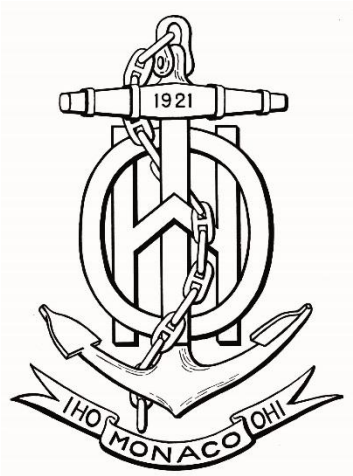
INTERNATIONAL HYDROGRAPHIC ORGANIZATION



IHO GUIDELINE STANDARD

FOR CREATING S-100 PRODUCT SPECIFICATIONS

PART B

Draft Version 0.1

2017-12-31

Special Publication No. S-??

Guideline for Creating an S-100 Product Specification

Part B - Execution

Published by the
International Hydrographic Organization
MONACO

Error! Reference source not found.

Revision History

Changes to this Specification are coordinated by the IHO S-100 Working Group. New editions will be made available via the IHO web site. Maintenance of the Specification shall conform to IHO Technical Resolution 2/2007 (revised 2010).

Version Number	Date	Author	Purpose
0.1	2017-12-31	RM, EM	First draft

[Copyright statement]

Contents

Part B - Steps in the Development of a Product Specification

1 Introduction

This guideline is intended to serve as a guide for anyone planning to develop an S-100 compliant product specification. The guideline consists of two main parts; Part A (a separate document) is an in-depth description of the various components of an S-100-based product specification, and Part B (this document) describes the typical steps and activities involved in creating an S-100-based product specification. Part B describes the overall process, specific activities and tasks, and includes hints for solving specific problems while the product specification is being developed.

1.1 References

IHO S-58 ENC Validation Checks, Edition 6.0.0, May 2017

IHO S-99 Operational Procedures for the Organization and Management of the S-100 Geospatial Information Registry, Edition 1.1.0, November 2012

IHO S-100 Information Paper, January 2011

IHO S-100 Universal Hydrographic Data Model Editions 3.0.0 and 4.0.0 (in preparation)

IHO S-122 Marine Protected Areas, 2017 (draft)

1.2 Terms and Abbreviations

GML	Geography Markup Language
IALA	International Association of Lighthouse Authorities
IEC	International Electrotechnical Commission
IHO	International Hydrographic Organization
IMO	International Maritime Organization
RENC	Regional ENC Coordinating Centre
XML	eXtensible Markup Language
XSD	XML Schema Definition

1.3 Overview of steps for development

The stages in developing a product specification are summarized below. Subsequent sections describe the stages in detail. The final section describes work processes involved in developing a product specification.

1) **Initiation**. Identify the need for a new data product, define its scope, and decide the boundaries between the new product and existing data product specifications. Describe typical application use cases.
2) **Develop the domain model/application schema**. Define the classes and attributes that describe the domain and which are relevant to the data product, define the relationships between the classes, specify applicable constraints. Prepare one or more UML diagrams describing the domain model.
3) **Populate the GI registry from the domain model**. Propose amendments to existing classes and attributes and propose new classes and attributes for addition to the GI registry using the IHO registry interface. Follow up on any returned proposals or queries from the registry manager or domain control body.
4) **Develop the feature catalogue**. Prepare the XML feature catalogue from the feature and information classes, attributes and relationships as approved in the GI registry.

5) **Define data transfer modes and packaging**. Determine whether data products are to be delivered as data files contained in transfer (exchange) sets, by web services (and if so, identify or outline a service protocol), e-mail, etc. Determine whether data is to be delivered in real or near real-time. Identify constraints and requirements arising from delivery mechanisms and communication constraints such as message size, bandwidth limitations, availability of communications to customers, licensing and payments, encryption, etc.

6) **Define metadata**. Survey the metadata elements listed in S-100 for their appropriateness to the data product and its allowed packaging and delivery methods. Define appropriate values and restrictions for the metadata elements listed in S-100. Consider whether additional product-specific metadata elements are needed.

7) **Define the data format**. Select an appropriate data format. S-100 provides for 3 standard delivery formats (ISO 8211, GML, and HDF5). Define format-imposed items (e.g., embedded header metadata). Prepare format-specific artifacts if necessary (e.g., GML "application schema" XSD files for the GML format). If a non-standard data format is defined instead of one of the standard formats, specify conversion rules from the data format to one of the standard formats.

8) **Define portrayal**. Determine the symbols to be used for portrayal and the rules for generating displays from the data product. Prepare a portrayal catalogue.

9) **Define spatial reference system**. Identify the recommended coordinate reference system and vertical datum(s).

10) **Define data product packaging and maintenance**. Define the content and structure of delivery packages, updating of data, and any auxiliary content delivered either with or as an adjunct to data.

11) **Define validation checks**. Define tests for the spatial, structural, and conceptual integrity of datasets. Define format-specific implementations of validation checks (e.g., Schematron rules for the GML format).

12) **Prepare for interoperability** with other data products. Jointly with the IHO interoperability catalogue maintenance team, determine which if any product groups in interoperability catalogues are supplemented or enhanced by the data product, and determine whether and how the IHO interoperability catalogue will be affected by the new product, including updates to display priorities, interleaving, predefined combinations, and other interoperability rules and operations.

13) **Prepare sample data** for test-beds. Create sample data conforming to the data format and feature catalogue.

14) **Testing and feedback**. Carry out tests of data production and use of sample data in selected applications to validate the correctness, completeness, consistency, and utility of the product specification, including related artifacts such as the feature catalogue and XML schemas.

Development of product specifications will be an iterative refinement process, and authors should expect to cycle through the stages above multiple times, in smaller or larger cycles depending on experiences and results of each stage. For example, the first preparation of sample data (Stage 13) might identify gaps in the domain model and require revisiting Stage 2 (development of the domain model) changes to the domain model; or the development of portrayal rules (Stage 8) may identify gaps in the domain model (Stage 2). Passage through some intermediate stages in the cycle may be trivial on the second and later passes, as the results may be unaffected by discoveries during later stages in the process – for example, packaging and delivery or choice of spatial reference systems are likely to be unaffected by changes identified during the preparation of sample data. Figure 1-1 depicts the development process, including probable cycles.

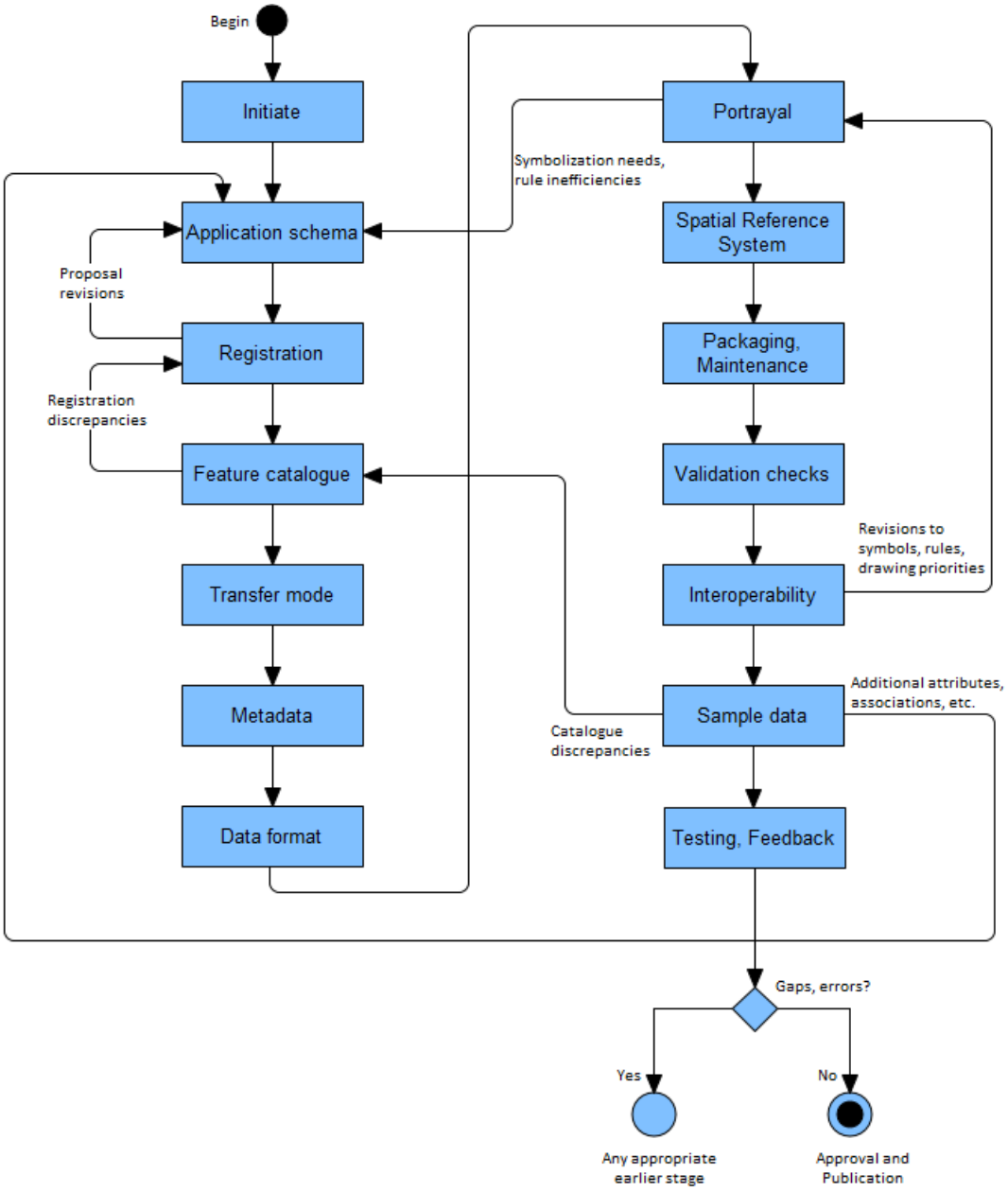


Figure 1-1. Product specification development process

Appendix 11-A in S-100 3.0.0 describes an idealized process (Figure 11-A-1 – Product specification process) for the basic mechanical steps in developing a domain model plus related registry actions. The figure also includes black-box stages for “Coordinate Reference System” and “Product Specification Documentation” at the end of the process. The actual process can be expected to be more an iterative refinement process, as described in the previous paragraph.

2 Initiation

The initiation phase consists of the following activities. Except for the first (identifying the need) most of the other activities can be carried on in parallel, e.g., recruitment of a project team should be an ongoing task commencing at the latest after determination if an existing specification can be extended.

1) Identify a need that can be filled by a data product – this will normally be the result of external circumstances such as definition of a service portfolio, application-driven demand for new kinds of information, new legal requirements for shipping, IMO decisions, etc.
2) Define the scope of overall product – the subject area, the kind of information it is expected to contain, and equally importantly, what information it will not contain.
3) Determine if existing product specifications can be extended. If so, such an extension will probably consume less time and effort than developing a new product specification.
4) Determine sub-areas within the product, i.e., the scopes within the product specification, or a new scope for an existing product specification
5) Define constraints – domain, application, platform, etc.
6) Collect samples of source information. These generally include existing databases, official, unofficial, government, and commercial publications – especially those in wide use.
7) Define application use cases
8) Outline application functionality enabled by the data product
9) Define delivery modes (transfer set, messages, web services, etc.).
10) Obtain approval from the appropriate sponsoring organization.
11) Put together a project team.

3 Develop data model (Application schema)

3.1 Introduction

The Application Schema as defined in S-100 is usually synonymous with “domain model” as the latter term is used in information modeling. It is a specification of the classes, attributes, and relationships relevant to the data product.

Generally, a product specification will include only a single application schema (which may be broken up into multiple diagrams). However, in theory a product specification that describes different scopes may need to distinguish application schema for different scopes – for example a product that includes both vector and coverage data might need two application schemas, or a product designed for information exchange by exchange set as well as web services may need two application schemas.

To minimize complexity, product specification developers should endeavor to avoid defining multiple application schemas. Note that this does not prevent a single application schema from being depicted using multiple diagrams.

The application schema should describe only the features, information types, and their attributes and relationships which are to be included in the data product. Any other classes, constraints, or elements (such

as actors, state diagrams, process flows, etc.) which are considered necessary for an understanding of the data product and its role in applications, services, or service portfolios should be distinguished from the application schema and documented separately in the product specification.

Application schemas are generally a tradeoff between abstraction, complexity, and delivery and implementation considerations. Model developers should try to limit the number of model elements while still allowing implementations to make appropriate (conceptually appropriate, logically appropriate, consistent, correct, and performance-based) distinctions.

The principles of data normalization learned in relational database design should be kept in mind, but model developers should also note that an S-100-based domain model and applications schema are object-attribute-relationship models and not database designs.

3.2 Steps in model development

Determine whether the data product is coverage or vector data. Coverage data is characterized by values of characteristics distributed over an area or areas, while vector data is characterized by localized regions (points, and/or areas) that possess boundaries and do not exhibit internal variation in characteristic values (or where such internal variations can be ignored).

Identify the concepts in the application domain. This will involve reviewing the source material to identify important features and information chunks in the domain which will be useful to end-users in the context of the use cases defined in the initiation phase. Source material will include the sample texts identified in the initiation phase and, if available, documentation and data dictionaries of relevant applications, requirements for existing or hypothetical applications, related standards and circulars from IEC, IMO, IALA, etc.

Search for existing concepts (classes, attributes, and relationships) in the registry which can be re-used. Searching for key words in the name of the concept or element (feature, attribute, etc.) is the best search option afforded by the registry interface at this time.

Develop new concepts only for those concepts which do not yet exist in the registry. This will involve examining the source material mentioned earlier in more detail to pin down concepts and their definitions. The process for submitting proposals is described in S-99.

In S-100 4.0 and the next-generation GI Registry, concepts which from the concept register must be entered into the data dictionary registry database by designating its item type (e.g. feature or attribute, etc.) and submitting additional information.

Define the classes and attributes that describe the domain and which are relevant to the data product. If classes and attributes already defined in the registry or existing product specifications can be reused, so much the better. If not, the project team will need to develop and define classes and attributes, including listed values for enumeration and codelist types.

Define the relationships between the classes. Relationships should be defined in order to capture relationships that exist in the real world and to make links which are useful for application processing. Both reasons will often apply.

EXAMPLE 1: A structure/equipment relationship between classes modeling structure objects and classes modeling equipment mounted on the structure.

EXAMPLE 2: A contact information relationship between classes modeling pilot service areas and contact information for pilot services available in that area.

Specify any constraints applicable to the classes, attributes, and relationships. Examples are constraints requiring conditional encoding of attributes, exclusive relationships (i.e., when an instance is allowed to participate in only one of multiple possible relationships), etc. Structural restrictions are generally depicted in UML class diagrams, while value restrictions on individual attributes are generally not (to reduce clutter). Whether depicted or not, any restrictions must be enforced (if possible) and documented in the appropriate section or artifact of the product specification.

Prepare one or more UML class diagrams describing the domain model. Recommended practices for S-100 models are based on ISO TC211 recommended practices as modified by Section 3.6 of this document.

Prepare application schema documentation of the UML model of the application schema.

Prepare supporting text explaining the overall structure of the application schema. Also prepare text for each diagram explaining the purpose of each diagram, the relationships between the classes. Also explain any additional subtleties of the classes or their relationships that may not be obvious or should be specially noted. This text should not be a mere listing of classes, attributes, and relationships, but instead should clarify the purpose of the model fragment depicted in the diagram by explaining what domain phenomenon the diagram captures and how the classes and relationships express it. For example, from S-122:

> *Some protected areas require reports to be filed with authorities when certain events occur such as an animal strike or pollution event. Other areas require reporting to specified authorities when entering, leaving, etc. These requirements are modeled by association of a ShipReport class to the Authority class. The area in question is modeled by a feature of the requisite type, e.g., a MarineProtectedArea or VesselTrafficServiceArea, as described in clause 6.2.1.3 [of S-122]. Any time requirements or constraints on the filing of the report are described by the noticeTime attribute, with explanations, if any provided in text form in the textContent attribute of ShipReport. Required reporting formats, if necessary to be included, are also described in the textContent attribute. [Figure 3-1] shows the model elements that are used to carry these conditions, note that not all permitted associations or roles are included, in order to reduce clutter.*

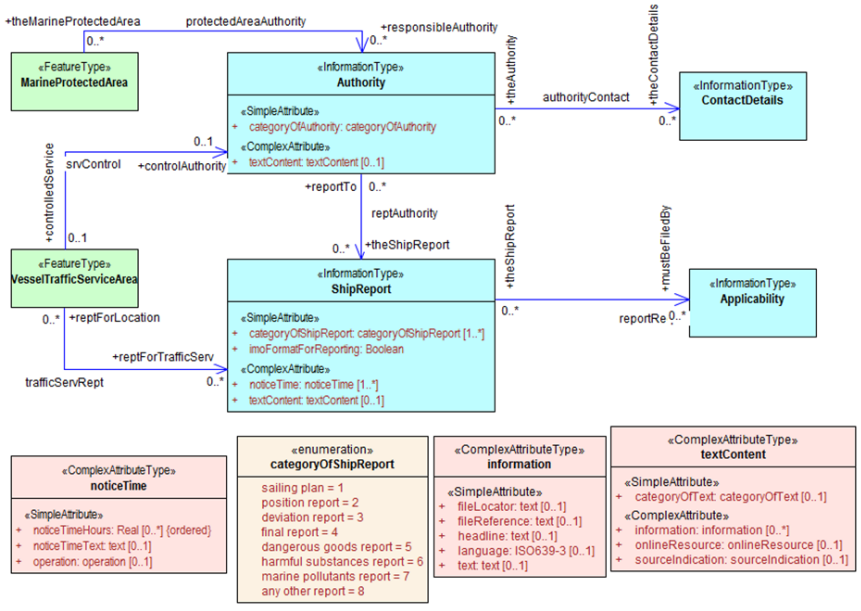


Figure 3-1. UML diagram depicting part of the S-122 Application Schema (from S-122)

Source information collected and use cases developed in the initiation stage should be reviewed regularly, as both will be useful in identifying gaps in application schemas under development.

3.3 Relationship to General Feature Model

Features and information types in the application schema must be realizations of the meta-classes S100_GF_FeatureType and S100_GF_InformationType from the S-100 General Feature Model (GFM) (S-100 Part 3 Figure 3-1), or subclasses of a class that realizes the appropriate meta-class. Attributes must be realizations of the thematic or spatial attribute meta-classes defined in S-100 Part 3 (Figure 3-2), or subclasses of a realization. Product specifications may define local root classes from which all their feature and information classes are derived, as shown in the figure below, or may realize features and information type classes from S100_GF_FeatureType and S100_InformationType respectively. An optional diagram depicting the realizations may be included in the Application Schema section of the product specification, as depicted in the figure below.

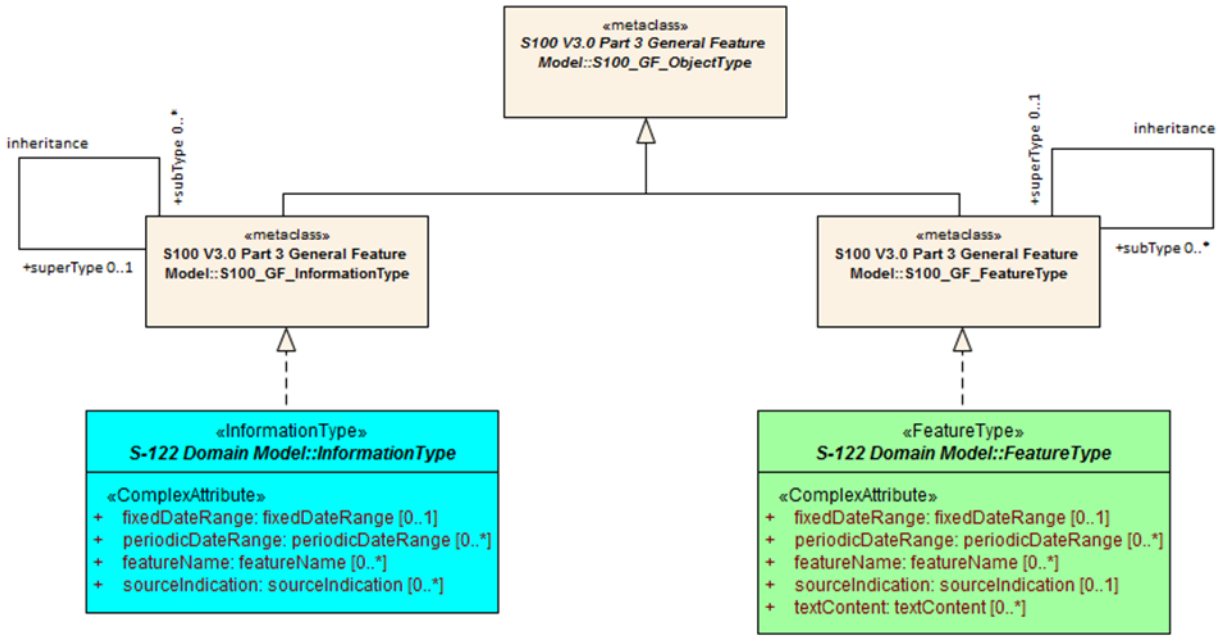


Figure 3-2. Example of realization from S-100 GFM

3.4 Rules for application schemas

S-100 rules for application schemas are based on ISO 19109. The S-100 rules for application schemas are given in S-100 Part 3 (clauses 3-6 and 3-7 in edition 3.0.0).

3.4.1 Application schemas for vector data

The rules for application schemas for vector data are given in S-100 clause 3-6. The main rules are summarized below:

- Features, information types, and complex attributes must be modeled as classes.

- Relationships are modeled by UML associations or association classes (the latter only when the association itself is characterized by attributes – see Section 3.5.4.2 in this document).

- Attributes are modeled by UML attributes in the appropriate class.

- Associations must be labeled (have association names). Navigable association ends must also be labeled (should have role names).

Note: Diagrams may suppress depiction of labels for clarity and to reduce clutter. Labels may be defined by specific rules given in the product specification text instead of the UML diagram (e.g., a product specification is allowed to 'label' an association end using a statement like "The role of *FeatureX* in all its associations is *theFeatureX*" (see S-100 Edition 3.0.0 clause 3-5.4.5 on default names for association ends).

- Spatial attributes must be modeled either as attributes with data type one or more (i.e., union) of the allowed spatial types in the spatial schema, or an association between the class that represents a feature and one of the spatial objects defined in the spatial schema.
- Enumeration types and their listed values must be modeled by UML enumerations; codelists must be modeled as UML classes with tags specified in S-100 Edition 3.0.0 clause 3-6.7.
- Standard schemas (e.g., the spatial schema, feature catalogue schema) shall not be extended within application schemas.

- All classes used within an application schema for data transfer shall be instantiable. This implies that the integrated class must not be stereotyped <<interface>>.

- An UML application schema shall be described within a UML package, which shall carry the name of the application schema and the version stated in the documentation of the package.

For detailed rules, see S-100 clause 3-6.

3.4.2 Application schemas for coverage data

The rules for application schemas for vector data are given in S-100 clause 3-7 and Part 8.

The rules are similar to the rules for vector data, with the following differences:

- Spatial types for coverage features are modeled by the appropriate grid or pointset type defined in S-100 Part 8.

3.4.3 Additional rules

Names of features and information types must be their camel case codes.

Vector feature classes must use the stereotype <<FeatureType>> and information classes must use the stereotype <<InformationType>>.

Coverage *type* elements (describing the coverage geometry) compliant to S-100 Edition 3.0.0 should use the appropriate stereotype from S-100 Part 8, and application schemas for coverage data may depict the data attributes by defining a <<FeatureType>> element with the thematic data attributes and associating it with the coverage type element. This rule may be revised in S-100 Edition 4.0.0 or later editions or as and when the base ISO standard (ISO 19123) is updated.

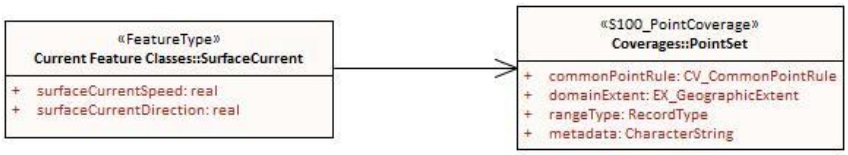


Figure 3-3. Example of coverage feature and type elements conforming to S-100 ed. 3.0.0 (from S-111)

If necessary, product specifications may use domain-specific stereotypes in addition to the standard stereotypes.

Abstract classes must be indicated as such by italicizing the class name (Enterprise Architect does this automatically if the “Abstract” checkbox is checked in the UI).

S-100 states that “the use of multiple inheritance shall be minimized, because it tends to increase model complexity”. Multiple inheritance is the situation where a class has more than one immediate superclass. Application schema developers should note that multiple inheritance contravenes the S-100 GFM, which allows feature and information types to have at most one super-type.

3.5 Other conventions and recommendations

3.5.1 Reuse and harmonization

The registry should be checked for existing elements that can be re-used before new elements are defined. Features, information types, and attributes should be re-used whenever possible. Structure and associations should be harmonized with S-101 and other existing related or complementary products. Defining similar but slightly different items should be avoided unless absolutely essential. Extensions such as additional listed values in an enumeration can be proposed to the registry, but conflicts such as different definitions for the same terms must be avoided if at all possible. Existing items may be re-usable with the addition of product specific constraints, such as limiting the set of allowed values for an enumeration or codelist type. Such harmonization includes, for example, re-using complex attributes defined in other product specifications with restrictions that exclude some of their sub-attributes.

3.5.2 Features and information types

3.5.3 Superclasses and subclasses

Defining abstract superclasses is recommended when 3 or more conceptually similar classes exist in the model, the similar classes have some of the same attributes or relationships, and the allowed values of shared attributes are the same. It is not necessary that the classes bind exactly the same sets of attributes or have exactly the same relationships.

Note that both associations and attributes are inherited by sub-classes, unless explicitly overridden.

3.5.3.1 Subclasses versus category attribute

One common problem arises when a class can be partitioned into disjoint subsets distinguished by a type characteristic. For example, the class “Buoy” can be partitioned into different types of buoys: cardinal buoys, lateral buoys, isolated danger buoys, safe water buoys, special-purpose buoys, emergency wreck markers, and installation buoys. S-101 defines a feature class for each of the 7 types. An alternative approach might

be to model a **Buoy** feature class with a **categoryOfBuoy** attribute that can take one of 7 values (cardinal, lateral, etc.) as depicted in Figure 3-4.

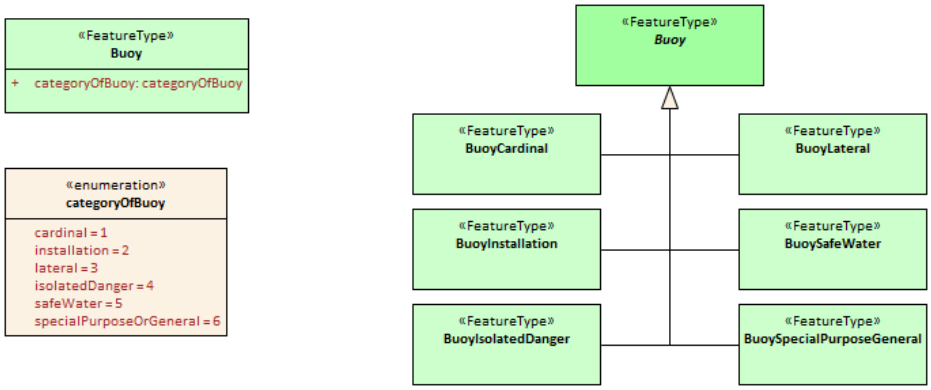


Figure 3-4. Illustration of alternative models using category attribute and sibling subclasses

If the subclass approach is adopted, it may result in sibling classes that have the same attributes but differ in the class name and definition. In theory such siblings can be removed and a "categoryOf..." attribute added to the superclass. Consideration of the following issues may help resolve the question of which approach is better:

1) Will either approach result in a significant divergence from some external source?
2) Will the “category attribute” approach cause issues for portrayal because the symbols for the different types are presumably different? Or will the subclass approach result in unnecessary portrayal rules because the symbols are the same?
3) Will any of the sub-classes have its own specific attributes or relationships? If so, the sub-class approach is preferable.
4) Are the different categories/subclasses in (or likely to be placed in) different viewing groups, or have different drawing order? If so, there is a slight preference for making sub-classes. (Only “slight” because the portrayal rules and interoperability catalogue can use attribute values in assigning viewing groups to feature instances.
5) If they are made sub-classes are there going to be situations where it may be necessary to encode coincident objects with different categories? The answer Yes suggests a preference for the categoryOf... approach.
6) Are the subclasses conceptually very different? Yes implies the sub-classes approach.
7) Which approach is likely to be compatible with external resources like existing databases and implementations?
8) If there are a large number of subtypes, then the categoryOf... approach may be preferable because it leads to more compact representations in UML diagrams and more compact DCEGs. (“Large” is obviously subjective, but will generally between 5 and 9 based on research into human cognitive psychology and probable implementation methods in user interfaces – there will be variations dependent on concept semantics and similarities).
9) Overall complexity of the application schema and feature catalogue. Sibling sub-classes of features (or information type) generate more artifacts and documentation than a category attribute. They

certainly mean an additional table for each sub-class in the DCEG, an additional XML element for each in the feature catalogue, and a box in the UML diagram application schema for each class. To that extent sibling sub-classes are a greater cognitive burden on encoders and developers.

Note that some of the buoy types have unique attributes (cardinal buoys, installation buoys, lateral buoys, and special/general buoys), and applying criterion (3) would lead to the decision to use sibling subclasses. (It is interesting to further note that since the distinct attributes of the subclasses are all "category" attributes, in theory it is possible to merge those categories into the "categoryOfBuoy" enumeration.)

3.5.4 Associations and association classes

3.5.4.1 Navigability, source, and target

Association navigability should be indicated if the association is navigable in only one direction, i.e., you expect to access one object from the other, but not vice versa. Feature/feature and information/information associations are usually navigable in both directions, while feature/information associations must be navigable from the feature end but are generally not modelled as navigable in the other direction. UML regards navigability information in UML diagrams as hints to implementations rather than hard requirements, and implementations and data formats are free to implement navigability in the most efficient manner.

Unidirectional navigability will normally also determine the source and target of the association.

An association's source and target should be grammatically and semantically compatible with the name and definition of the association, e.g., for the association Person/subscribes/Magazine the source should be Person and the target Magazine.). Enterprise Architect always has a source and target for associations (see the figure below).

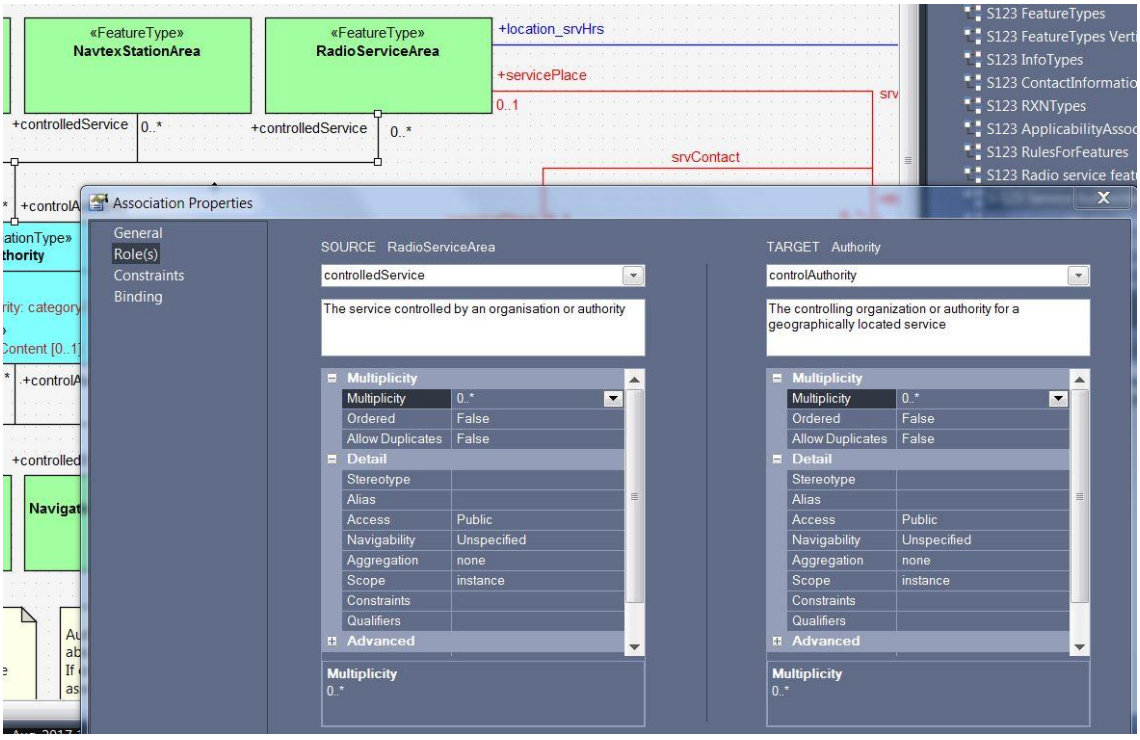


Figure 3-5. Association data screen in Enterprise Architect showing roles

Figure 3-5 depicts one of the data entry screens for an association in Enterprise Architect, depicting information entry for association ends. The association name would be entered on the “General” tab in the data entry panel.

For feature/feature associations both ends should be named; for feature/info associations the info end should be named and the feature end may be named. This is an S-100 requirement, not a UML requirement.

3.5.4.2 Association classes

Association classes can be regarded as means of adding parameters (characteristics) to associations, rather than either of the classes at the end of an association. An attribute of the association class characterizes the relationship between the classes at the ends of the association.

The use case for association classes is basically “whenever a relationship is characterized by one or more attributes.”

EXAMPLE 1: A specified set of vessels is COVERED by a regulation and another set of vessels is EXEMPT from the regulation. The sets of vessels are described by an information type class; the regulation by another information type class; and the relationship between them by an association class which has an attribute characterizing the relationship as inclusion or exclusion (of the specified subset in the specific regulation).

EXAMPLE 2: Vessels with specified cargo and dimensions must use a specified pilot boarding place, vessels of smaller dimensions are recommended to use the boarding place, and warships are exempt from using the pilot boarding place. The sets of vessels are described by an information type class; the pilot boarding place is a feature class; and the relationship between them by an association class which has an attribute stating whether the specified set of vessels is required/recommended/exempt from use of the pilot boarding place.

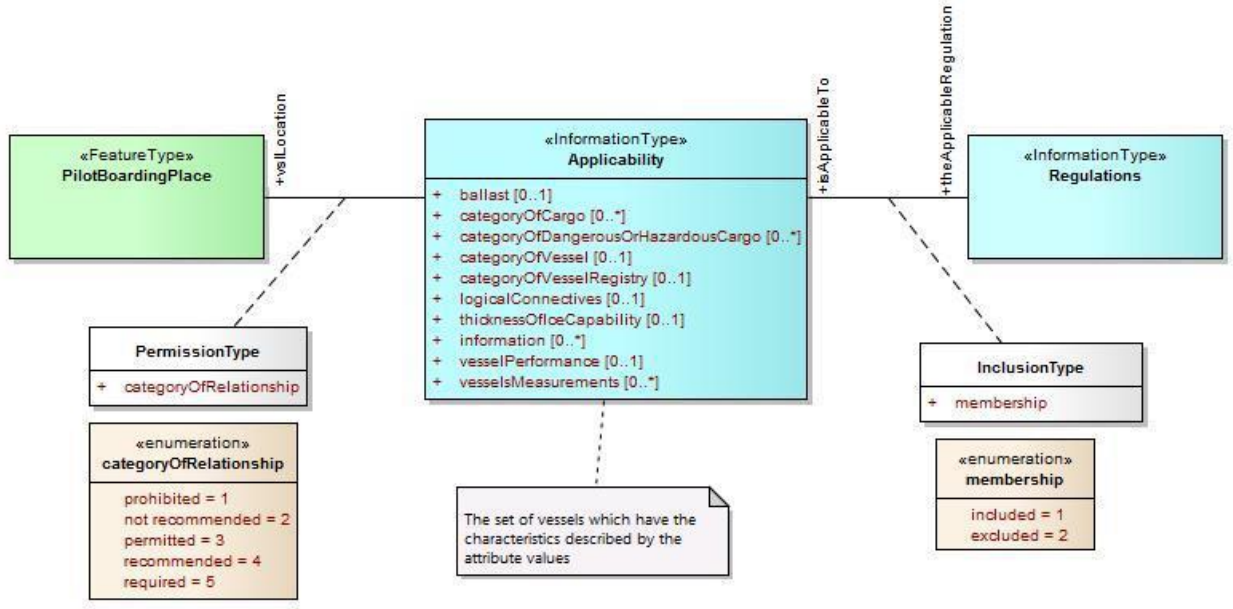


Figure 3-6. Examples of modelling with association classes

3.5.5 Attributes in general

Certain attributes may use the same set of listed values as other attributes. For example, the enumeration for compass points may be shared by attribute **windDirectionCompassPoint** being the direction for where wind is coming from, and **directionOfMovement** the attribute describing where a weather system is going toward. At present this can be simply handled in the attribute's definition. (The current incarnation of the registry does not include a data types register.)

Complex and spatial attributes can be modeled as either named attributes in the UML model class element with a type corresponding to the spatial primitive or the name of the complex attribute, or alternatively, separate model elements linked to feature/information class by an association (ordinary association for spatial type, composition for complex attributes). The two methods are illustrated in Figure 3-7 below. The second method is not suitable for complex models due to the additional boxes and association lines. The disadvantages of the second method become more apparent if the complex attribute itself contains complex attributes.

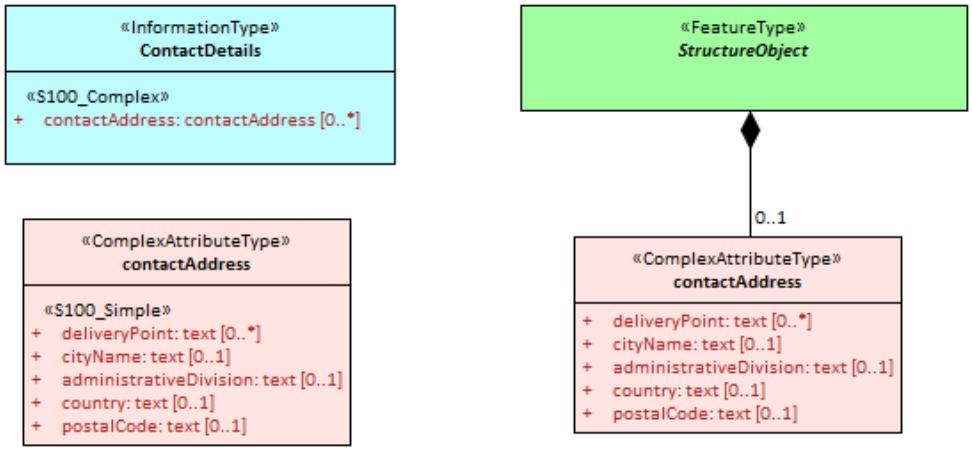


Figure 3-7. Methods of representing complex attribute bindings

3.5.6 Codelist and enumeration attributes

Codelists should be used only when an enumeration is either unusable or inefficient (for example, if the full list of values is not known to the specification authors or the list of allowed values is long, volatile, controlled by another authority, and/or shared by multiple domains). More detailed criteria for deciding when to use codelist attributes and when to use enumeration attributes are given in S-100 Appendix 11-C (clause 11-C.2).

3.5.7 Labels and definitions for listed values

Labels should be short but informative, keeping in mind that both end-users and encoders may view them, given that implementers of end-user display and production tools are likely to use listed value labels as ‘tooltips’ or explanatory text, or as the ‘display text’ of the attribute numeric code value for end users. End-users are more likely to see labels than full definitions due to other demands on their attention and screen display constraints.

3.5.8 Data types

S-100 defines a set of primitive and derived data types (Table 2a-4-10 in edition 3.0.0). Attribute values of thematic attributes should be one of the types listed in that table. If restrictions on the values are needed, product specifications may define constraints, if possible encoded using one or more of the elements in S100_CD_Constraints (length, range, pattern, and precision). Constraints which cannot be thus encoded must be documented as a 'Remark' or note.

The data types defined in S-100 can in principle be extended by application schemas but if this is done the product specification must define the extended data type in terms of the predefined data types in S-100 and use the predefined data type in the feature catalogue, since the feature catalogue schema does not currently permit user-defined data types. Data formats may use their equivalent built-in types which are defined in the underlying format standard (e.g., HDF5 and XML built-in types) in order to leverage standard data validation software provided the equivalence is documented either in the product specification or the underlying format standard.

3.6 Recommended practices

3.6.1 Reviews of model elements and structure

Models should be frequently reviewed while under development, with reviews involving domain experts as well as information modeling experts.

3.6.2 Diagram layout

Common 'best practices' for layout of UML diagrams should be followed. In particular, diagram should not contain too many elements, should minimize line crossings, and use vertical layouts for hierarchies (or left-right horizontal layouts if a vertical layout does not work). Lines representing associations should minimize the use of curved segments.

3.6.3 Color coding of model elements

Color coding should be used to distinguish diagram elements for features, information types, enumerations and codelists, complex attributes, association classes, and constraints and notes. Abstract types should be indicated by darker shades.

Figure 3-1 and Figure 3-6 illustrate the use of color coding to depict different kinds of UML elements. Compare the shades of the non-abstract feature and information classes in these figures to the abstract feature and information classes in Figure 3-2.

S-100 departs from ISO TC211 recommendations for the use of black-and-white-only UML diagrams in order to distinguish between feature and information types (the concept of information type is unique to S-100) and since the ISO recommendation appears to be based on considerations for the production of commercially printed documents (as opposed to office laser or ink-jet printers or on-screen displays).

3.6.4 Documentation tables

This may be formatted like the UML schema documentation tables in S-100 or generated by the UML software. Whichever method is used, the documentation must document the classes, attributes, enumeration and codelist types, and asociations in the application schema, including names, definitions, multiplicities, data types, and roles.

3.6.5 Software support

Product specification development teams should use a sandbox tool like a wiki to work on data model.

Product specification development teams should use Enterprise Architect to develop the UML application schema(s). Other UML tools or special templates in off-the-shelf editors may also be used are likely to have minor differences in UML notations which will need to be adjusted or explained in the product specification. Even ordinary diagram editors can be used if necessary but are likely to be more time-consuming than UML software.

XML data including feature catalogues and metadata is easier to view in open-source or COTS XML software rather than ordinary text editors.

3.6.6 Identification of models

The identification of each application schema shall include a name and a version. If there is only one application schema in the product specification, this identification is implicit in the name and version of the product specification. Product specifications with more than one application schema must identify each, potentially by associating it with a scope.

4 IHO GI Registry

Procedures for registration are explained in S-99. It is recommended that at least one member of the project team or working group be a Submitting Organization. Submitting Organizations propose changes and additions to the contents of Registers. Submitting Organizations will normally represent a recognized body or stakeholder group (such as from government, industry, academia, and relevant user groups). Registered submitting organizations may submit proposals for consideration under any domain in a register. Stakeholders and any other interested parties who do not wish to enroll should submit proposals through an existing Submitting Organization.

To harmonize with other specifications, propose extensions to registry items where possible, e.g., propose generalization or specialization of an existing element, or additional values in an enumeration or codelist type. Restrictions of existing types can become new sub-types rather than changes to an already defined type; or it may suffice to define a constraint in the product specification.

5 Feature catalogue

The Feature Catalogue is an XML document which conforms to the S-100 XML Feature Catalogue Schema. Note, for Imagery and Gridded Data, a coverage is a feature type, and a product specification feature catalogue should define the attributes, coverage feature (with spatial primitive type ‘coverage’).

Feature catalogues should be documented by a text-based documentation of their contents, which should also be reviewed by the project team and responsible working group. For review purposes, this text-based documentation should be generated from the XML feature catalogue. The resultant text can be in Word or PDF, or another format preferred by the team.

6 Data transfer modes and packaging

Define mode of delivery – exchange set, message, service.

This needs to be done before metadata is specified because some metadata elements as well as the treatment of metadata (e.g., separate vs. embedded) depends on delivery mechanisms, constraints, and protocols.

Details of packaging and transfer content can be finalized in a later stage (see section 10).

7 Metadata

The minimum metadata requirements are set forth in Part 4 of S-100 (Appendix 4a-D for vector data, Parts 4b/8 for coverage data). Product Specification developers should consider whether the metadata elements listed in S-100 are relevant to the data product and which of them are appropriate for its allowed packaging and delivery methods. For relevant elements, define appropriate values and restrictions if necessary for the metadata elements listed in S-100 Appendix 4a-D or Parts 4b/8. Developers should note that Part 4b is quite skeletal in Edition 3.0.0 and the development team will need to use the underlying ISO standards and ISO metadata schemas.

If additional metadata elements are needed they should be documented in the product specification Metadata section and extensions to the standard metadata schemas developed using the standard ISO extension mechanism.

IHO metadata XML schemas for exchange catalogues and discovery metadata have been developed and are available at the IHO software distribution site (https://github.com/IHO-S100WG).

8 Define data encoding format

8.1 Selection of encoding format

The encoding format should be selected based on the type of product and other requirements, including production and processing. The characteristics of the three standard data formats included in S-100 Edition 3.0.0 are summarized below for convenience.

	ISO 8211	**GML**	**HDF5**
Type of product to which suited	Nautical charts and feature-heavy vector data	Nautical publications and information-heavy vector data; discrete weather information; small datasets such as marine safety information; data delivered via messages and web services	Coverage-based data
Generic data format	Yes	Yes	Yes
Data production complexity	Requires custom tools	Can be produced with a range of tools from text editors to custom apps and database SQL queries	Custom apps that use off-the-shelf libraries
Processing complexity	High	Low	High

Supporting off-the-shelf software	Not much	off-the-shelf viewers and server software; can be viewed with ordinary text editors	off-the-shelf viewer
Data volumes	Lower	Higher	Lower
Type of data	Vector	Vector; coverage schemas are defined in the GML specification but not used in S-100	Gridded
Supporting artifacts needed in product specifications	Feature catalogue	XML schemas for data validation; datasets can be processed by apps without XML schemas; self-documented format (tags indicate objects and attributes); Feature catalogue optional	Embedded object and attribute tags; feature catalogue optional(?)

While other formats than the three standard encodings are possible, the use of a non-standard format has the following implications:

- Loss of genericity, requiring special purpose development by implementers or conversion to a standard format with impacts on performance.
- Potential loss of compatibility with interoperability.

The question of appropriate formats for transactional, web-service, or message-based information has not been formally addressed in depth at this time, but product specifications needing such delivery modes should endeavor to use one of the standard formats in order to minimize implementation complexities.

8.2 Data format definition artifacts

Selection of the GML format will require definition of XSD files encapsulating the S-100 application schema as XSD files conforming to the GML specification (ISO 19136 and S-100 Part 10b). There must be a 1/1 mapping from the application schema to XML schema elements in the GML XSD. GML schemas. Schema developers should note that conformance to the GML specification requires conformance to the rules set forth in the GML specification (ISO 19136 / OGC 07-036), not merely XML-validation against the GML schemas.

S-100 Part 10b describes the S-100 GML profile. The XSD files for the S-100 GML profile are available at the IHO S-100 distribution site (https://github.com/IHO-S100WG). Previously defined GML schemas for other product specifications (e.g., S-122, S-123) will provide useful guidance and will be made available through the IHO product specification distribution site or GitHub distribution site.

Guidance for generic processing for GML datasets will be included in S-100 Edition 4.0.0.

9 Portrayal

Portrayal catalogues are necessary only for data products that are intended to be displayed graphically (as opposed to text or other processing).

S-100 Edition 3.0.0 defines an XSLT-based portrayal mechanism in Part 9. Edition 4.0.0 will incorporate the LUA scripting language as a second portrayal mechanism. S-100 will continue to include the existing XSLT mechanism.

The portrayal section of a product specification should include:

Pictorial representations of symbols and colors, accompanied by symbol specifications (the latter preferably in the form of machine-processable files as well as formal specifications in the text of the portrayal section).

Specifications and recommendations for the use of symbols by implementers, e.g., calculating orientation from attribute values and the use of thinning to reduce crowding of displays at small display scales, display of text accompanying symbols, masking of boundaries, etc.

An IHO portrayal catalogue builder is under development.

10 Data product delivery

10.1 Delivery content and structure

Define the content and structure of delivery packages: Exchange sets, messages, or web services.

Exchange set structure should be defined, either by using the structural diagram from S-100 (reproduced below) as is, or restricting the allowed components, or defining extensions of the individual components. If there is internal structure in the exchange sets (e.g., folders and sub-folders), determine the required layout and naming conventions. Determine how the exchange set as a whole is packaged (e.g., ordinary folders, zip file, etc.).

For message and web service modes of delivery, specify the container format, packaging, and specify the transfer protocol, e.g., REST (Representational state transfer), SOAP (Simple Object Access Protocol), WSDL (Web Services Delivery Language), WFS (Web Feature Service).

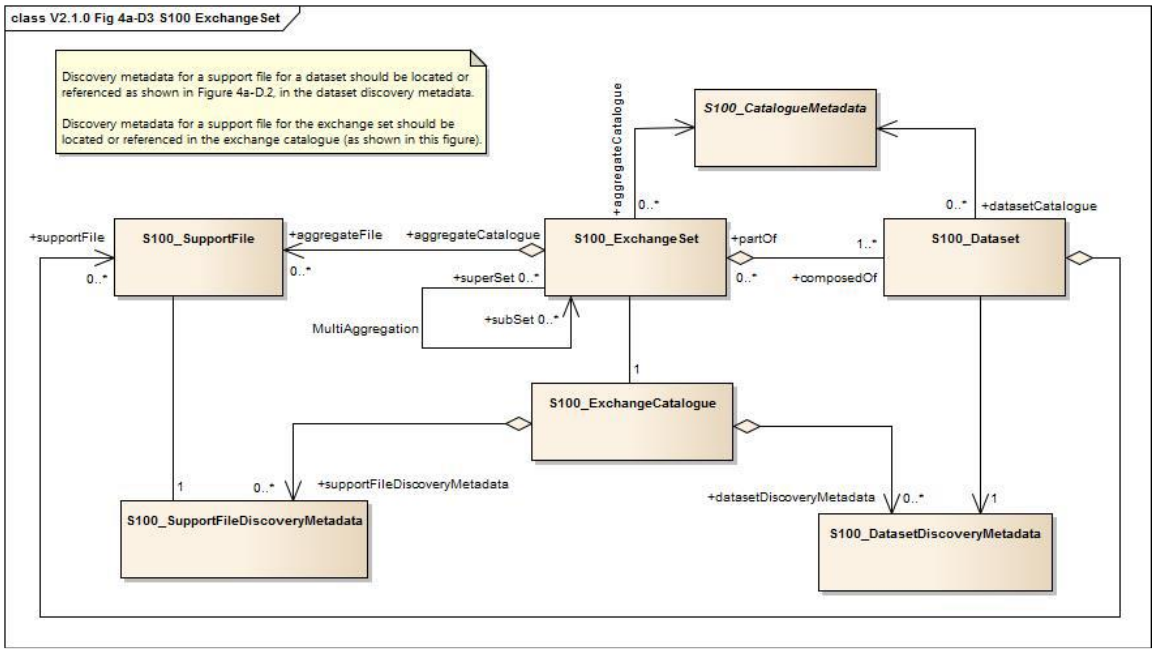


Figure 10-1. Prototype exchange set structure

Determine what naming conventions, if any must apply to individual components of the exchange set - dataset files, exchange sets, and support files. Naming conventions are generally not needed for message-based or service modes of data delivery, but a unique identifier will generally be needed for each message or service transaction.

10.2 Dataset updates

10.2.1 General considerations for updates

Define the conditions and mechanisms for data updates. specifically:

- update cycles – how frequently data must be updated, whether updates are issued on a regular cycle, as-needed, or a combination of both;
- how long each dataset is valid and how to validity periods will be indicated
- mechanisms for cancellation, replacement, and reissue of datasets
- metadata for updates
- types of updates – whether the data product requires incremental, whole-dataset replacement, irregular, or cumulative updates.
- Criteria for determining when datasets must be replaced by new datasets, superseded, reissued, updated, and cancelled.

10.2.2 Format-specific update considerations

ISO 8211 – Part 10a describes update mechanisms for ISO 8211 datasets. Each specification must define the structure of update datasets.

GML: Replacement of whole objects is the recommended method, but there are XML specifications that allow update of individual attributes. (Details are currently in a working paper, “Updating GML Datasets” - TSM5-4.11.)

HDF5: Feature (coverage) can be updated in its entirety or in part (the update can be a sub-grid).

10.3 Supporting information

Describe how any auxiliary content is delivered either with or as an adjunct to data. S-100 provides for ‘support files’ to be included in exchange sets. Support files can be graphic or text information files referenced by dataset objects, or other files such as dictionaries and catalogues (including feature or portrayal catalogues). Define allowed file formats and naming conventions for support files.

Note that since feature and portrayal catalogues are shared by all datasets conforming to a specific version of a product specification, it will generally be more efficient to deliver feature and portrayal catalogues once rather than with every exchange set. Possible methods of such special deliveries have not been standardized yet and are left to product specification developers, but may include special exchange sets distributed through the usual channels or a centralized means such as publication on a web server. If a central distribution mechanism is adopted consideration must be given to the possible needs of end users who have only infrequent or no access to the distribution hub (e.g., low-bandwidth or no Internet access).

11 Validation checks/data quality

At least two types of validation checks are needed:

- Dataset validation checks, for individual datasets. These checks operate on individual objects in datasets and on individual datasets as a whole. They should check the integrity of individual objects in the dataset (spatial, feature, and information types), associations between objects in the dataset, any embedded metadata or header information in the dataset, and support files referenced in the dataset.

- Package validation checks, for verifying the structure and content of packages (e.g., exchange sets) and accompanying metadata.

11.1 Validation checks for datasets

Validation tests for datasets should cover:

1) Completeness, including population of attributes and presence of required information, complex attributes without sub-attributes, etc.
2) Logical consistency – e.g., missing association targets.
3) Spatial consistency – e.g., topological sanity checks for non-crossing external boundaries, excessive vertex density in lines, etc.
4) Positional accuracy.
5) In principle, they can also cover temporal accuracy, for time sensitive data, e.g. forecast data for water level and currents. Forecast data is generally computed and therefore may be accurate to the forecast model, but when compared with observed data, there is often a time shift one way or another. Maybe there are some expressions of what uncertainty is expected between forecast and observations. There is currently no use case for temporal accuracy.
6) Thematic accuracy, such as attribute values that are consistent with any other related attributes and within allowed ranges or sets.
7) References to support files.
8) Other requirements specific to the product – e.g., encryption, signatures, etc.

11.2 Validation checks for packages

Validation checks for packages should cover:

Package completeness – whether all required components are included, including datasets, support files, metadata, and appropriate catalogues (e.g., exchange set catalogues, feature catalogues, portrayal catalogues). Note that the product specification must indicate which catalogues are appropriate to the delivery method – for example, message-based delivery methods may not include catalogues in the delivery packages.

Package container format and structure – whether the package is in the approved container format (e.g., ,), and whether appropriate encryption and signatures have been applied at the container level. Examples of package validation checks are:

- Assuming the product specification specified delivery as zip files, is the container a zip file of the appropriate type?
- If the package is arranged in a directory (folder) structure, are the structure and names of directories (folders) as required in the product specification?

11.3 Common validation checks

Given that some features, information types, and application schema constructs are used in multiple products, there will be validation checks in common with existing product specifications and any such related product specifications should be consulted for validation checks. Spatial consistency checks in particular, as well as consistency checks related to meta-features, can be expected to be in common with several data specifications.

Spatial operations used in validation checks must be the operations defined in IHO ENC Validation Checks (S-58 6.0.0 or its successor).

11.4 Validation checks for base versus update datasets

If the product specification defines an update dataset format, the validation checks developed for new datasets should be reviewed for their applicability to update dataset formats.

12 Reference systems

The preferred coordinate reference system is EPSG 4326 which is based on the WGS 84 horizontal datum.

Horizontal datum will normally be referenced by giving its code in the EPSG register. If the coordinate reference system is not one of the coordinate reference systems in the EPSG register, a datum may be specified in a support file as described in S-100 Part 6, should there be a use case in the scope of the data product.

A set of vertical datums is listed in S-100 Part 4a (S100_VerticalAndSoundingDatum). Specification developers are encouraged to adopt the S-101 ENC datum (either standard or local S-101 ENC datum) as a common vertical reference datum if possible. If a need for an additional datum is identified it should be proposed as a revision to S-100.

S-100 includes ‘local datum’ as an allowed value for datum attributes, but this is of limited utility even within a data product. (S-101 cites an example of use of local datum in a non-tidal basin, where depths may refer to a sounding datum different from that in open waters. If this area is navigable at the maximum display scale of the ENC data, the value of this datum must be encoded using attribute vertical datum = 24 (local datum), in a meta-feature co-incident with the area covered by the dock.) Data conversion to/from unspecified local datums would be problematic.

13 Sample data / test datasets

Test data should be created in sufficient quantity to validate the main characteristics of the application schema. Specifically, the first test dataset should contain:

- At least one instance of each kind of feature and information type.

- A representative set of feature and information associations, preferably at least one instance of each named association. (It is not necessary to create an instance for each and every pair of object classes which may be linked by an association.)
- At least one instance of each meta-feature and data quality feature.

At least one update dataset should also be prepared, to validate the update dataset format and packaging.

Additional test datasets should test typical data volumes, representative data capture problems, and error cases.

If delivery is supposed to be in the form of exchange sets, the test datasets should be packaged as complete exchange sets, including sample metadata files. Sample packages for other forms of delivery (transactions, messages, web services, etc.) should be emulated as realistically as is practical at this stage – i.e., setting up a web server, service broker, etc. for web services should be done if doing so it practical but is not an essential requirement (it can and should be done as part of the testing stage).

14 Preparing for interoperability

Determine which if any product groups in interoperability catalogues are supplemented or enhanced by the data product.

Determine whether and how the IHO interoperability catalogue will be affected by the new product, including updates to display priorities, interleaving, predefined combinations, and other interoperability rules and operations.

Revise portrayal catalogues upon recommendation by the IHO interoperability team.

[To be expanded as the interoperability specification matures.]

15 Testing and feedback

A formal test plan should be prepared, including test cases.

More details about development and approval of a test plan will be added later.

16 Work processes

16.1 Registration and getting an S- number

Apply for a product specification number – this should ordinarily be assigned when the development project is approved during the initiation stage.

Registration of product specification artifacts in the GI registry should be done in accordance with the procedures established in IHO publication S-99.

16.2 Project teams

Project teams should involve domain specialists, information modeling specialists, and representatives of OEM/developer communities.

16.3 Iterative refinement as a development process

Development should plan for iterative refinement, with the following being reviewed at the indicated stages of development:

- Initial application schema. Reviews of subsequent revisions can be rolled into the reviews of the main product specification document.
- First drafts of main document of product specification and DCEG. Subsequent revisions should be reviewed as ready.
- Feature catalogue, Data format and sample datasets should be checked after each major revision to the application schema and feature catalogue.
- Portrayal catalogue – first draft and and significant revisions
- Other artifacts or components, such as validation tests – when substantially completed, and after revisions to due to the application schema, feature catalogue, or data format.
- The product specification as a whole – after the complete package is ready.

Reviews of different components can obviously be combined to fit the development schedule or workgroup meeting schedules. The application schema, main product specification, and DCEG should be expected to undergo multiple reviews during different stages of development.

Reviews during the development stage should be requested from:

- Project team – initial reviews
- Technical group(s) sponsoring the specification, as well as related technical groups – after some stability has been achieved.
- Developers, implementers, and OEMs – formal reviews after initial reviews and stabilization in the project team and sponsoring technical group. Note that individual implementer/OEM/developers should ideally be involved from the earliest stages of development if available.

Reviews should be completeness, correctness, ability to capture and express the domain, performance/efficiency, as well as conformance reviews for verifying compliance to S-100 and underlying standards.

Stakeholder reviews should be requested as the specification matures, and should involve:

- Producers
- Developers and OEMs

Subsequent stages should involve users and user testing, and preparation of an impact study.

Test development and testing should commence upon the feature catalogue and data format achieving reasonable stability, presumably after one or two cycles of review.

Final assessment will be at the HSSC level for IHO or equivalent for other organizations.

Pre-publication review by IHO or other publishing organization prior to publication, to check production issues.

16.4 Maintenance of product specifications

Clarifications, corrections, and revisions should be designated in accordance with the same criteria used for S-100 described in S-100 3.0.0 Section 12.2 (Maintenance Procedures).

Specifications should undergo periodic review. A two-year review period is suggested for new specifications, which may be increased to five years after the specification reaches maturity.