtended) ?t.
  ?p gvp:broaderPartitiveExtended ?pp1.
  ?pp1 gvp:broaderPartitiveExtended ?pp2}
```

5.3.11 Capitals by Association

Cities usually are not immediate children of the entity they are capital of, e.g.:

- above Sofia (city) is Sofiya-grad (administrative region),
- above Sacramento (city) is Sacramento (county).

TGN includes an associative relation that connects capitals to entities, this finds 352:

```
select * {
  ?cap gvp:tgn3201_capital_of ?ent.
  ?cap skos:inScheme tgn;; gvp:prefLabelGVP [xl:literalForm ?capLab].
  ?ent skos:inScheme tgn;; gvp:prefLabelGVP [xl:literalForm ?entLab]}
```

Note: currently there is a bit of hesitation about the true code of this relation, so if the above doesn't work, try with gvp:tgn3101_capital_of.

5.3.12 Members of the European Union

Let's find the member countries of the European Union by associative relation, finds 25:

```
select * {
  ?c gvp:tgn3317_member_of [rdfs:label "European Union"@en];
  gvp:prefLabelGVP [xl:literalForm ?lab]}
```

5.3.13 Members of the United Nations

Let's find the member countries of the United Nations (190). But this time let's fetch:

- The English labels, not the GVP-preferred labels (which are usually in the Vernacular). All the UN members have an English label in TGN.

- Optionally, the date of joining (turns out no nation has ever left the UN).

```
select * {
  ?c gvp:tgn3317_member_of [rdfs:label "United Nations"@en];
  xl:prefLabel [xl:literalForm ?lab; dct:language gvp_lang:en].
  optional {
    [rdf:subject ?c; rdf:predicate gvp:tgn3317_member_of; rdf:object [rdfs:label "United Nations"@en];
    schema:startDate ?start]}}
```

5.3.14 Geo Chart with SPARQL

We can use the excellent [sgvizler](#) library that makes Google Charts out of SPARQL results. Let's explore the year each country joined the UN:

```
select ?country ?year {
  [rdf:subject [xl:prefLabel [xl:literalForm ?country; dct:language gvp_lang:en]];
  rdf:predicate gvp:tgn3317_member_of;
  rdf:object [rdfs:label 'United Nations'@en];
  schema:startDate ?start].
  bind(xsd:integer(str(?start)) as ?year)}
```

We need to fiddle with datatypes a bit:

- `str()` removes the datatype `xsd:gYear`
- `xsd:integer()` converts to int, so Google can set the color scale right.

You should read the [Google GeoChart documentation](#) for the available formats and options. You can see the code and results at <http://jsfiddle.net/valexiev/NULCH/>.

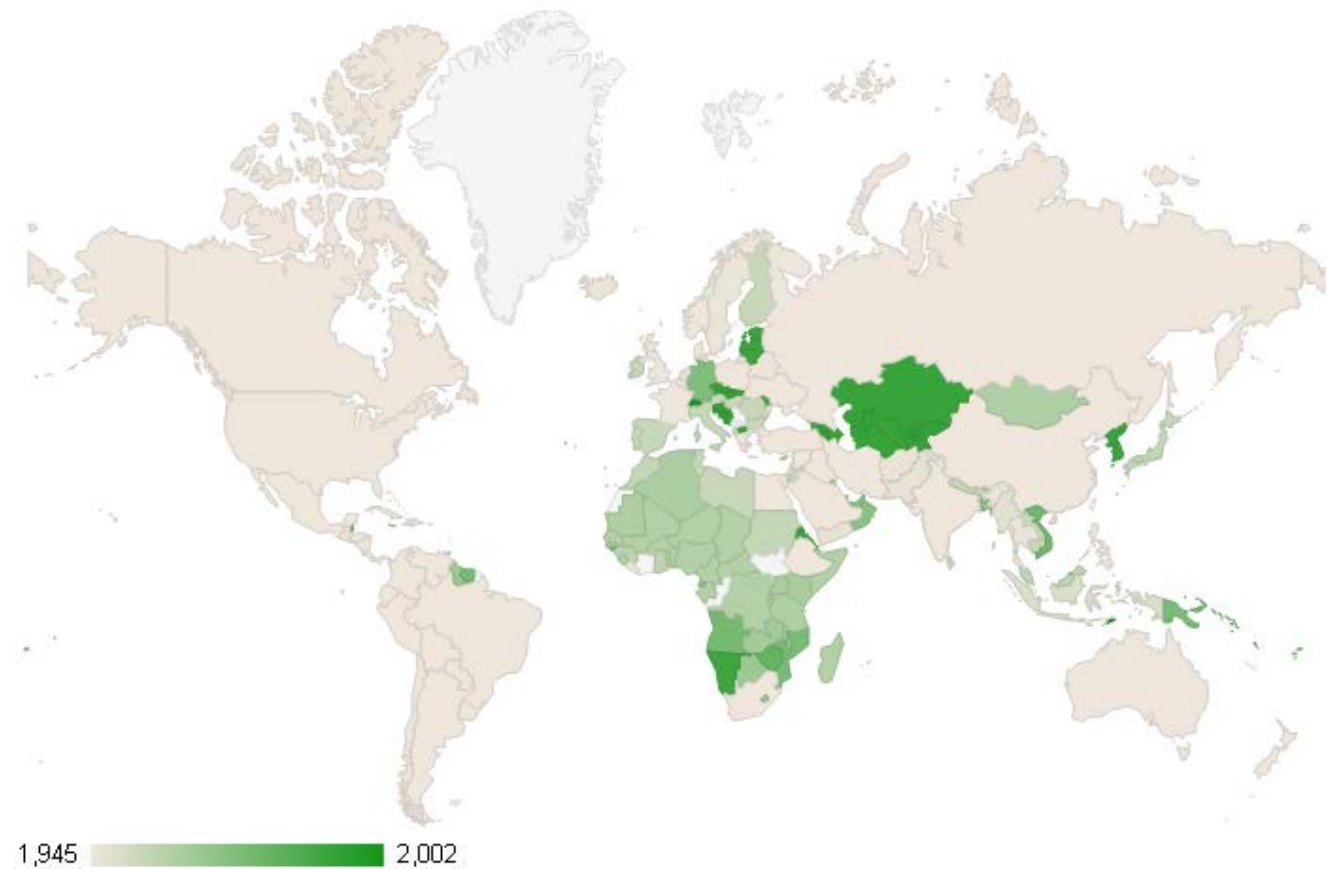
Hints:

- You must specify "sparql.json" as the endpoint, to ensure sgvizler gets the data in the required format:

```
data-sgvizler-endpoint="http://vocab.getty.edu/sparql.json"
```

- Explore the available chart types and options at [Google Charts Gallery](#)
- Tweak data-sgvizler-chart-options following the examples (the separator is |, don't enclose string values in anything)

Here is the final result:



5.3.15 Column Chart with SPARQL

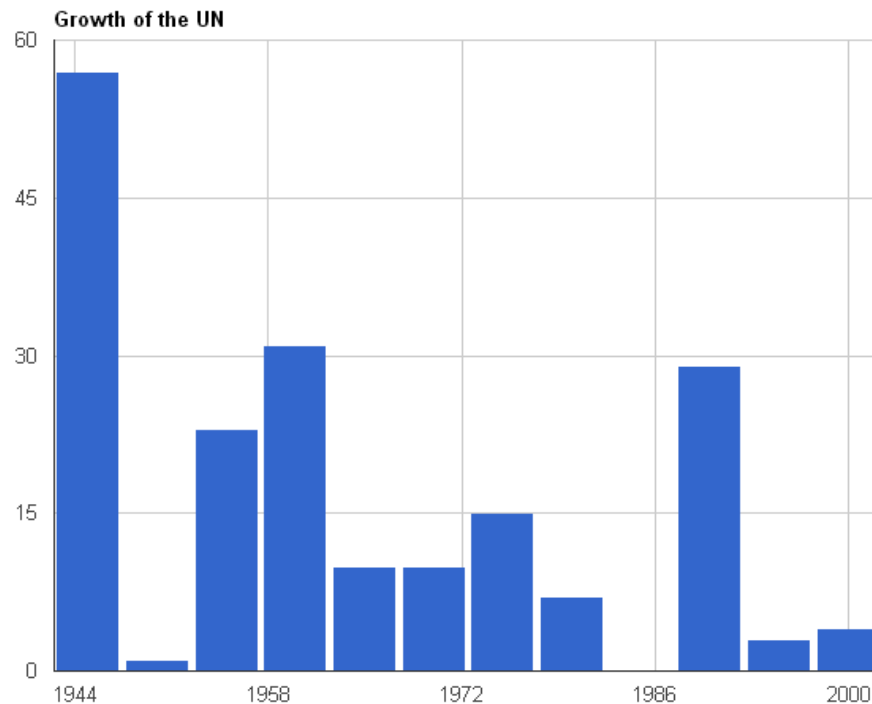
Let's explore the number of nations that joined the UN, grouping in buckets of 5 years:

```
select ?year (count(*) as ?countries) {
  [rdf:predicate gvp:tgn3317_member_of;
   rdf:object [rdfs:label 'United Nations'@en];
   schema:startDate ?start].
  bind (xsd:integer(xsd:integer(str(?start))/5)*5 as ?year)
} group by ?year order by ?year
```

We need to fiddle with datatypes a bit:

- `str()` removes the datatype `xsd:gYear`
- The inner `xsd:integer()` converts to int, so we can apply the division operation
- The outer `xsd:integer()` rounds down to integer

It takes more fiddling to get the [Google ColumnChart options right](#) (e.g. `hAxis.format=####|legend.position=none`). You can see the code at <http://jsfiddle.net/valexiev/TCr59/>, and here is the final result:



5.3.16 Countries and Capitals By Type and Containment

Rather than using the associative relation `tn3201_capital_of`, let's try to find nations and capitals by type and containment. Here we use direct type for `?capital` (since that type doesn't have any subtypes), but hierarchical type for `?nation`.

```
select distinct * {
  ?capital gvp:broaderPartitiveExtended ?nation.
  ?capital gvp:placeType [rdfs:label "national capital"@en].
  ?nation gvp:placeType|(gvp:placeType/gvp:broaderGenericExtended) [rdfs:label "sovereign state"@en].
  ?capital gvp:prefLabelGVP [xl:literalForm ?capital_name].
  ?nation gvp:prefLabelGVP [xl:literalForm ?nation_name]}
```

5.3.17 Places by Coordinate Bounding Box

Find places whose coordinates fall in a specific rectangle (in this case, enclosing the continental territory of The Netherlands):

```
select ?place ?name ?lat ?long {
  ?place skos:inScheme tgn: ;
  foaf:focus [wgs:lat ?lat; wgs:long ?long];
  gvp:prefLabelGVP [xl:literalForm ?name].
  filter (50.787185 <= ?lat && ?lat <= 53.542265 && 3.389722 <= ?long && ?long <= 7.169019)}
```

Note that the comparison may be problematic for rectangles that cross the +/-180 degree meridian.

Take a look at [TGN Overview](#): the coordinates are available:

- One hop away from the main node (`foaf:focus`) as `wgs:lat`, `wgs:long`
- Two hops away from the main node (`foaf:focus/schema:geo`) as `schema:latitude`, `schema:longitude`.

We prefer the shorter variant.

Note: a few regions like the Great Lakes Region include a bounding box in `foaf:focus/schema:geo/schema:box`.

This returns 23084 places:

- Not all of them are in The Netherlands: some are in neighboring countries.
- On the other hand, this excludes overseas territories of The Netherlands

5.3.18 Places Within Bounding Box

OWLIM includes some [Geo-spatial Extensions](#) that allow you to simplify the above query, make it more efficient, and avoid the confusion around the +/-180 degree meridian:

```
prefix ontogeo: <http://www.ontotext.com/owlim/geo#>
select * {
  ?place skos:inScheme tgn: ;
  foaf:focus [ontogeo:within(50.787185 3.389722 53.542265 7.169019)];
  gvp:prefLabelGVP [xl:literalForm ?name]}
```

We don't mention the coordinate fields, because ontogeo:within() looks in wgs:lat and wgs:long.

5.3.19 Places by Type Within Bounding Box

Let's specialize the previous query and look for castles around The Netherlands, we get 170:

```
prefix ontogeo: <http://www.ontotext.com/owlim/geo#>
select distinct * {
  ?place skos:inScheme tgn: ;
  gvp:placeType|(gvp:placeType/gvp:broaderGenericExtended) [rdfs:label "castles (fortifications)"@en];
  foaf:focus [ontogeo:within(50.787185 3.389722 53.542265 7.169019)];
  gvp:prefLabelGVP [xl:literalForm ?name];
  gvp:parentString ?parents}
```

5.3.20 Places Outside Bounding Box (Overseas Possessions)

Let's find places that are administratively part of The Netherlands, but are not within the bounding box of its main continental territory (i.e. Overseas Possessions):

```
select ?place ?name ?lat ?long {
  ?place skos:inScheme tgn: ;
  foaf:focus [wgs:lat ?lat; wgs:long ?long];
  gvp:prefLabelGVP [xl:literalForm ?name];
  gvp:broaderPartitiveExtended [rdfs:label "The Netherlands"@en]
  filter (!(50.787185 <= ?lat && ?lat <= 53.542265 && 3.389722 <= ?long && ?long <= 7.169019))}
```

This includes 564 places in [Bonaire, Sint Eustatius and Saba](#) (Lesser Antilles)

5.3.21 Places Nearby Each Other

OWLIM includes some [Geo-spatial Extensions](#) that go beyond searching within a rectangle: you can search for places nearby other places (by radius), compute distance, and search within a polygon.

Let's look for castles nearby mountains (within 10km). We look only for direct types gvp:placeType, because adding 2 hierarchical types slows down the query considerably but doesn't find more results.

The distance can be specified in "km" (default) or in "mi". We also return the distance (always in km), and sort by it.

TGN knows about 719 castles within 10km of a mountain (e.g. Castillo de Atalaya is exactly at Sierra de la Atalaya in Murcia, Spain). Get your hiking shoes ready!!

```
select * {
  ?castle skos:inScheme tgn;;
  gvp:placeType [rdfs:label "castles (fortifications)"@en];
  gvp:prefLabelGVP [xl:literalForm ?castle_name];
  foaf:focus [wgs:lat ?castle_lat; wgs:long ?castle_long];
  gvp:parentString ?parents.
  ?mountain skos:inScheme tgn;;
  gvp:placeType [rdfs:label "mountains"@en];
  gvp:prefLabelGVP [xl:literalForm ?mountain_name];
  foaf:focus [wgs:lat ?mountain_lat; wgs:long ?mountain_long;
    ontogeo:nearby(?castle_lat ?castle_long 10)].
  bind (ontogeo:distance(?castle_lat, ?castle_long, ?mountain_lat, ?mountain_long) as ?dist)
} order by asc(?dist)
```

Notes:

- `onto:within` and `onto:nearby` are predicates, so they take their arguments without commas (this is in fact an `rdf:List`), and are used in a predicate position
- `onto:distance()` is a function, so it takes its arguments comma-separated, and is used as a function (in this case in a `bind()`)
- You can see more examples of OWLIM-specific Geo queries at the link above
- We first find all castles (total of 490) and then restrict mountains (total of 25282) by `ontogeo:nearby()`. This method is significantly faster than either of:
 - First find castles, then find mountains, then filter by `?dist<=10`
 - First find mountains (greater cardinality), then restrict castles (smaller cardinality)

5.4 Counting and Descriptive Info

Counts can give you a better feel of the available semantic information.

5.4.1 Descriptive Info from VOID

As an alternative to getting the VOID descriptive info as a Turtle file (see [VOID Deployment](#)), you can query it in the repository. The following query returns all descriptive triples (same as the Turtle file):

```
select * {graph <http://vocab.getty.edu/.well-known/void> {?s ?p ?o}}
```

5.4.2 Number of Entities from VOID

Get the class partitions (number of entities per class), ordering by decreasing count:

```
select ?class ?count {?void void:classPartition [void:class ?class; void:entities ?count]}
order by desc(?count)
```

This uses the pre-calculated counts in the VOID descriptive info, so it's very fast.

5.4.3 Number of Entities (Dynamic)

One of the class partitions (see previous section) does not return very meaningful numbers: `bibo:Document` returns both local and global sources. We can get them separately with the queries below, though this is slower:

- [Local Sources](#)

```
select (count(*) as ?c) {?x a bibo:DocumentPart}
```

- Global Sources

```
select (count(*) as ?c) {
  ?x a bibo:Document
  filter not exists {?x a bibo:DocumentPart}}

```

5.4.4 Number of Terms per Language

```
select ?lang ?name (count(*) as ?c) {
  ?x xl:prefLabel|xl:altLabel ?l.
  optional {?l dct:language ?lang. ?lang gvp:prefLabelGVP/xl:literalForm ?name}
} group by ?lang ?name order by desc(?c)
```

5.4.5 Number of Revision Actions

Count of [Revision History](#) actions (Create, Modify, Publish) by ?action kind and entity ?type

```
select ?action ?type (count(*) as ?c) {
  ?x skos:changeNote ?y. ?y dc:type ?action
  bind(if(exists{?x a bibo:Document}, "Source", "Subject") as ?type)
} group by ?action ?type
```

5.5 Explore the Ontology

The [GVP Ontology](#) is described in this document, and a reference is provided in the [ontology documentation \(namespace document\)](#). But you can also explore it with queries.

5.5.1 Ontology Classes and Properties

This returns the classes and properties defined by the ontology, together with some details:

```
select ?x ?type (coalesce(?descr,?label) as ?description) ?domain ?range {
  ?x rdfs:isDefinedBy <http://vocab.getty.edu/ontology>; a ?type.
  optional {?x dct:description ?descr}
  optional {?x rdfs:label ?label}
  optional {?x rdfs:domain ?domain}
  optional {?x rdfs:range ?range}}
```

- Uncheck "Include inferred" so you don't get duplicate rows due to super-types.
- coalesce() returns the more detailed dct:description if available, else rdfs:label

5.5.2 Ontology Values

This returns all values (skos:Concepts, mostly [Term Characteristics](#)) in small schemes defined by the ontology:

```
select * {
  ?x skos:inScheme [rdfs:isDefinedBy <http://vocab.getty.edu/ontology>; rdfs:label ?scheme];
  skos:prefLabel ?value; skos:scopeNote ?note; skos:example ?example}
```

Because the scheme URL is always a prefix of the value URL, we print the scheme's label instead. Some example results (the prefix <http://vocab.getty.edu/> is omitted):

x	scheme	value	note	example
term/flag/Vernacular	Term Flag	Vernacular	Term is in the "vernacular" language	"Firenze" is the vernacular in Italian (TGN)
term/kind/Abbreviation	Term Kind	Abbreviation	Term is an abbreviation, initialism, or acronym	DVD, CD-ROM (AAT)
term/kind/CommonTerm	Term Kind	Common term	Preferred common language term. Used for subjects that also include a Scientific term	domestic cat (AAT)
term/kind/ScientificOrTechnical	Term Kind	Scientific or Technical term	A Scientific term	"Felis domesticus" is the scientific term for "cats" (AAT)

You can click on the value URLs (first column) and then the Object tab to explore terms having that characteristic. For example, there are 628 AAT terms that have Term Flag "Vernacular":

Vernacular

Source:<http://vocab.getty.edu/aat/term/flag/Vernacular>

Subject	Predicate	Object (100 of 628)	All	Download
				Inf
Statements in which the resource exists as a object.				
Subject	Predicate			
aat_term:1000409692-en, aat_term:1000408629-en, aat_term:1000402435-en, aat_term:1000402435-ja-Latn, aat_term:1000402458-en, aat_term:1000402458-ja-Latn, aat_term:1000402462-en, aat_term:1000402462-ja-Latn, aat_term:1000402476-en, aat_term:1000402476-ja-Latn, aat_term:1000402478-en, aat_term:1000402478-ja-Latn, aat_term:1000408611-en, aat_term:1000397833-en, aat_term:1000397833-ja-Latn, aat_term:1000397851-en, aat_term:1000397851-ja-Latn, aat_term:1000397868-en, aat_term:1000397868-ja-Latn, aat_term:1000397873-en, aat_term:1000397873-ja-Latn, aat_term:1000408793-en, aat_term:1000408793-ja-Latn, aat_term:1000408698-en, aat_term:1000397282-en, aat_term:1000397282-ja-Latn, aat_term:1000397346-en, aat_term:1000397346-ja-Latn, aat_term:1000397350-en, aat_term:1000397350-ja-Latn,	gvp:termFlag			