International
Virtual
Observatory
Alliance

Model Instances in Votables
Version 1.0

IVOA Working Draft 2020-08-18

Working group
DM
This version
http://www.ivoa.net/documents/vodml-instance-vot/20200818
Latest version
http://www.ivoa.net/documents/vodml-instance-vot
Previous versions
This is the first public release
Author(s)
François Bonnarel, Gilles Landais, Laurent Michel, Jesus Salgado
Editor(s)
Laurent Michel

Abstract

Vodml-instance-vot proposes a syntax to map VOTable data on any model serialized in VO-DML. Vodml-instance-vot annotations are grouped in a single XML block located in the VOTable head. The annotation block allows to easily reconstruct the model structure. It designed in a way that the block can be reused on different data sets in order to facilitate the annotation process. Vodml-instance-vot is enable to join data from different tables

Status of this document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/documents/.

Contents

Acknowledgments

CDS/TDIG/SourceDM contributors

Conformance-related definitions

The words “MUST”, “SHALL”, “SHOULD”, “MAY”, “RECOMMENDED”, and “OPTIONAL” (in upper or lower case) used in this document are to be interpreted as described in IETF standard RFC2119 (**?**).

The *Virtual Observatory (VO)* is a general term for a collection of federated resources that can be used to conduct astronomical research, education, and outreach. The International Virtual Observatory Alliance (IVOA) is a global collaboration of separately funded projects to develop standards and infrastructure that enable VO applications.

1 Introduction

The first purpose of a model is to provide, for a particular domain, a formal description of the relevant quantities and of the way they are connected together . This documentary role facilitates the communication between the stack-holders and thus the design of interoperability protocols.

At data level, interoperability consists in arranging searched data in a way that a client can understand them without taking care of their origin. So that, the same code can process and compare data coming from different sources. That way to arrange data is given by the model.

This is not done by default with VOtables because VOTables are containers. The VOTable schema cannot say how data are mapped on a given model or whether they match any model at all. This is not an issue for simple protocol responses (ref) because the VOTable structure is defined by the protocol itself but this is however a big issue for VOTables containing native data such as Vizier or TAP query responses.

The challenge here is to bind native data with a given model in a way that a model aware software can see them as model instances while maintaining the possibility to access them in their original forms.

This is partially done with UTypes which may connect FIELDs or PARAMs with model leaves in the case of simple tree-views of the model. Unfortunately, there is no more unique way to build and parse UTypes in the context of more complex the models. This occurs when e.g the same class is used in different location of the model or when the model contains loops. It is also not possible to refer data from different tables with UTypes.

The landscape has dramatically changed in 2016 when VODML (ref) became a recommendation. VODML is a meta-model that gives a standard way to design VO models and to make them machine-readable. In VODML, model leaves are no longer identified by a simple string like UTypes do but by a certain role played in a given location in the model hierarchy. The consequence is that any annotation mechanism based on VODML must preserve

the model hierarchy to save the role played by any components. In this context, it might be easy to re-construct model instances from the annotations.

The main concept of VODML mapping is to insert on top of the VOTable an XML block following the model structure and containing references to the actual data. In such a way that in order to build a model instance, a model-aware client only has to make a copy of that structure and to resolve the references. More generic model-unaware clients can just ignore the mapping block. This approach, has been proposed by (GL and OL).

We have tested the syntax originally proposed and it turned out that it was not well suited for archival data or for TAP responses where the annotation process must be automated as much as possible (entirely for TAP).

Vodml-instance-vot is based on the same principles as the original proposal but with a particular attention given to the annotation of archival data by keeping focused on both client needs and easiness of the annotation process. This requires the syntax to be as simple as possible and as flexible as possible to be usable with a wide range of data sets. With Vodml-instance-vot, the model hierarchy is built upon attributes, tuples and arrays (ATTRIBUTE, INSTANCE and COLLECTIONS). Some other elements have been added to guide the parser (FILTER, TABLE_ROW_TEMPLATE ...)

The connections with the data are setup by XML element attributes, so that the mapping structure just depends on the model but not on the mapped data.

These ideas were first tested in the framework of the TDIG on VOTABLEs containing time series provided by different missions such as Gaia or ZWICKI. Then, the syntax has been refined to be used to validate the Mango model on real data.

1.1 Role within the VO Architecture

Fig. 1 shows the role this document plays within the IVOA architecture (**?**).

???? and so on, LaTeX as you know and love it. ????

2 Use Cases and Requirement

2.1 Use Cases

2.1.1 Client Side

The mapping is self consistent. The role of the mapping is to give the client all information it needs to reconstruct a datastructure similar to a set of instances of the the model. A model-aware client must be able to do this without implementing any code specific to any particular model. The mapping syntax is independant of the model. The structure of the mapped

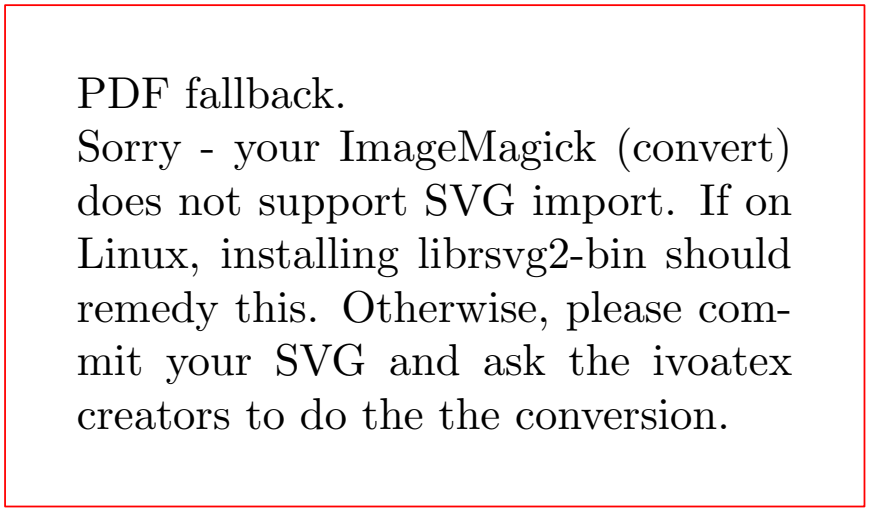

Figure 1: Architecture diagram for this document

model is given by the arrangement of the mapping elements, not by the elements themselves

Identifying the nature of the content of a VOTable: A client can get it by a quick look at the annotation block.

Measurement discovery: A client wants to discover whether a VOTable contains some peculiar measurements (position, velocity...). The annotation block allows it to get an answer by a quick parsing of the annotation block.

Data set comparison: A client wants to compare different data set (Xmatch, plot). The annotation provides a homogeneous data representation of these data that allows to put them together in a consistent way.

Data set export: A client wants to export (e.g. with SAMP) a model instance in a convenient format (e.g. json). The JSON model instance can be buit from the annotation block and exported to a another party.

2.1.2 Server Side

The server use cases are to make possible the realization of those of the clients for a reasonable cost. The annotation process can represent a significant extra work for the curator team that must be limited as much as possible. To do so the mapping syntax is designed to facilitate the use of templates and components.

3 server types that could annotate data have been identified:

1. Mission data provider: the data annotation can be set once forever for each data product at the design phase.

2. Archival data provider: The data annotation must be done for each archived datat set. The curator has a little control, on the data format and he/she has to do his best to match data with the model(s)

3. TAP data provider: In case of TAP services, the annotation process is in charge of the TAP server that must dynamically match the queried data with the model quantities for each specific query.

The goal of this version of the specification is to support requirements 1) and 2) with a special attention to make 2) easier. Support of requirement 3) is still an experimental feature at the time this specification is written.

2.2 Requirements

- Shy Annotations: The data mapping must not affect the operation of existing clients.
- Faithful Annotations: The structure of the annotation must be faithful to any VODML compliant model.
- Different Usage Levels
  - The data mapping must be easily ignored by the client.
  - The data mapping must allow clients to easily detect the model on which data are mapped.
  - The data mapping must allow clients to easily get the general metadata (e.g. coordinate systems).
  - The data mapping must allow clients to get full model instances for each table row.
- Easy to Build
  - The mapping structure must be independent of the data structure.
  - The mapping syntax should make easy the building of both mapping components and templates.
- Complex Data Mapping
  - The mapping syntax must support to retrieve data spread over several tables.
  - The mapping syntax must be able to filter data rows which are part of a specific instance.
  - The mapping syntax must be able to group data rows in an set of instances.

3 Syntax

The syntax specified in this standard gives rules to build consistent annotations for any model. However, it do not prevent to do foolish things, in the same way that a programming language grammar does not protect people against writing irrelevant software. In the follwoing examples, attribute values do not refer toi any particular model or VOTable. They have been to help reader to figure out their meanings.

3.1 Mapping Block Structure

The rules below must be updated accordingly to the XML schema .. in progress....

The mapping block is outside of the data tables. Its scope is the whole VOTable. Its stucture is given below.

```
<VODML>
   <MODELS> ... </MODELS>
   <GLOBALS> ... </GLOBALS>

   <TEMPLATES tableref=...> ... </TEMPLATES tableref=...>
   <TEMPLATES tableref=...> ... </TEMPLATES tableref=...>
   ...
</VODML>
```

Listing 1: INSTANCE bloc example

The mapping construction rules are the same whatever the model or the data layout are.

- The mapping is located in a `<VODML>` block, child of `<VOTABLE>`.
- The mapping elements reflect the model structure.
- The `<VODML>` block starts with a list of implemented models.
- There is one `<TEMPLATES>` per mapped `<TABLE>`.
- There is one `<GLOBALS>` block containing data shared by the whole mapping.

3.2 MODELS

The MODELS blocks contains the list of the models mapped in the block. It can be left empty.

- Models referenced in MODELS are not necessary VO standards, but they must be accessible by a VODML URI.
- It is to be noted that the mapping syntax allows to reconstruct the model structure without parsing any VODML model serialization.

```
<MODELS>
    <MODEL>
        <NAME>ivoa</NAME>
        <URL>http://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml
        </URL>
    </MODEL>
    <MODEL>
        <NAME>coords</NAME>
        <URL>http://www.ivoa.net/xml/STC_coords-v1.0.vo-dml.xml
    </URL>
    </MODEL>
    <MODEL>
        <NAME>meas</NAME>
        <URL>http://www.ivoa.net/xml/STC_meas-v1.0.vo-dml.xml
        </URL>
    </MODEL>
</MODELS>
```

Listing 2: GLOBALS block example

MODELS, MODEL, NAME and URL have no attributes.

3.3 GLOBALS

Contains INSTANCEs that can be used everywhere in the VODML.

- INSTANCEs children of GLOBALS should have an @ID attribute so that they can be referenced from other instances.
- The role of the GLOBALSs children (INSTANCE by construction) must be ignored although being mandatory.
- References within GLOBALSs sub-elements to VOTable data (FIELD ot PARAM) must be searched in all tables. They must be resolved by the first occurence matching the reference found.
- GLOBALS has no attributes.

```
<GLOBALS>
  <INSTANCE ID="SpaceCoordFrame" >
    <INSTANCE dmrole="coords:SpaceFrame.refPosition"
              dmtype="coords:StdRefLocation">
       <ATTRIBUTE dmrole="coords:StdRefLocation.position"
                  dmtype="ivoa:string" value="NoSet"/>
    </INSTANCE>
    <ATTRIBUTE dmrole="coords:SpaceFrame.spaceRefFrame"
               dmtype="ivoa:string" value="ICRS"/>
    <ATTRIBUTE dmrole="coords:SpaceFrame.equinox"
               dmtype="coords:Epoch" value="NoSet"/>
  </INSTANCE>
</GLOBALS>
```

Listing 3: GLOBALS block example

Child	Role
INSTANCE	Model instances with a scope covering the whole VOTable .

Table 1: Allowed GLOBALS children

3.4 TEMPLATES

TEMPLATE blocks contain the mapping statements of the data contained in one TABLE .

- There is one TEMPLATE block for each mapped TABLE in the VOTAble
- A TABLE cannot be referenced by more than one TEMPLATE.
- There is TEMPLATE reference (@tableref) must be first resolved against the TABLE identifier (@ID).

```
<TEMPLATES tableref="OtherResults">
    <COLLECTION dmrole="test:detections">
        <TABLE_ROW_TEMPLATE>
            <INSTANCE dmrole="test:detection" dmtype="test:Detection">
               <ATTRIBUTE dmrole="test:detection.num" dmtype="ivoa:real"
                          ref="_num_148" />
               <ATTRIBUTE dmrole="test:detection.id" dmtype="ivoa:real"
                          ref="_foreign" />
            </INSTANCE>
        </TABLE_ROW_TEMPLATE>
    </COLLECTION>
</TEMPLATES>
```

Listing 4: GLOBALS block example

Child	Role
INSTANCE	The table data are mapped on these instances.
TABLE_ROW_TEMPLATE	There is one instance per table row. The structure of those instance is given by the TABLE_ROW_TEMPLATE children
COLLECTION	The table data are mapped on an instance list

Table 2: Allowed TEMPLATES children

Attribute	Role
@tableref	The @ID or the @name of the mapped table

Table 3: TEMPLATES attributes

@tableref	Pattern
MAND	Always mandatory

Table 4: Valid attribute patterns for `TEMPLATES`

3.5 INSTANCE

Mapping for either object types or a datatype instances.

```
<INSTANCE dmrole="ds:Dataset.dataID" dmtype="ds:DataID" ID="_ds_">
    <ATTRIBUTE dmrole="ds:DataID.title" value="Gaia TS Mapping Test" />
    <ATTRIBUTE dmrole="ds:DataID.datasetID" value="ivoa://gaia/ts/12345" />
    <ATTRIBUTE dmrole="ds:DataID.creatorDID" value="ivoa://esa/gaia/" />
    <ATTRIBUTE dmrole="ds:DataID.version" value="0.0" />
    <ATTRIBUTE dmrole="ds:DataID.date" value="2018:11:11" />
    <ATTRIBUTE dmrole="ds:DataID.creationType" value="LiteMappingTest" />
    <INSTANCE dmrole="ds:DataID.creator" dmtype="ds:Creator">
        <INSTANCE dmrole="ds:Role.party" dmtype="ds:party.Individual">
            <ATTRIBUTE dmrole="ds:Party.name" value="VODML-Team" />
      </INSTANCE>
    </INSTANCE>
</INSTANCE>
```

Listing 5: INSTANCE block example

Child	Role
INSTANCE	Another embedded instance .
ATTRIBUTE	Primitive attribute .
COLLECTION	Set of items

Table 5: Supported INSTANCE children

Attribute	Role
@dmrole	VODML role of the instance.
@dmtype	VODML type of the instance. Must never be empty
@dmref	Reference to another instance in the mapping block. Must never be empty
@ID	Unique identifier of the instance. Must never be empty

Table 6: INSTANCE attributes

@dmrole	@dmref	@dmtype	@ID	Pattern
MAND		MAND	OPT	Instance of a certain type playing a certain role. The role may be left empty for child instances of `GLOBALS`
MAND	MAND			Reference to another instance. No allowed children in this case.

Table 7: Valid attribute patterns for `INSTANCE`

3.6 ATTRIBUTE

Mapping statement for primitive attributes.

- `ATTRIBUTE`s are the model leaves that point onto real data or are set with literals
- `ATTRIBUTE`s have no children.

```
<INSTANCE dmrole="model:value.example" dmtype="model:value.Example">
    <ATTRIBUTE dmrole="model:preset.value" value="Preset Value" />
    <ATTRIBUTE dmrole="model:ref.value" ref="fieldID" />
    <ATTRIBUTE dmrole="model:refofpreset.value"
               value="Preset Value" ref="fieldID" />
</INSTANCE>
```

Listing 6: ATTRIBUTE examples

Attribute	Role
@dmrole	VODML role of the instance attribute.
@dmtype	VODML type of the instance attribute.
@value	Literal value of the instance attribute. If `ATTRIBUTE` has also a `@ref`, `@ref` MUST be resolved first. `ATTRIBUTE` MUST be taken when `@ref` cannot be resolved
@ref	Reference of the data element (`FIELD` or `PARAM`). MUST refer to an element of the `TABLE` referenced by the current `TEMPLATE` The client MUST first look for a `FIELD` matching `@ref`. In case of failure, it MUST look for a `PARAM`

Table 8: `ATTRIBUTE` attributes

@dmrole	@dmtype	@ref	@value	Pattern
MAND	MAND	MAND	OPT	The instance attribute must take the value pointed by `@ref`. If the reference cannot be resolved, the attribute takes the value of `@val` if present. It is considered as not set otherwise.
MAND	MAND		MAND	The attribute takes the value of `@val`.

Table 9: Valid attribute patterns for `ATTRIBUTE`

3.7 COLLECTION

Mapping statement fort sets of either instances or collections.

- A `COLLECTION` can contain a fixed set of instances or collections. In this case, each element must be mapped individually.
- A `COLLECTION` can contain an unbounded set of instances, one per selected table row. In this case, all items have the same type and thus mapping. They can come from the local table or from a joint table.

The example below show up a a fixed length `COLLECTION`.

```
<TEMPLATES tableref="Results">
    <COLLECTION dmrole="meas:Measure.errors" size="2">
        <INSTANCE dmref="globald=s_stat_error" />
        <INSTANCE dmref="globald=s_sys_error" />
    </COLLECTION>
</TEMPLATES>
```

Listing 7: `COLLECTION` example

Child	Role
`INSTANCE`	Collection item. A collection can contain multiple instances
`COLLECTION`	Collection item. A collection can contain multiple instances
`TABLE_ROW_TEMPLATE`	The collection is populated with with one instance per row of the current table. When present, this element must be the only child.
`JOIN`	The collection is populated with data read in another table. The primary join key must be one of the host instance attributes.This element must be only child.

Table 10: Valid `COLLECTION` children

Attribute	Role
@dmrole	Role played by the collection (VODML relation name usually). Cannot be empty.
@size	Collection size. This attribute is not necessary to parse the mapping block.

Table 11: Valid attributes for `COLLECTION`

@dmrole	@size	Role
MAND	OPT	Role played by the collection (VODML relation name usually). Cannot be empty

Table 12: Valid attribute patterns for `COLLECTION`

3.8 TABLE_ROW_TEMPLATE

This element indicates that one element must be added to the host `COLLECTION` for each table row.

- The mapping of the row is given by the `INSTANCE` child.
- We must map one and only one `INSTANCE` per row. This makes senses since collection elements cannot be made with more than one instance.
- `TABLE_ROW_TEMPLATE` has no attributes.

```
<TEMPLATES tableref="Results">
    <COLLECTION dmrole="test:detections">
        <TABLE_ROW_TEMPLATE>
            <INSTANCE dmtype="test:Detection">
                <ATTRIBUTE dmrole="test:detection.num" dmtype="ivoa:real"
                           ref="_num_148" />
                <ATTRIBUTE dmrole="test:detection.id" dmtype="ivoa:real"
                           ref="_num_149" />
           </INSTANCE>
        </TABLE_ROW_TEMPLATE>
    </COLLECTION>
</TEMPLATES>
```

Listing 8: TABLE_ROW_TEMPLATE examples

Child	Role
INSTANCE	Mapping to be applied to table row

Table 13: Supported `TABLE_ROW_TEMPLATE` children

3.9 FILTER

This element filters the table rows that are to mapped.

- The filtering condition is based on the equality of a column value with the filter value.
- The mapping specification does not specify the way to deal with data types.

In the example below::

- The light curve will be populated with table rows mapped by the `INSTANCE` of type `test:photometric.point`
- Each of these rows must have the value of the column `phot_filter_name` equals to `G`.

```
<COLLECTION dmrole="test.lightcurve">
    <TABLE_ROW_TEMPLATE>
        <FILTER ref="phot_filter_name" value="G"/>
            <INSTANCE dmtype="test:photometric.point">
                  <ATTRIBUTE dmrole="test:photometric.point.time"
                             dmtype="ivoa:real" ref="_num_148" />
                  <ATTRIBUTE dmrole="test:photometric.point.mag"
                             dmtype="ivoa:real" ref="_num_149" />
            </INSTANCE>
       </FILTER>
    </TABLE_ROW_TEMPLATE>
</COLLECTION>
```

Listing 9: FILTER examples

Child	Role
`INSTANCE`	Mapping to be applied to table rows matching the filter

Table 14: Valid `FILTER` children

Attribute	Role
`@ref`	Identifier of the column on which the filtering criteria must be applied
`@value`	Literal value that is used as filtering criteria

Table 15: `FILTER` attribute

@ref	**@value**	**Role**
MAND	MAND	All attributes must be set in any case

Table 16: Valid `FILTER` attribute pattern

3.10 JOIN

This element populates the host collection with data taken out from a foreign table and matching the join criteria.

- Each matching row of the foreign table is mapped as one `INSTANCE` of type `test:Detection` .
- Self-joins on the local table are allowed.
- The join criteria is based on the equality of the column values. The mapping specification does not specify the way to deal with data types.

```
<TABLE_ROW_TEMPLATE>
    <INSTANCE dmrole="primary:point" dmtype="Point">
        <ATTRIBUTE dmrole="test:detection.num" dmtype="ivoa:real"
                   ref="_poserr_148" />
        <COLLECTION dmrole="test.detections">
            <JOIN tableref="OtherResults" primary="_poserr_148"
                  foreign="_foreign">
                <INSTANCE dmtype="test:Detection">
                    <ATTRIBUTE dmrole="test:detection.num"
                               dmtype="ivoa:real" ref="_num_148" />
                   <ATTRIBUTE dmrole="test:detection.id"
                               dmtype="ivoa:real" ref="_foreign" />
                </INSTANCE>
            </JOIN>
        </COLLECTION>
    </INSTANCE>
</TABLE_ROW_TEMPLATE>
```

Listing 10: JOIN example

Child	Role
`INSTANCE`	Mapping to be applied to the matching rows.

Table 17: Supported `JOIN` children

Attribute	Role
`@primary`	Column name of the primary table used by the join
`@foreign`	Column name of the foreign table used by the join
`@tableref`	ID or name of the foreign table

Table 18: `JOIN` attributes

@primary	**@foreign**	**@tableref**	**Role**
MAND	MAND	MAND	All attributes must be set in any case

Table 19: Valid `JOIN` attribute pattern

3.11 GROUPBY

This element aggregates host table rows in groups which elements have all the same value for a given column.

- Each matching row is mapped as one instance of the `INSTANCE` child.

In the example below:

- The collection with `@dmrole=test.lightcurves` will be populated with a set of collections.
- Each of these sub-collections is populated with set of instances mapped by the `INSTANCE` of `test:photometric.point` type.
- All INSTANCEs are built wiith rows having all the same values for the column `source_name`

```
<COLLECTION dmrole="test.lightcurves">
    <GROUPBY ref="filter_name" dmrole="test.lightcurve">
        <INSTANCE dmtype="test:photometric.point">
                  <ATTRIBUTE dmrole="test:photometric.point.time"
                             dmtype="ivoa:real" ref="_num_148" />
                  <ATTRIBUTE dmrole="test:photometric.point.mag"
                             dmtype="ivoa:real" ref="_num_149" />
    </GROUPBY>
</COLLECTION>
```

Listing 11: GROUPBY examples

Child	Role
`INSTANCE`	Mapping to be applied to the matching rows.

Table 20: Valid `GROUPBY` children

Attribute	Role
`@ref`	Identifier of the column used for the grouping
`@dmrole`	Role of the grouped sub-collections

Table 21: `GROUPBY` attributes

@ref	**@dmrole**	**Role**
MAND	MAND	Must be set with non empty values

Table 22: Valid `GROUPBY` attribute pattern

3.12 Shortcuts

VODML encourgages people to use the `ivoa` model for the primitive types. Some of these types have a complex structures that associate units with values. This is the case for the types derived from `ivoa:Quantity` (`ivoa:RealQuantity` and `ivoa:IntegerQuantity`). The XML snippet below shows the regular mapping for a real quantity..

```
<INSTANCE dmrole="coords:PhysicalCoordinate.cval"
dmtype="ivoa:RealQuantity">
    <ATTRIBUTE dmrole="ivoa:RealQuantity.value" dmtype="ivoa:real"
                    ref="col_id" />
    <ATTRIBUTE dmrole="ivoa:RealQuantity.unit" dmtype="ivoa:Unit"
                    value="m/sec" />
</INSTANCE>
```

Listing 12: ivoa:RealQuantity example

This block maps a structure that is part of the VODML standards, therefore we can alias it with a compact element.

3.12.1 SC_REALQUANTITY

Shortcut for `ivoa:RealQuantity` class.

- Can only be used within an INSTANCE
- Using shorcuts requires units to be literals
- Both `@ref` and `@value` attribute work the same way as with `ATTRIBUTE`
- No `@dmtype`, it is set as ivoa:RealInteger by construction

```
<SC_REALQUANTITY dmrole="coords:PhysicalCoordinate.cval"
          ref="col_id" value="0.0" unit="m/sec" />
```

Listing 13: `ivoa:RealQuantity` example

3.12.2 SC_INTQUANTITY

Shortcut for `ivoa:IntegerQuantity` class.

- Can only be used within an INSTANCE
- Using shorcuts requires units to be literals
- Both `@ref` and `@value` attribute work the same way as with `ATTRIBUTE`.
- No `@dmtype`, it is set as ivoa:RealInteger by construction

```
<SC_INTQUANTITY dmrole="coords:PhysicalCoordinate.cval"
        ref="col_id" value="0" unit="m/sec" />
```

Listing 14: `ivoa:IntegerQuantity` example

A Changes from Previous Versions

No previous versions yet.