IMS Learning Design UML Models

IMS Learning Design UML models

Table of contents

Introduction

In order to model the DTD binding in UML we have used a method suggested by Rational in one of their white papers on this issue[1]. This method maps the constructs typically found in a standard manner to UML classes. This way it is possible to automatically derive a DTD or schema from the UML that compliant with the original. For this purpose some stereotypes have been defined that should be applied to every class in the diagrams. The following table gives an overview.

Stereotype	UML Element	Description
Element	Class	Represents a DTD or schema ELEMENT with either a element-only or mixed content model
ElementANY	Class	Represents a DTD or schema ELEMENT with an open content model, i.e. this DTD or schema does not specify the content model for this element.
ElementEMPTY	Class	Represents a DTD or schema ELEMENT with an empty content model (may still have attributes, simply no child elements).
ElementPCDATA	Class	Represents a DTD or schema ELEMENT with no child elements, simply text (may still have attributes, simply no child elements).
Group	Class	Used when multiple occurrences of the same element are possible (this could be either a sequence or selection).
ChoiceGroup	Class	Represents a group in a DTD or schema, with a choice of elements.
SequenceGroup	Class	Represents an ordered group of elements in a DTD or schema.
Type	Class	Represents a DTD parameter entity, which can be represented as a ComplexType or Group in a schema.

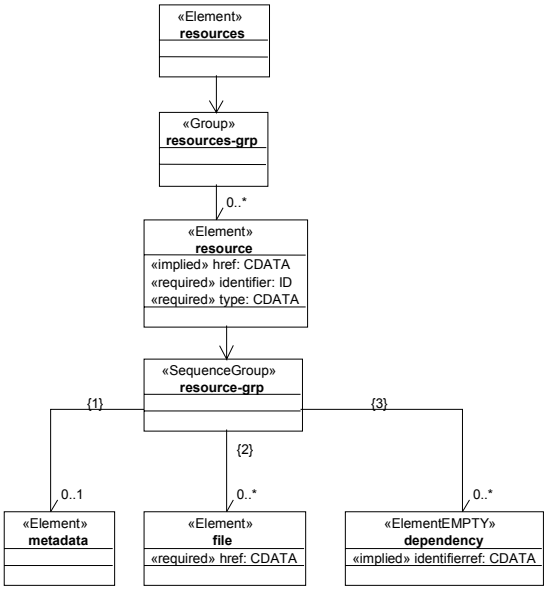
The following rules will be applied:

- Leaf nodes are always of stereotype «ElementPCDATA», «ElementANY» or «ElementEMPTY».
- If a node wraps (encloses) other nodes then this node is of type «Element». An association is created to a class which has one of the three following stereotypes: «SequenceGroup», «ChoiceGroup» or «Group». The class name for this class is derived from the referring class name with the extension of '-grp'. Obviously «SequenceGroup» and «ChoiceGroup» are used when a sequence or a selection of elements was enclosed by the referring element.
- Whenever the «SequenceGroup» is used the associations will be numbered according to their corresponding sequence in the DTD. The numbering is done using constraints on the association.
- Mutiplicities of the elements will be indicated in the standard UML manner. (0..1, 0..*, 1..*)
- The «Group» is used when multiple occurrences of the same element are possible (this could be either a sequence or selection).
- Mixed content is represented by a PCDATA stereotype that additionally has an association with an «ChoiceGroup» class.

Example:
Suppose we would have the following elements as part of our initial DTD:

```
<!ELEMENT resources    (resource*)>
<!ELEMENT resource     (metadata?,file*,dependency*)>
```

This would lead to the following class diagram:



Packages

The UML diagrams are based on a number of different packages. For the levelc binding we have defined 6 distinct packages being:

- **imsld_v0p8a** this package contains the level a learning design elements
- **imsld_v0p8b** this package contains the level b learning design elements
- **imsld_v0p8c** this package contains the level c learning design elements
- **imsld_globalb** this package contains the global elements for the level b learning design
- **imsld_globalc** this package contains the global elements for the level c learning design
- **imscp_v1p1** this package contains the elements defined in IMS content packaging

The relations between the packages are described in the package diagrams. The stereotype «access» on the dependency relation between packages defines that elements from one package can be used/accessed from the other.

Furthermore each class has an indication to which package it belongs. E.g. imsld_v0p8b::property-ref defines the class property-ref in package imsld_v0p8b.

Classes that are part of the DTD constructs as mentioned above, are not explicitly assigned to a particular package.

Validation of the models

In order to validate the models described in the following sections, a XSLT script has been written that generates a DTD on the basis of the exported UML model, which is an XML application. The resulting DTD and the original DTD's are identical with the exception of the following:

- sequence of the choice groups are arbitrary. The described model is the same
- IMS content package 1.1.3 elements have been used instead of the original DTD representation of them.
- Unnecessary wrappers like global-elements have been left out
- Namespace attributes have been left out

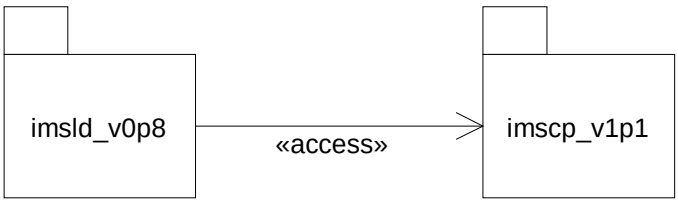
The resulting DTD's have been compared with the original one provided and differences have been detected with the exception of the ones mentioned above.

Models

The diagrams are separated into three models, each covering either levela, levelb or levelc of the learning design. The elements for each level are assigned to a corresponding package and are furthermore separated from the rest using different shades of grey.

Model level A

Chart ID : IMSLD
Chart Name : IMS learning design
Chart Type : UML Package Diagram



Class Diagram: item model

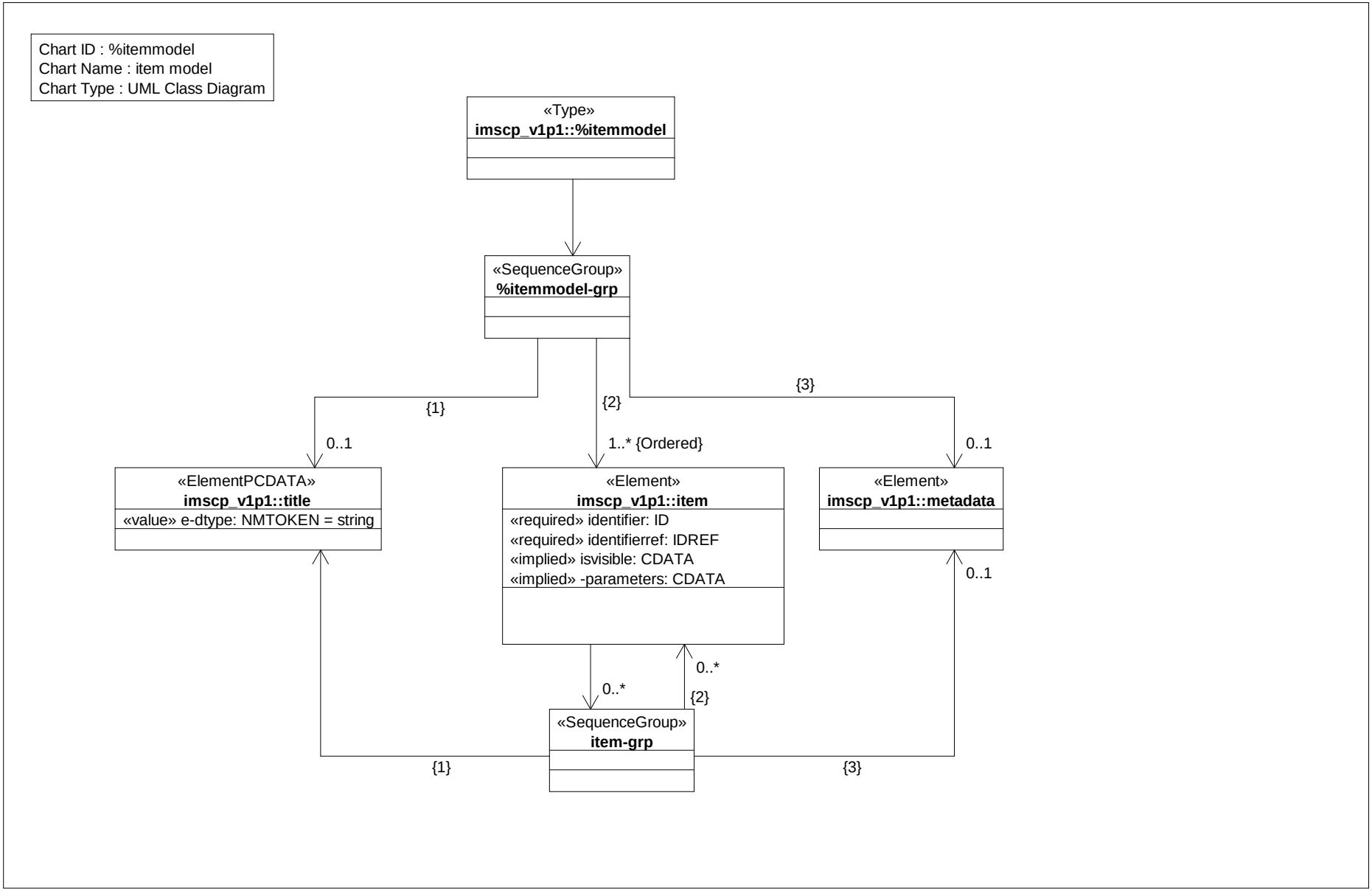
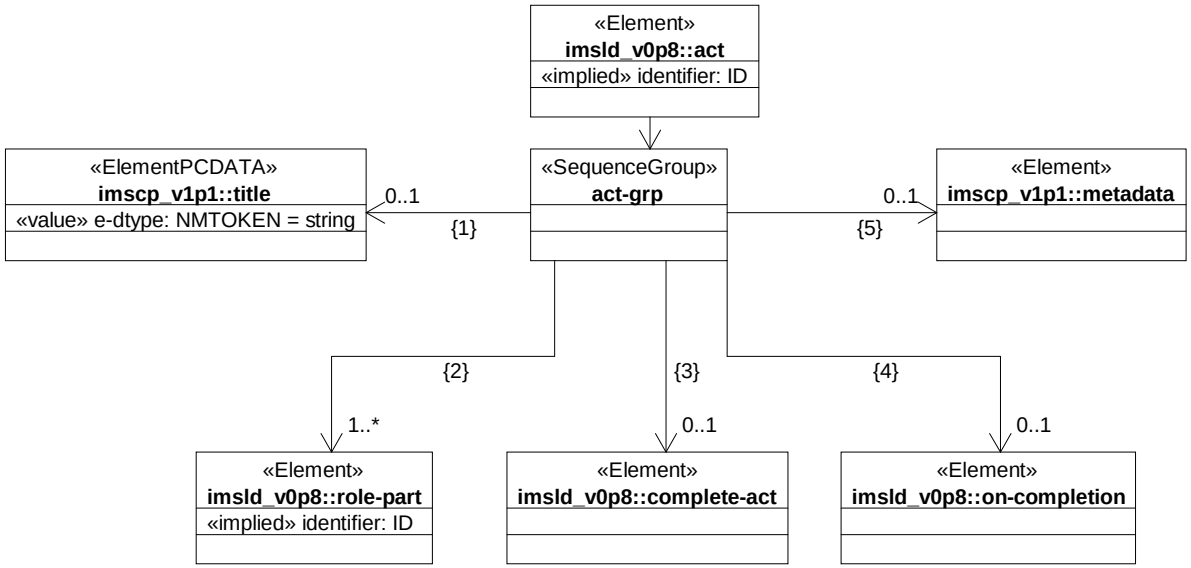


Chart ID : act
Chart Name : act
Chart Type : UML Class Diagram



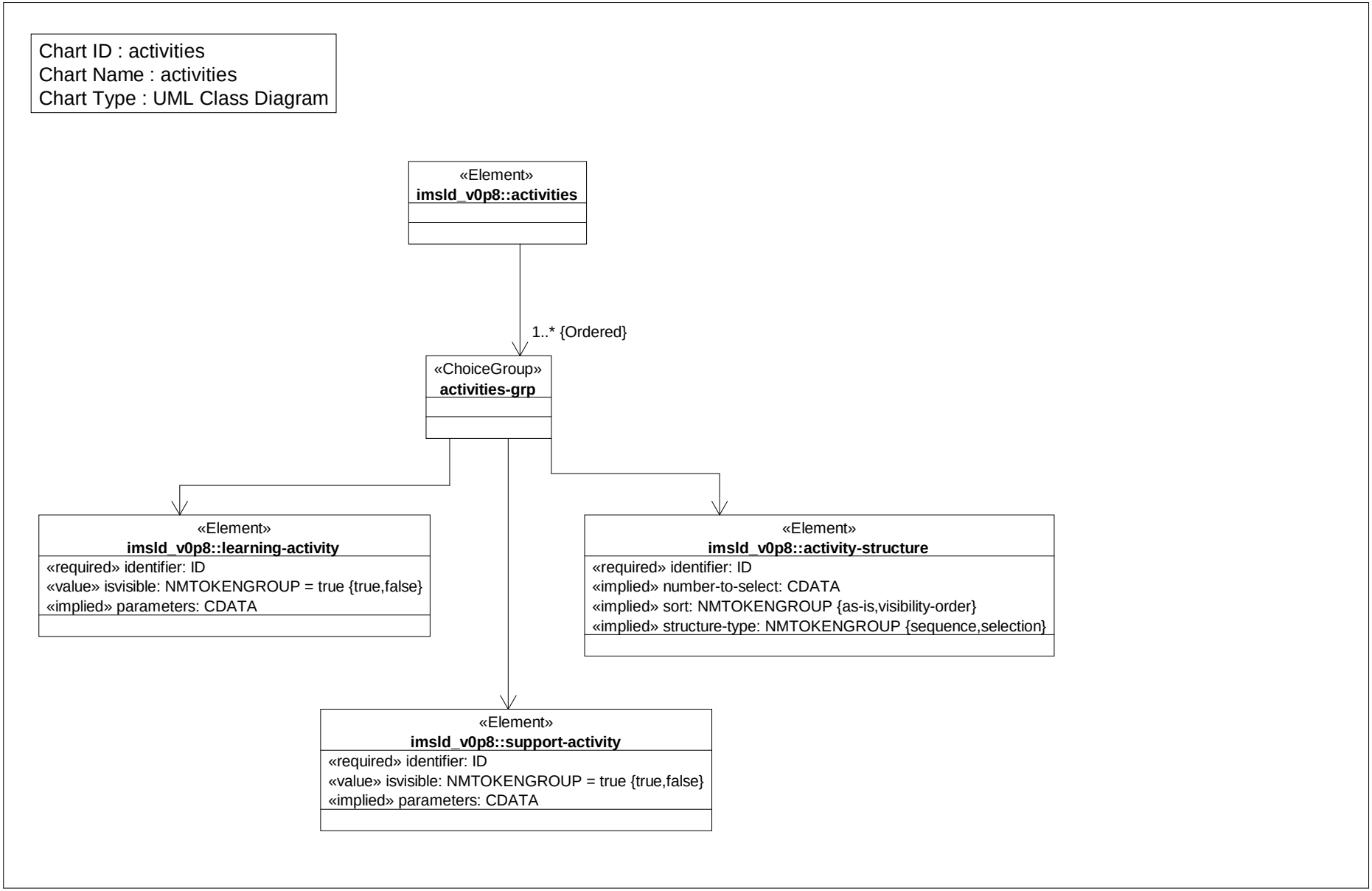


Chart ID : activities
Chart Name : activities
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::activities
1..* {Ordered}
«ChoiceGroup»
activities-grp
«Element»
imsld_v0p8::learning-activity
«required» identifier: ID
«value» isvisible: NMTOKENGROUP = true {true,false}
«implied» parameters: CDATA
«Element»
imsld_v0p8::activity-structure
«required» identifier: ID
«implied» number-to-select: CDATA
«implied» sort: NMTOKENGROUP {as-is,visibility-order}
«implied» structure-type: NMTOKENGROUP {sequence,selection}
«Element»
imsld_v0p8::support-activity
«required» identifier: ID
«value» isvisible: NMTOKENGROUP = true {true,false}
«implied» parameters: CDATA

Class Diagram: activty structure

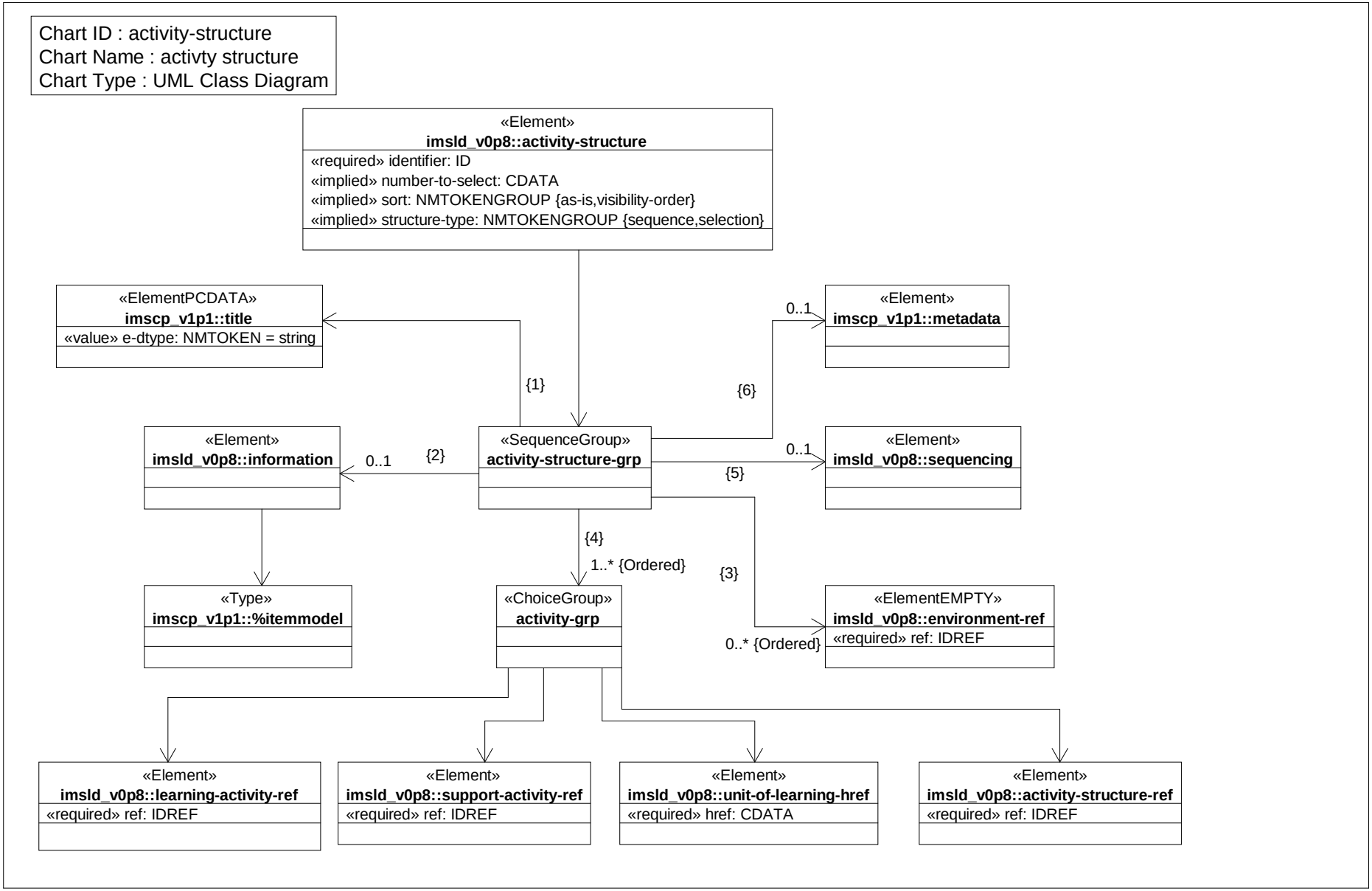


Chart ID : activity-structure-ref
Chart Name : activity structure ref
Chart Type : UML Class Diagram

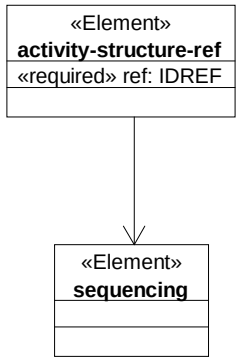


Chart ID : complete-act
Chart Name : complete act
Chart Type : UML Class Diagram

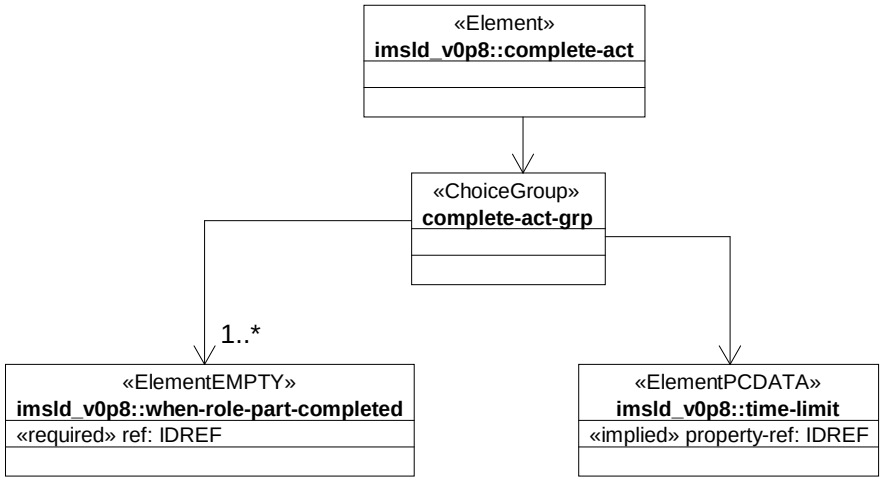


Chart ID : complete-play
Chart Name : complete play
Chart Type : UML Class Diagram

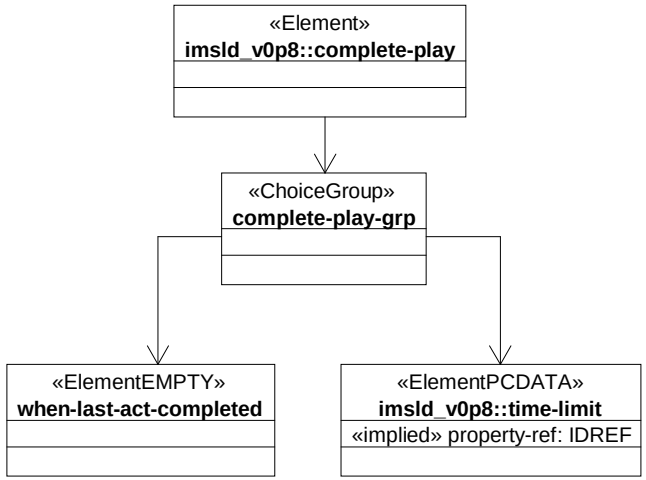
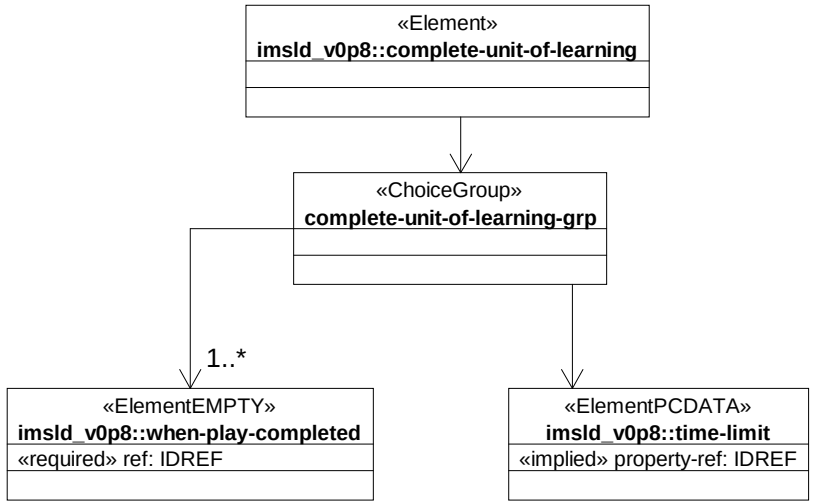
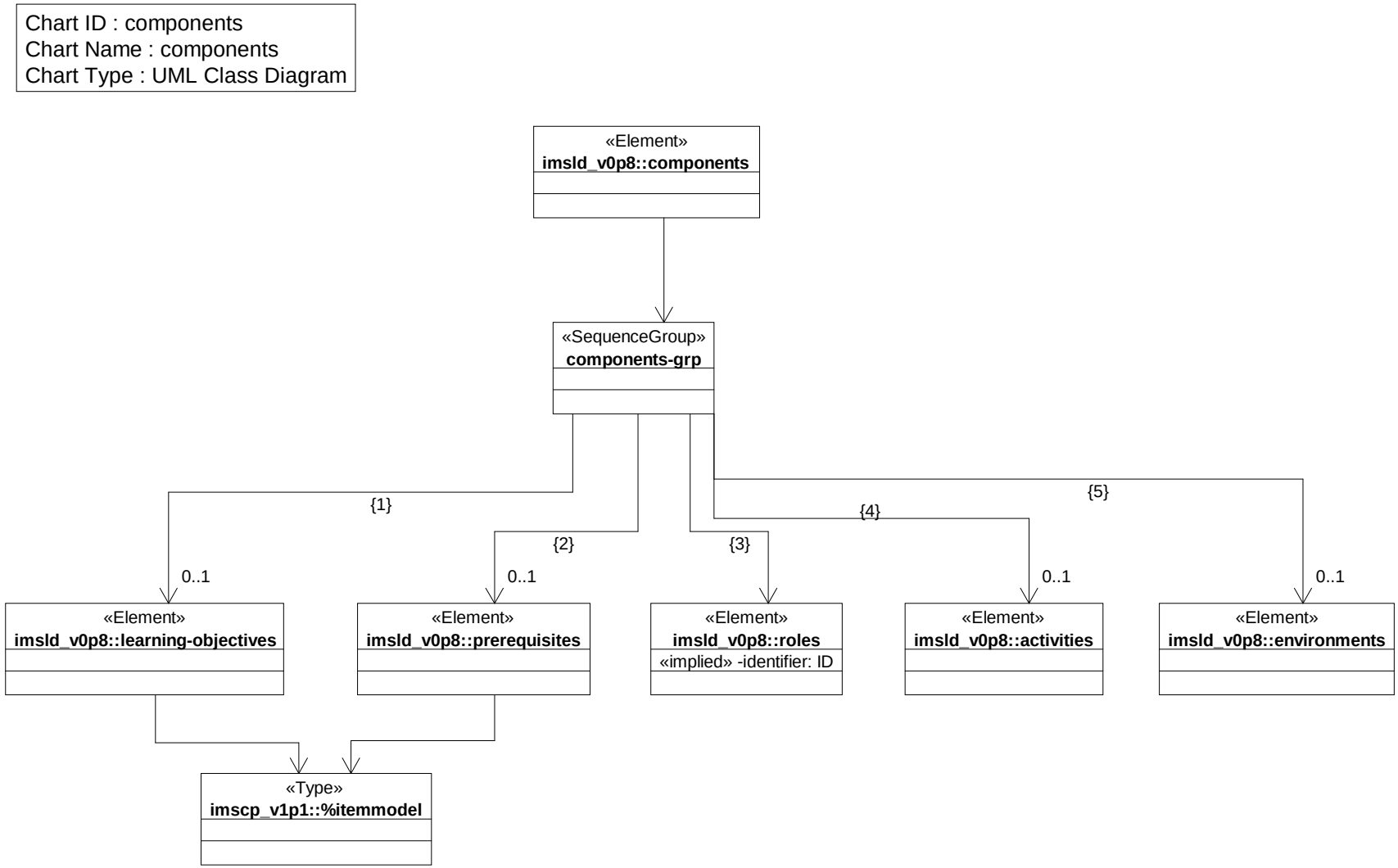


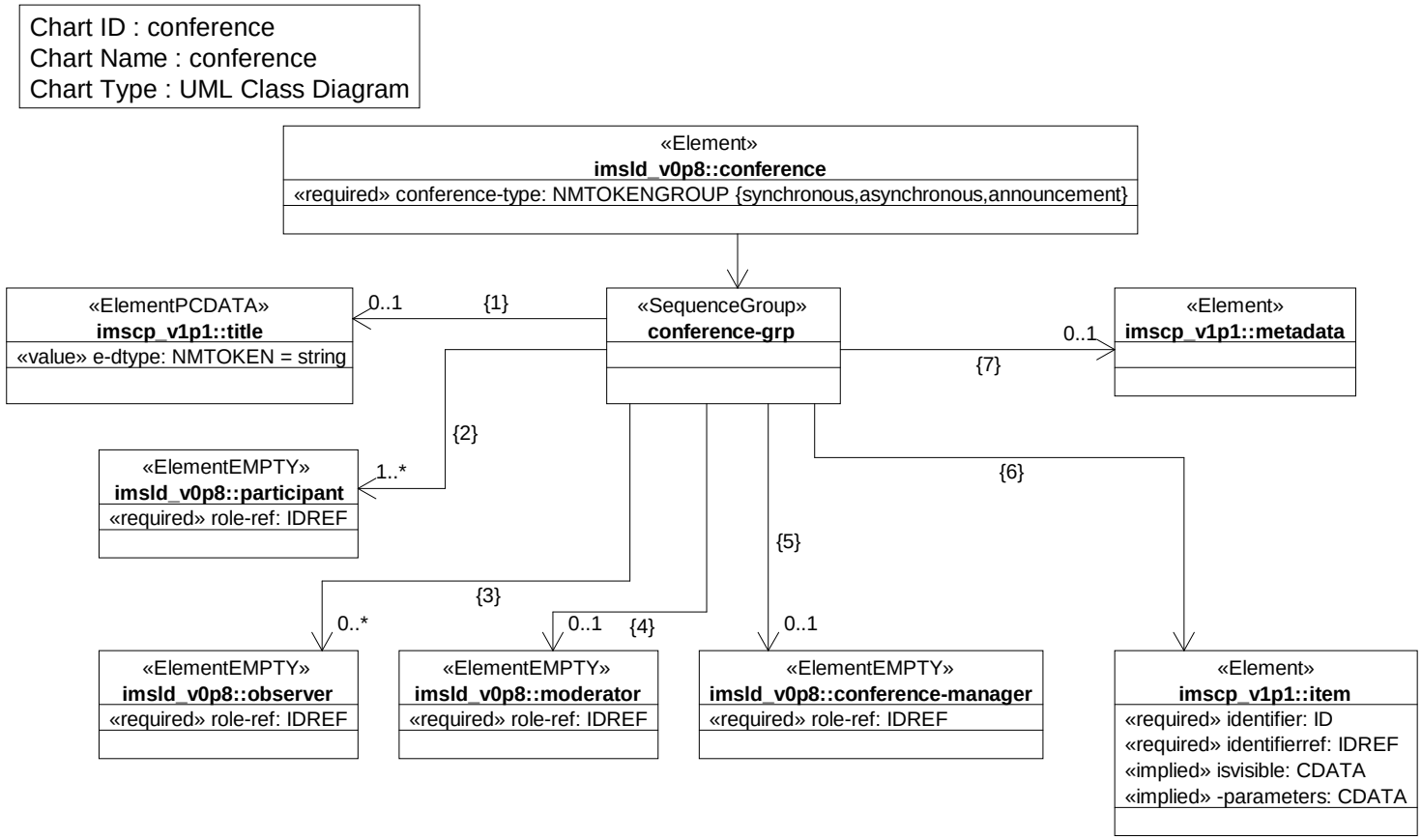
Chart ID : complete-unit-of-learning
Chart Name : complete unit of learning
Chart Type : UML Class Diagram



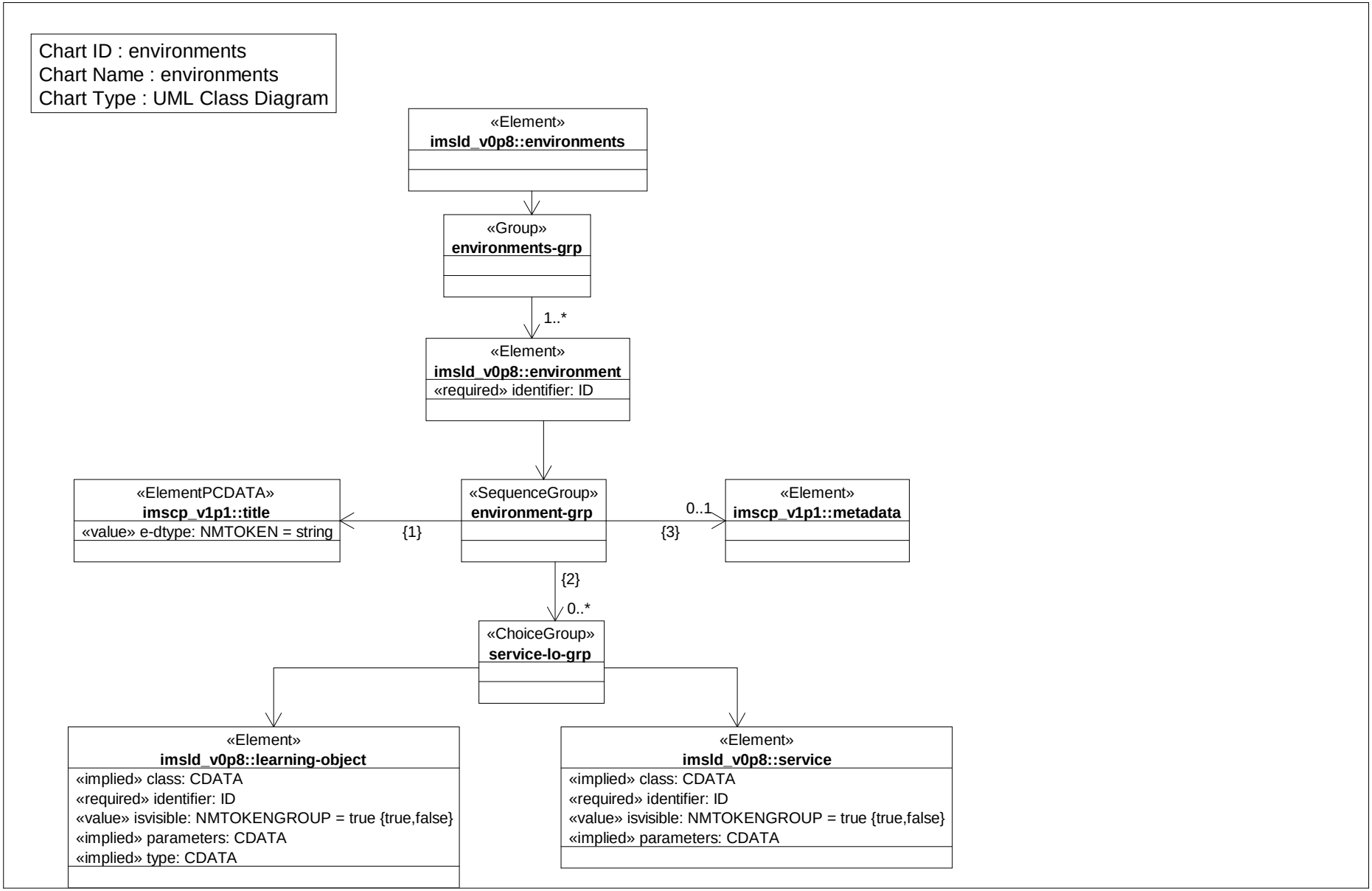
Class Diagram: components



Class Diagram: conference



Class Diagram: environments



Class Diagram: index search

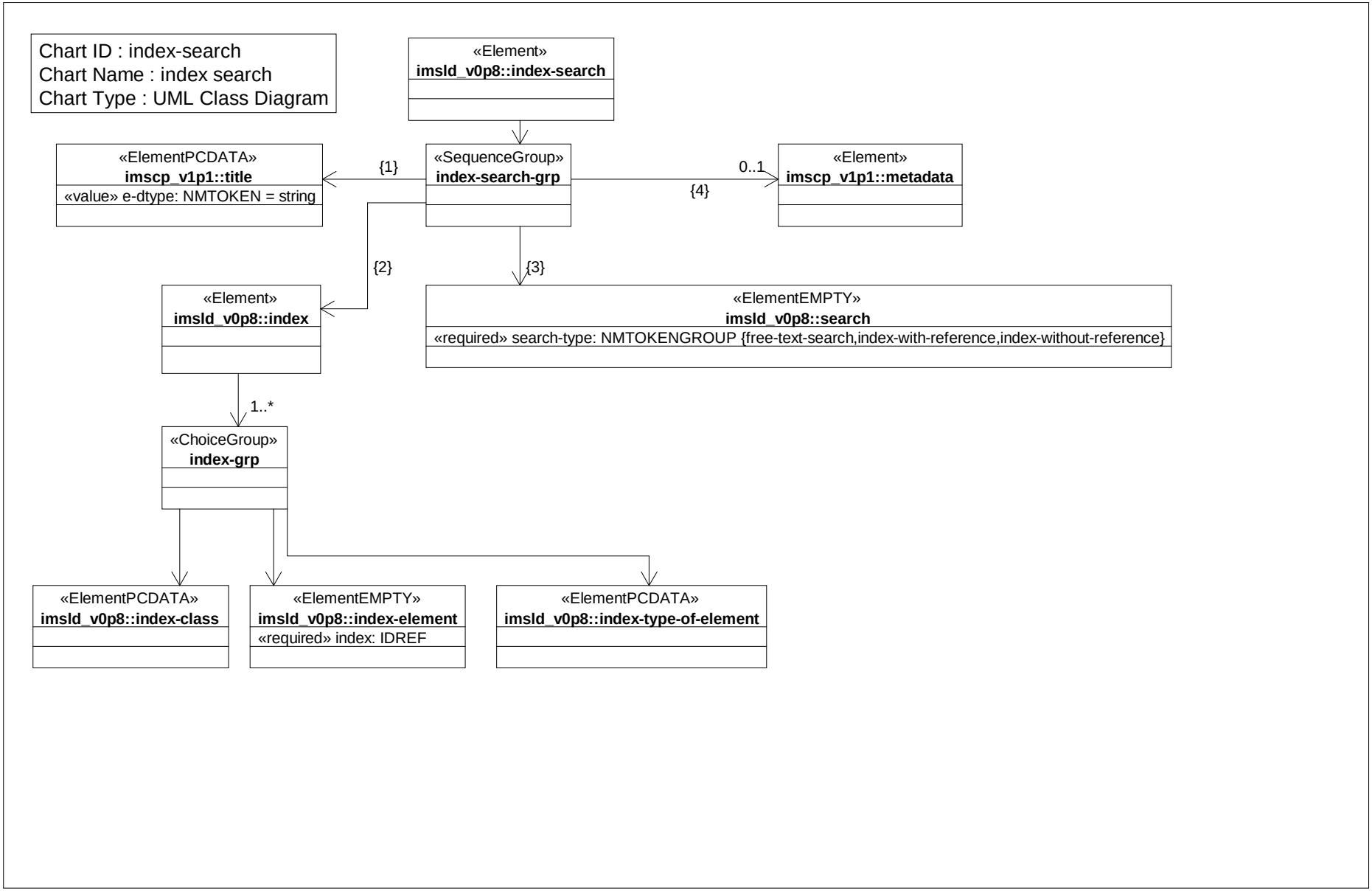
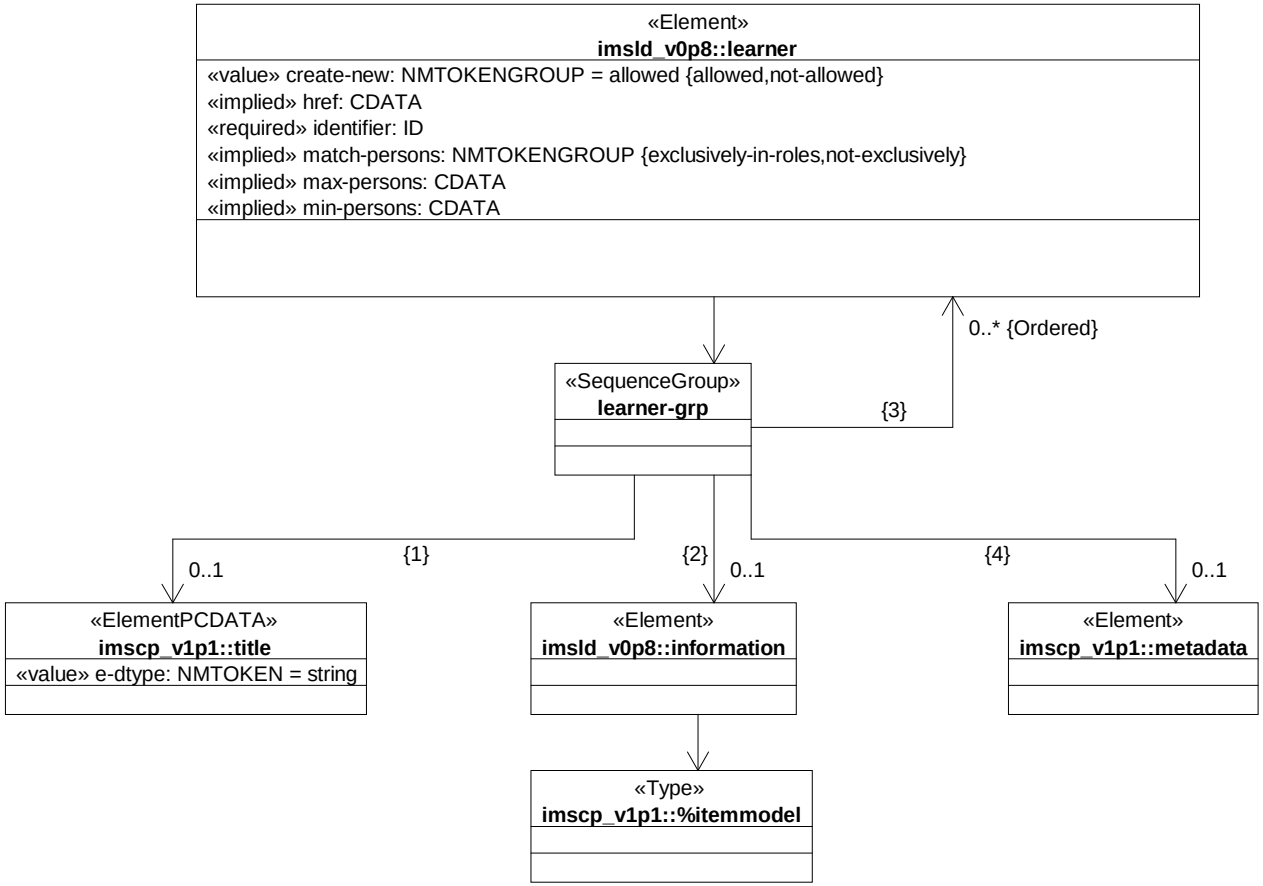


Chart ID : learner
Chart Name : learner
Chart Type : UML Class Diagram



Class Diagram: learning activity

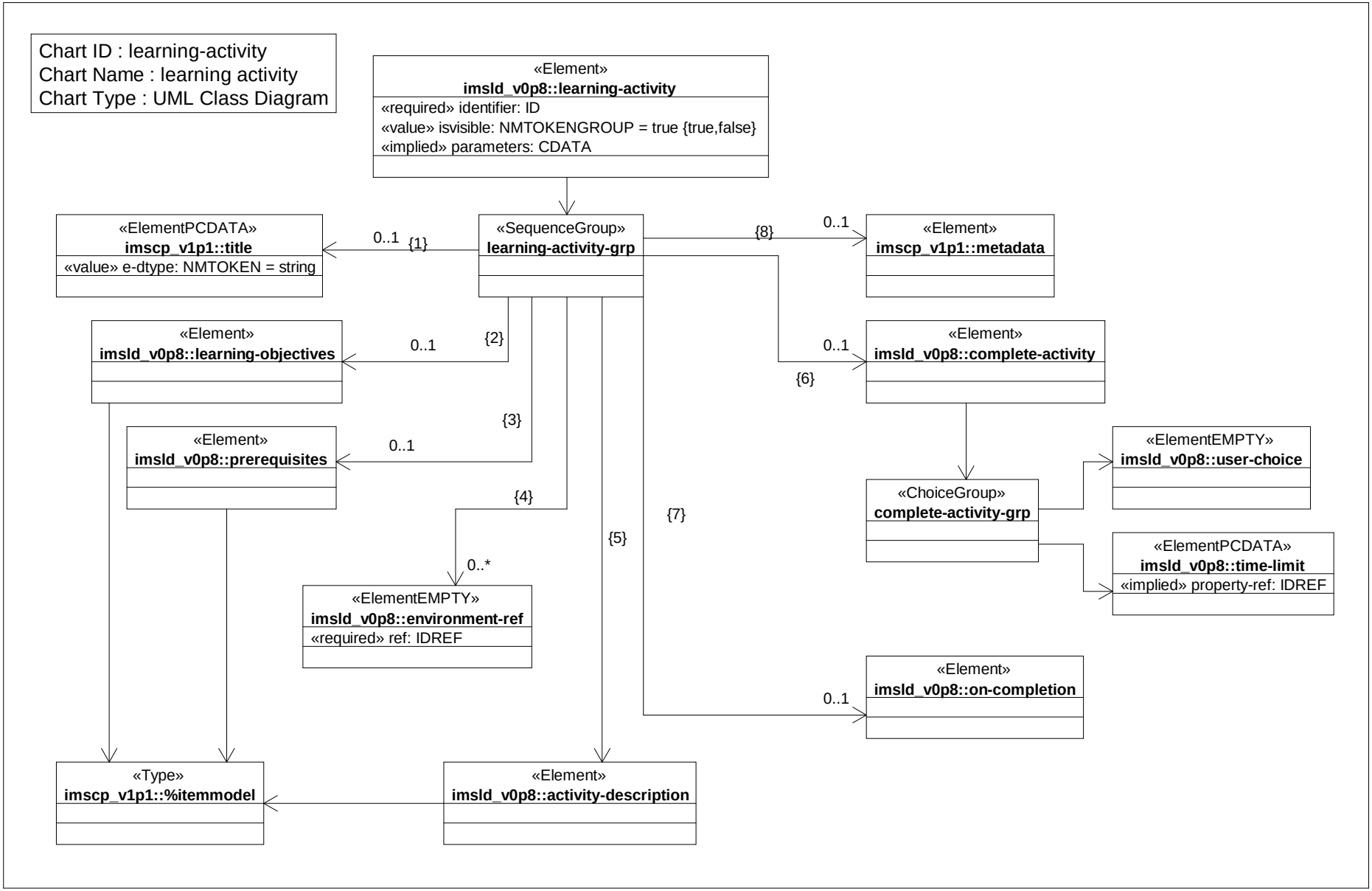


Chart ID : learning-activity-ref
Chart Name : learning activity ref
Chart Type : UML Class Diagram

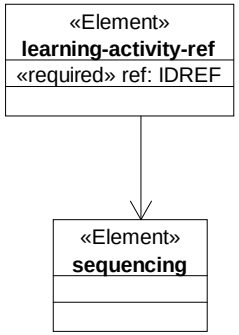


Chart ID : learning-design
Chart Name : learning design
Chart Type : UML Class Diagram

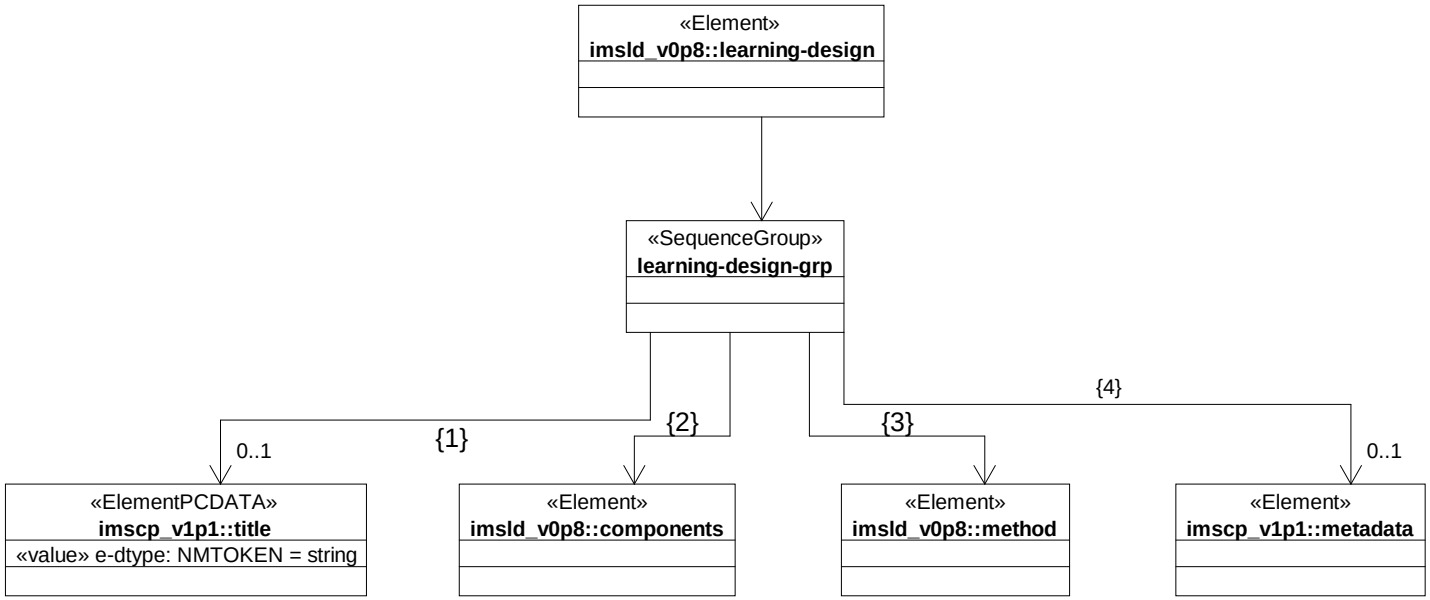
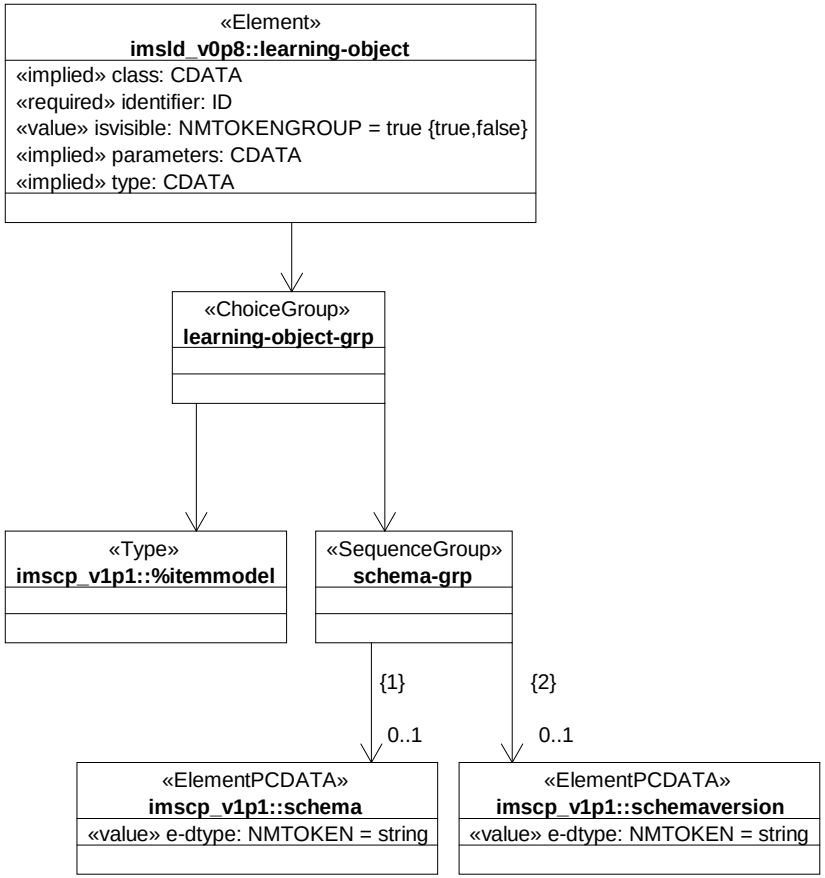


Chart ID : learning-object
Chart Name : learning object
Chart Type : UML Class Diagram



Class Diagram: manifest

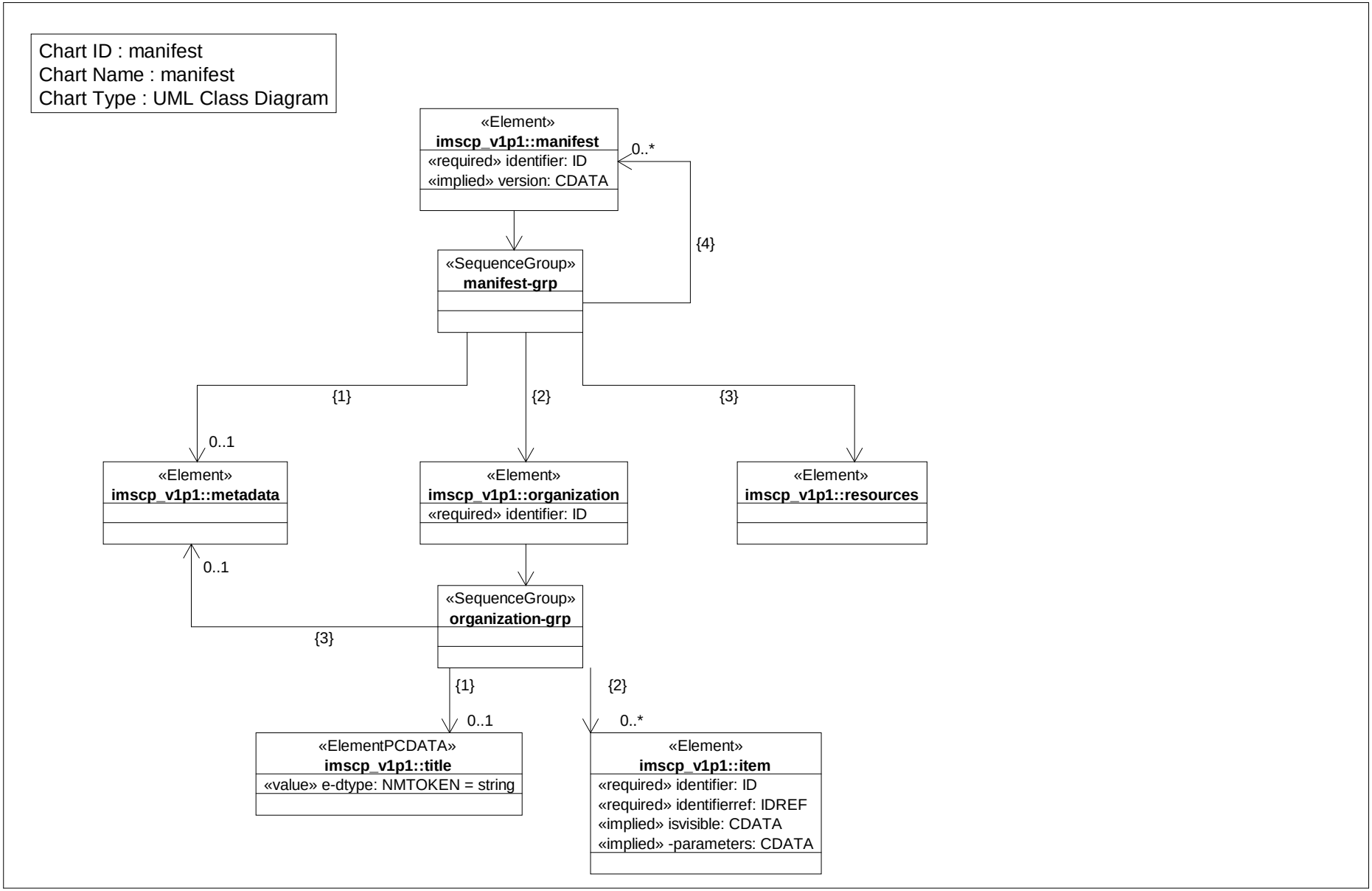


Chart ID : metadata
Chart Name : metadata
Chart Type : UML Class Diagram

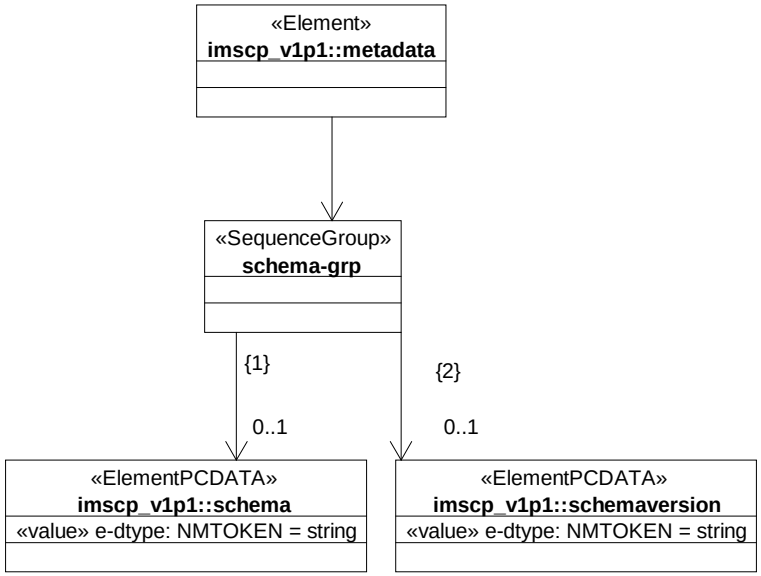


Chart ID : method
Chart Name : method
Chart Type : UML Class Diagram

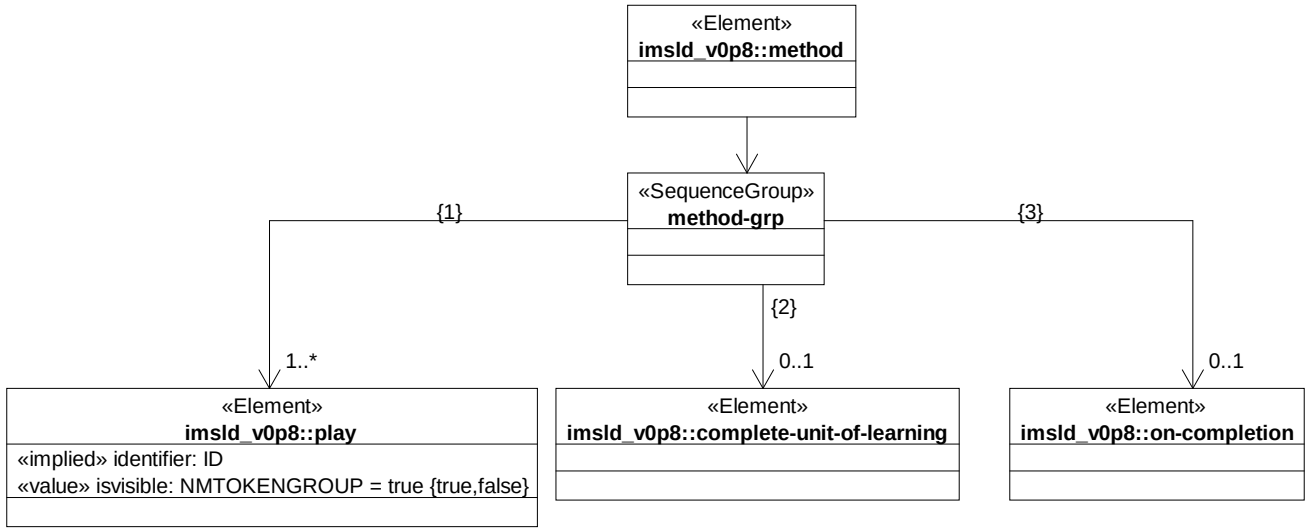
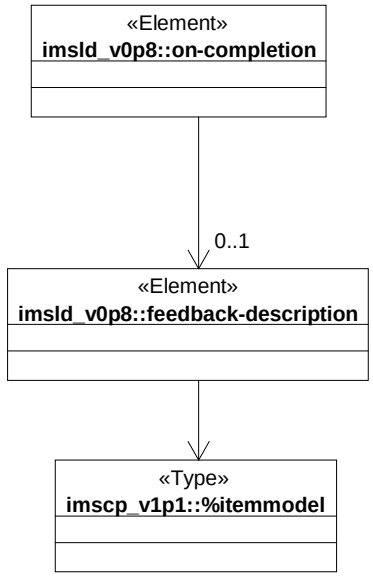
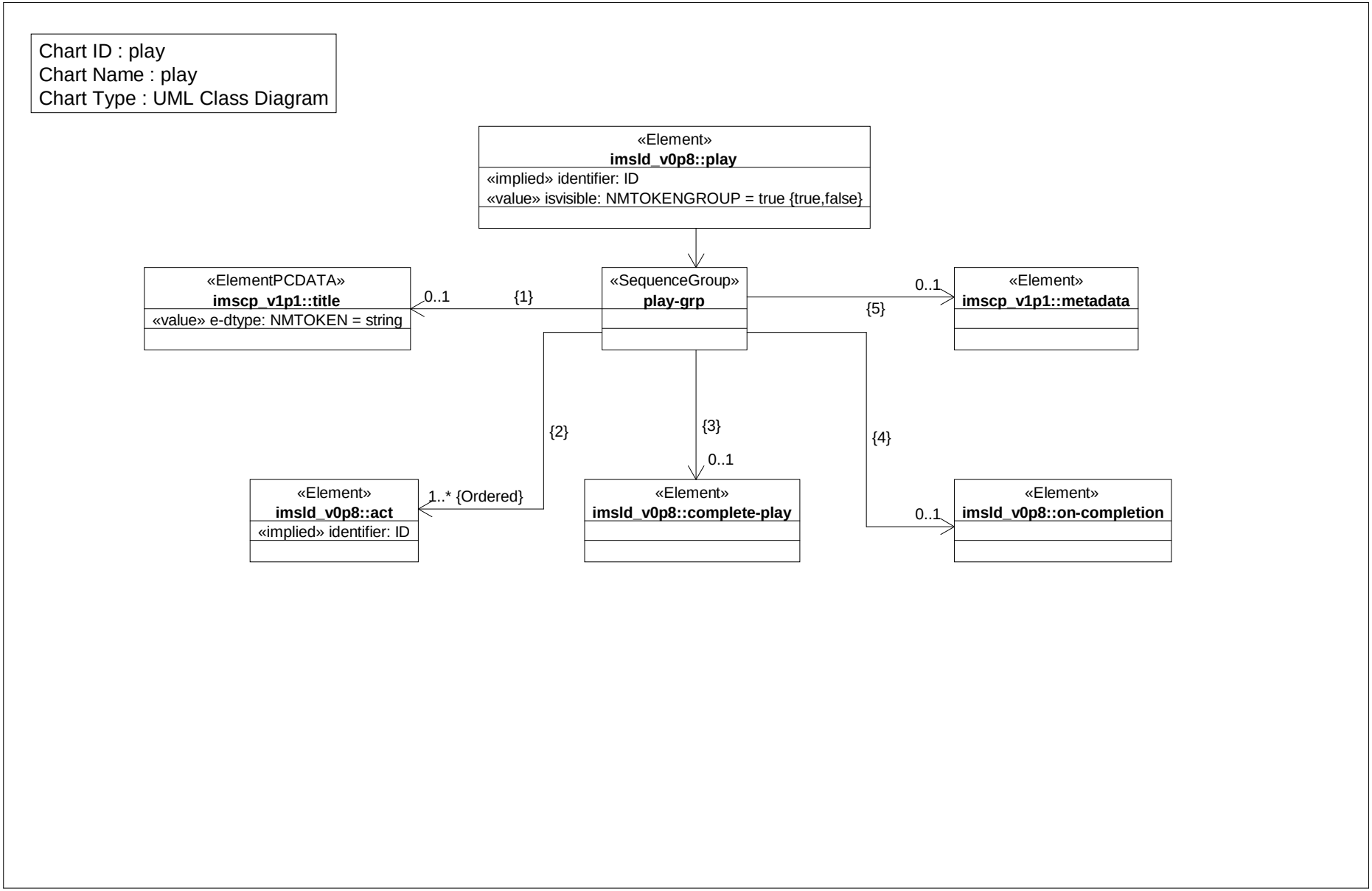


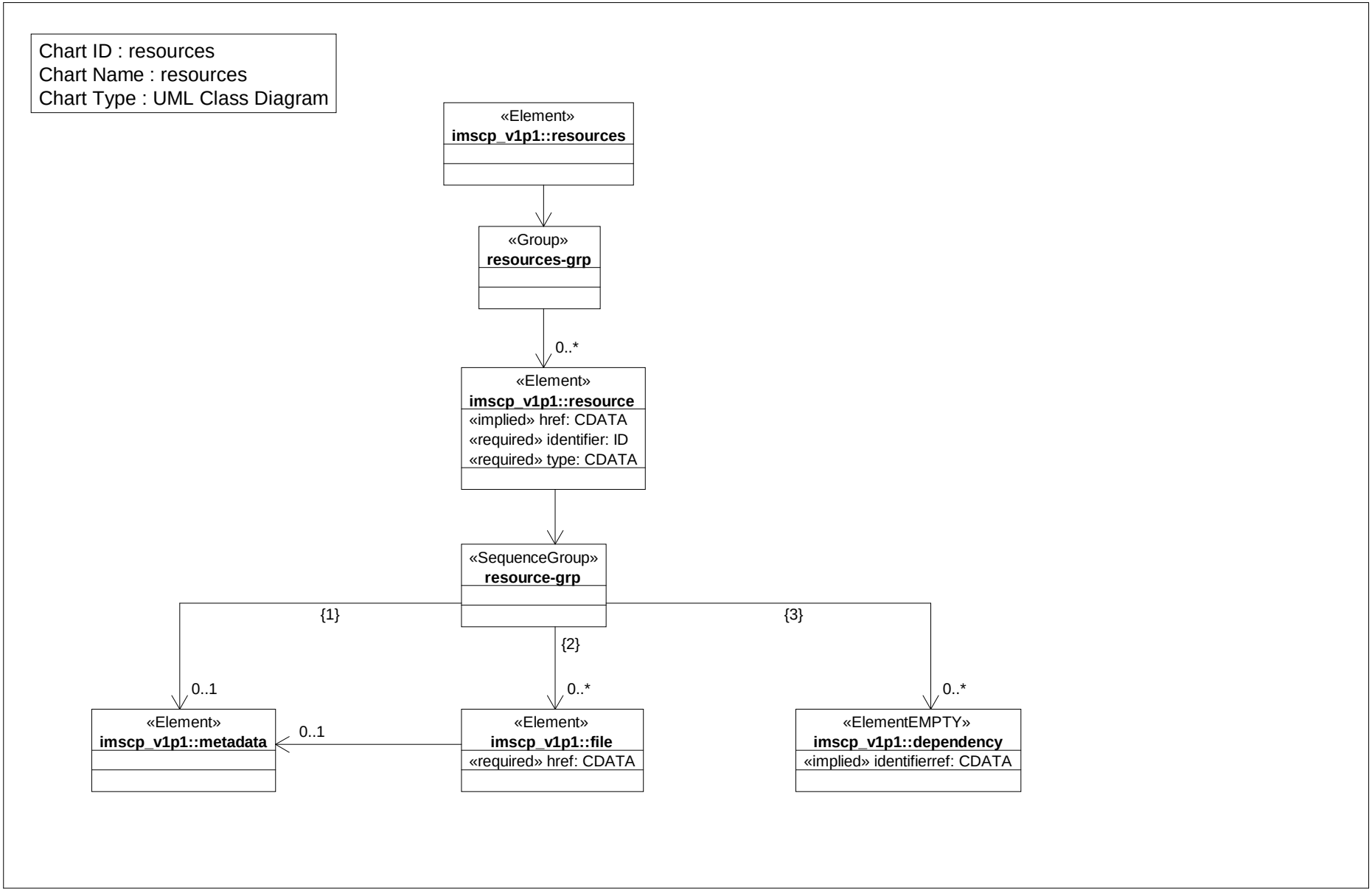
Chart ID : on-completion
Chart Name : on completion
Chart Type : UML Class Diagram



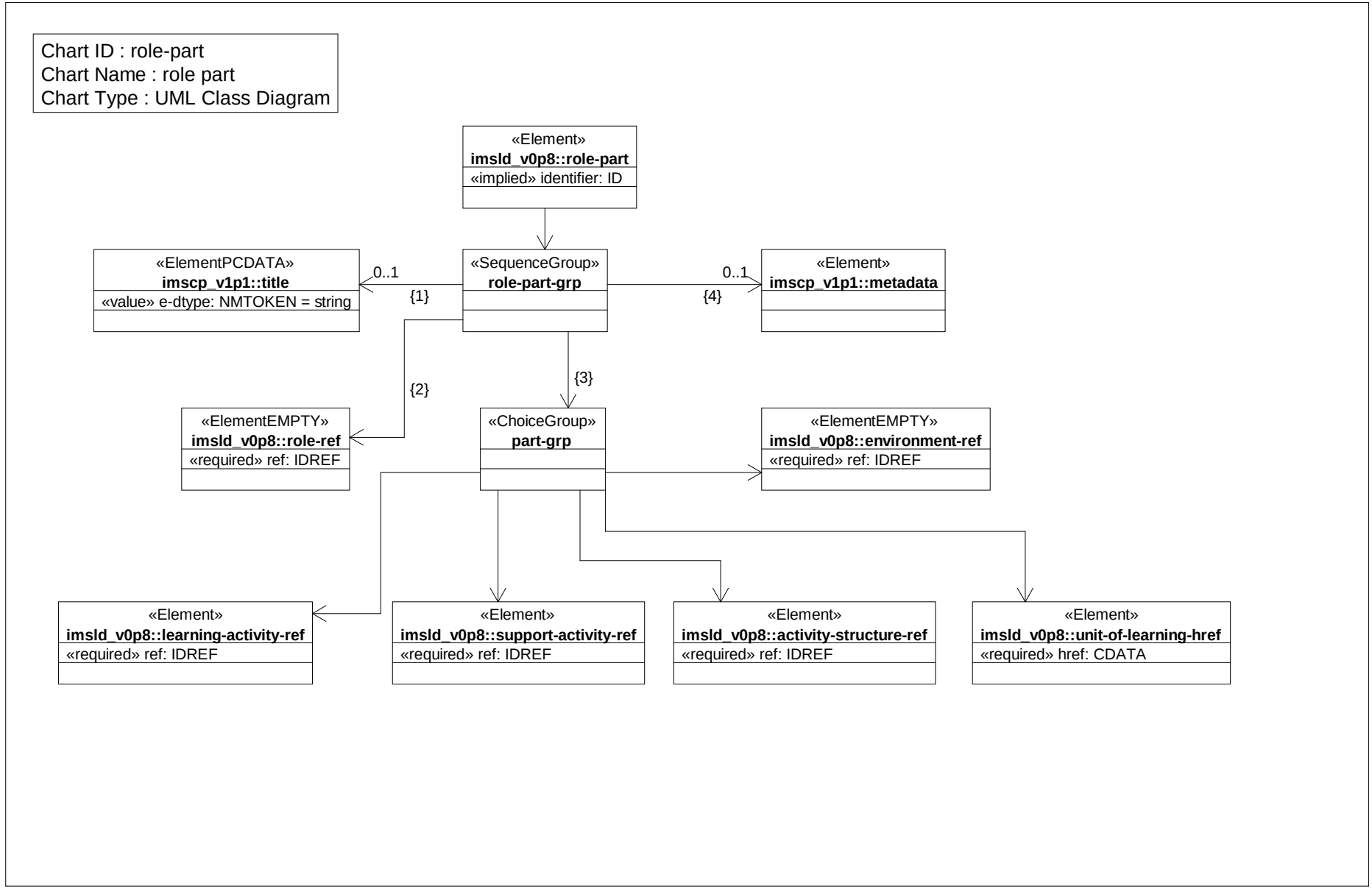
Class Diagram: play



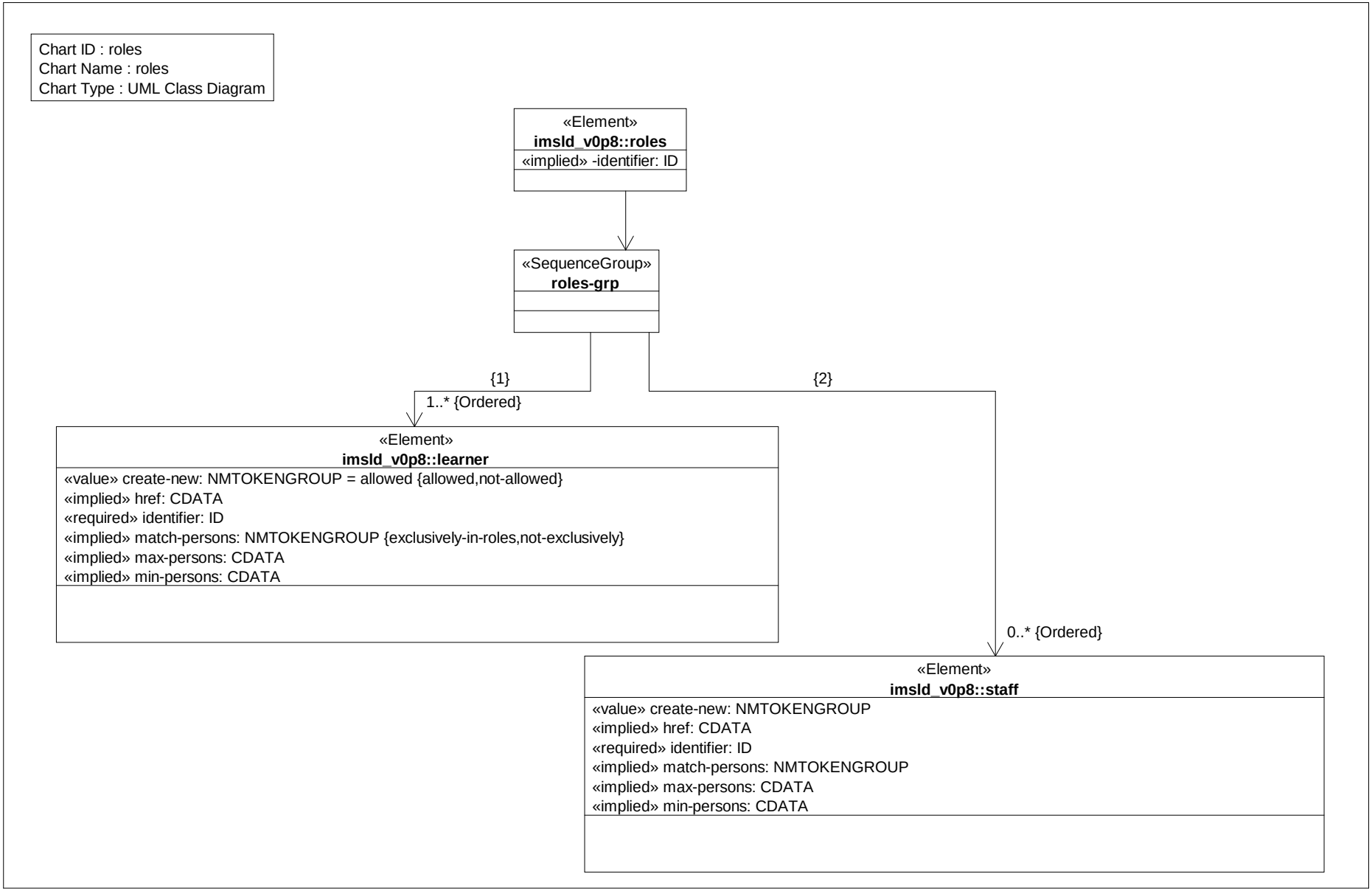
Class Diagram: resources



Class Diagram: role part



Class Diagram: roles



Class Diagram: send mail

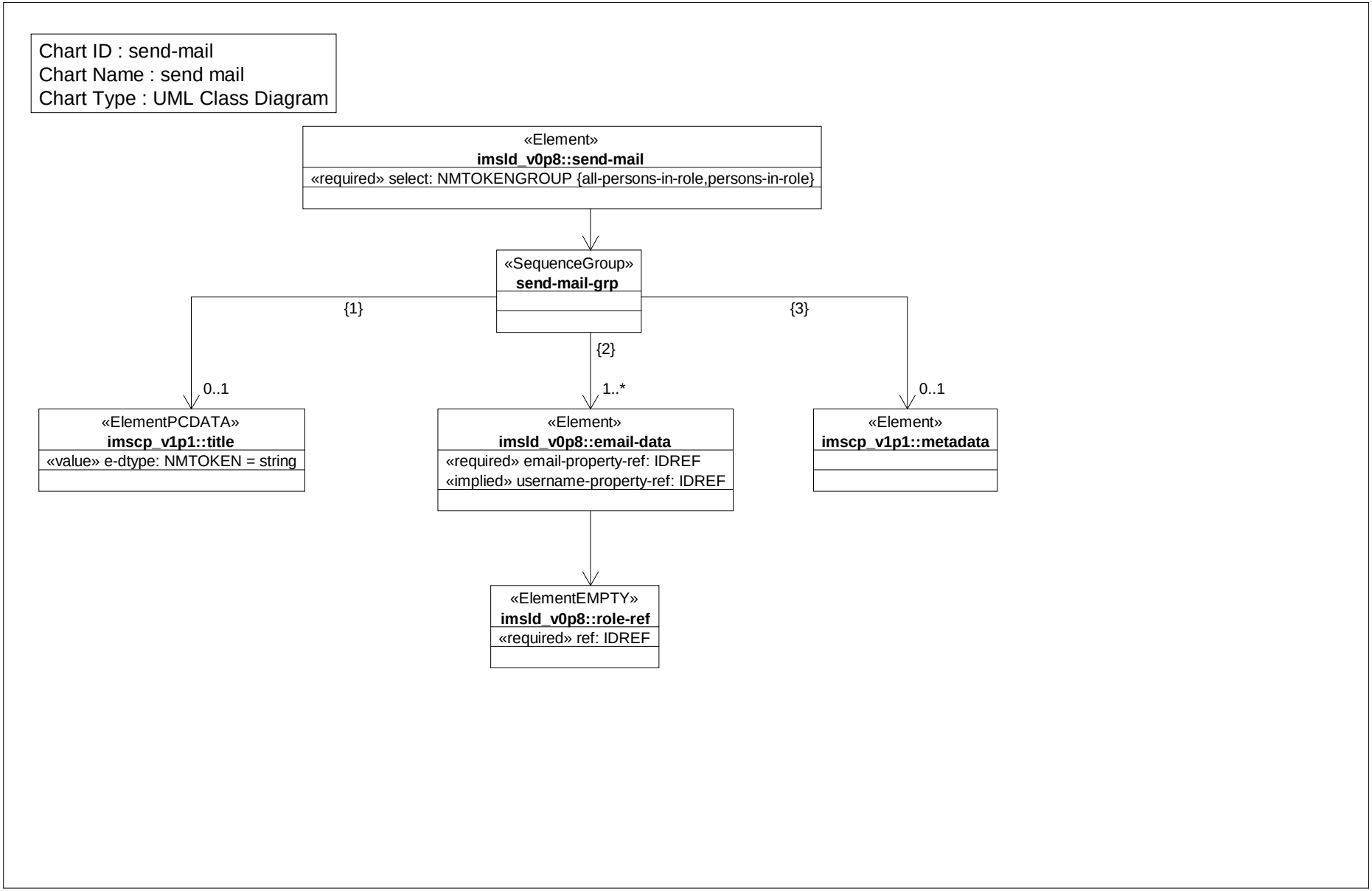
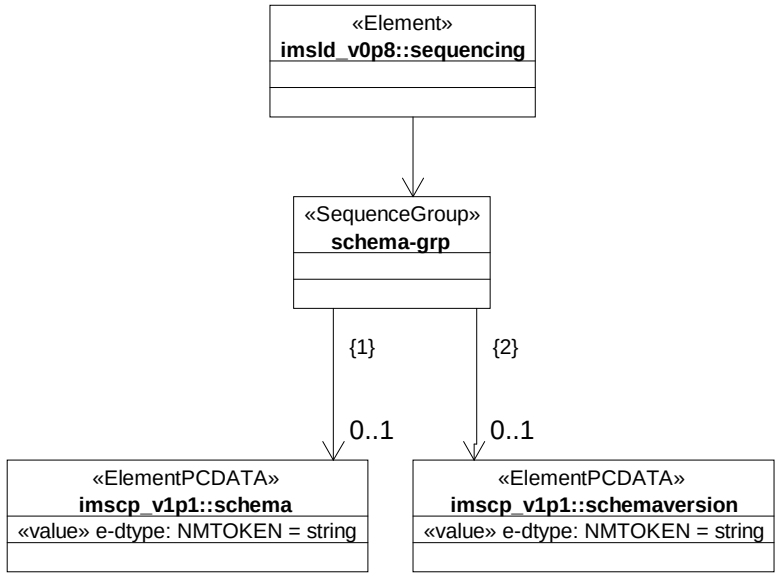


Chart ID : sequencing
Chart Name : sequencing
Chart Type : UML Class Diagram



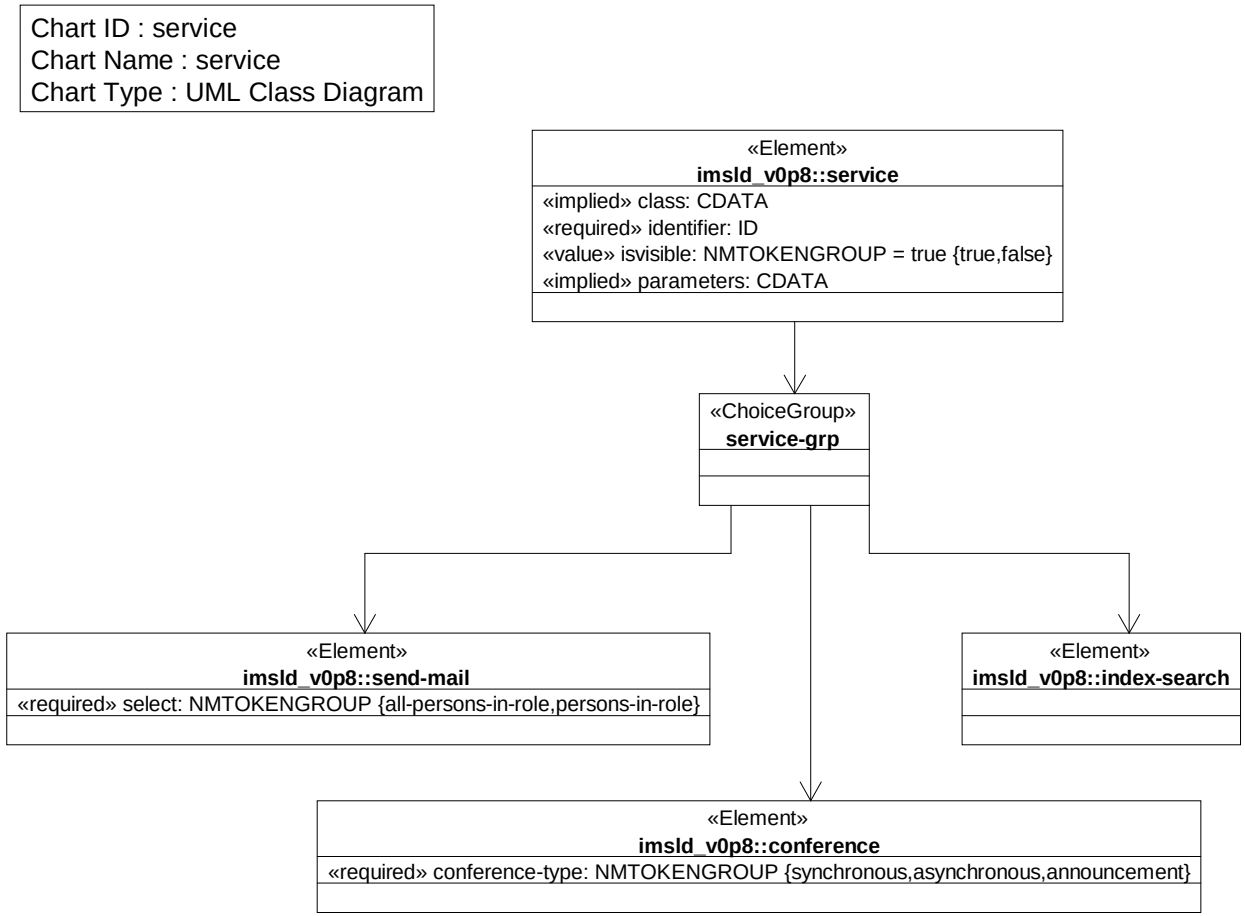


Chart ID : service
Chart Name : service
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::service
«implied» class: CDATA
«required» identifier: ID
«value» isvisible: NMTOKENGROUP = true {true,false}
«implied» parameters: CDATA
«ChoiceGroup»
service-grp
«Element»
imsld_v0p8::send-mail
«required» select: NMTOKENGROUP {all-persons-in-role,persons-in-role}
«Element»
imsld_v0p8::index-search
«Element»
imsld_v0p8::conference
«required» conference-type: NMTOKENGROUP {synchronous,asynchronous,announcement}

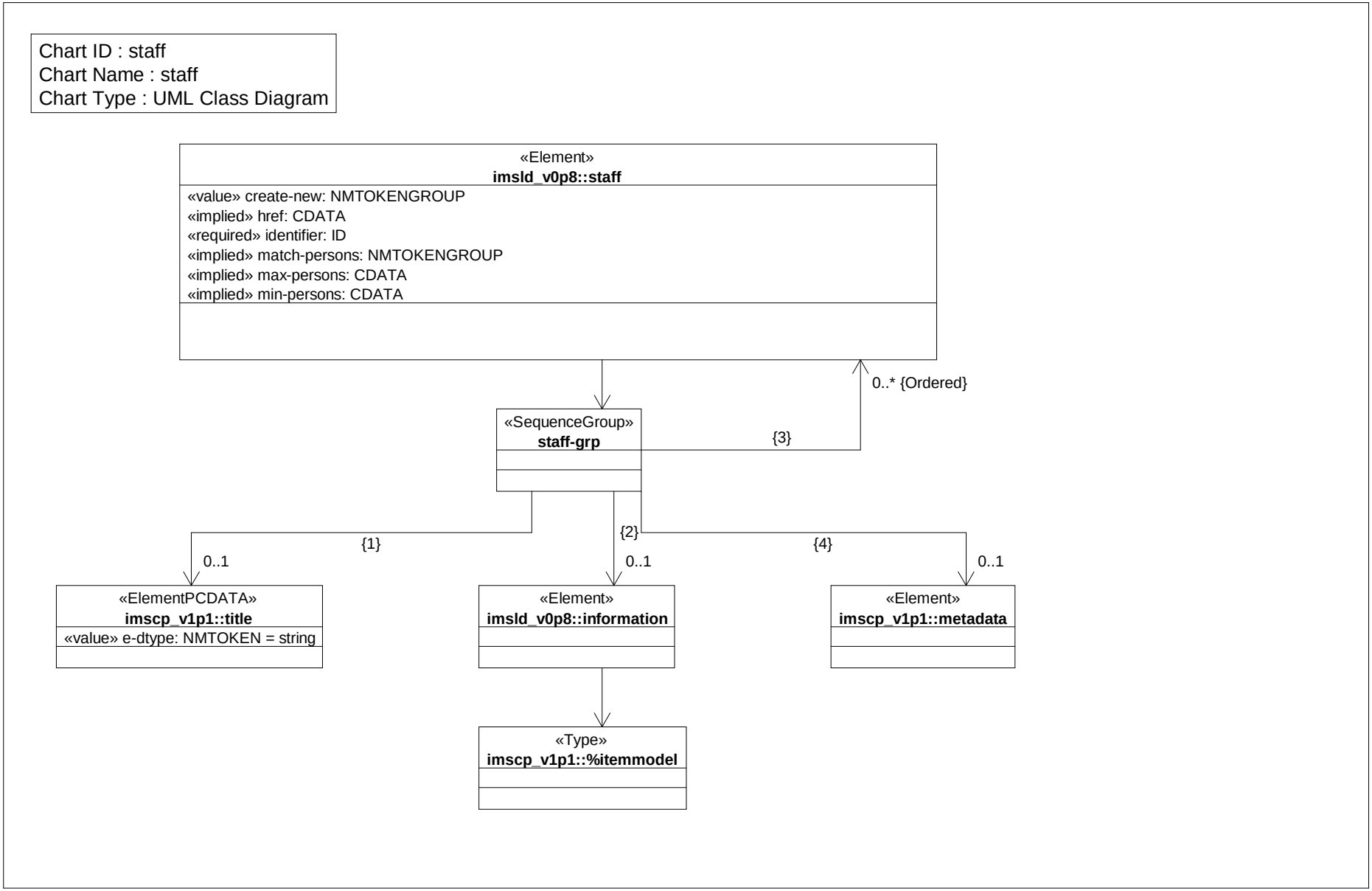


Chart ID : staff
Chart Name : staff
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::staff
«value» create-new: NMTOKENGROUP
«implied» href: CDATA
«required» identifier: ID
«implied» match-persons: NMTOKENGROUP
«implied» max-persons: CDATA
«implied» min-persons: CDATA
0..* {Ordered}
«SequenceGroup»
staff-grp
{3}
{1}
{2}
{4}
0..1
0..1
0..1
«ElementPCDATA»
imscp_v1p1::title
«value» e-dtype: NMTOKEN = string
«Element»
imsld_v0p8::information
«Element»
imscp_v1p1::metadata
«Type»
imscp_v1p1::%itemmodel

Class Diagram: support activity

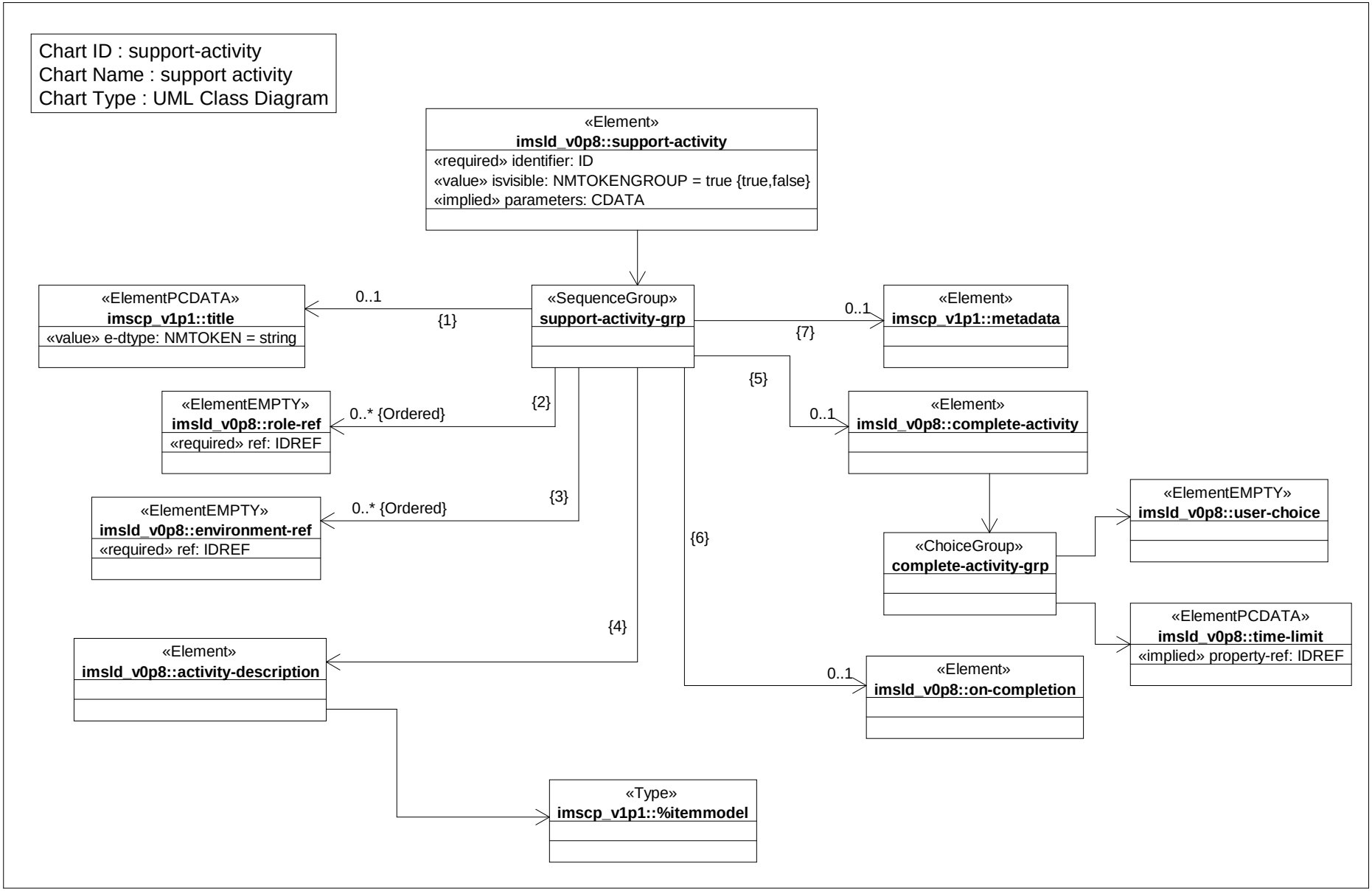
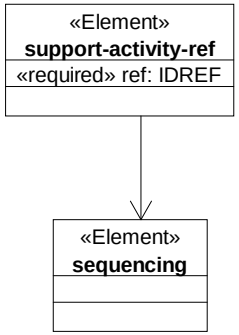


Chart ID : support-activity-ref
Chart Name : support activity ref
Chart Type : UML Class Diagram



Class Diagram: unit of learning

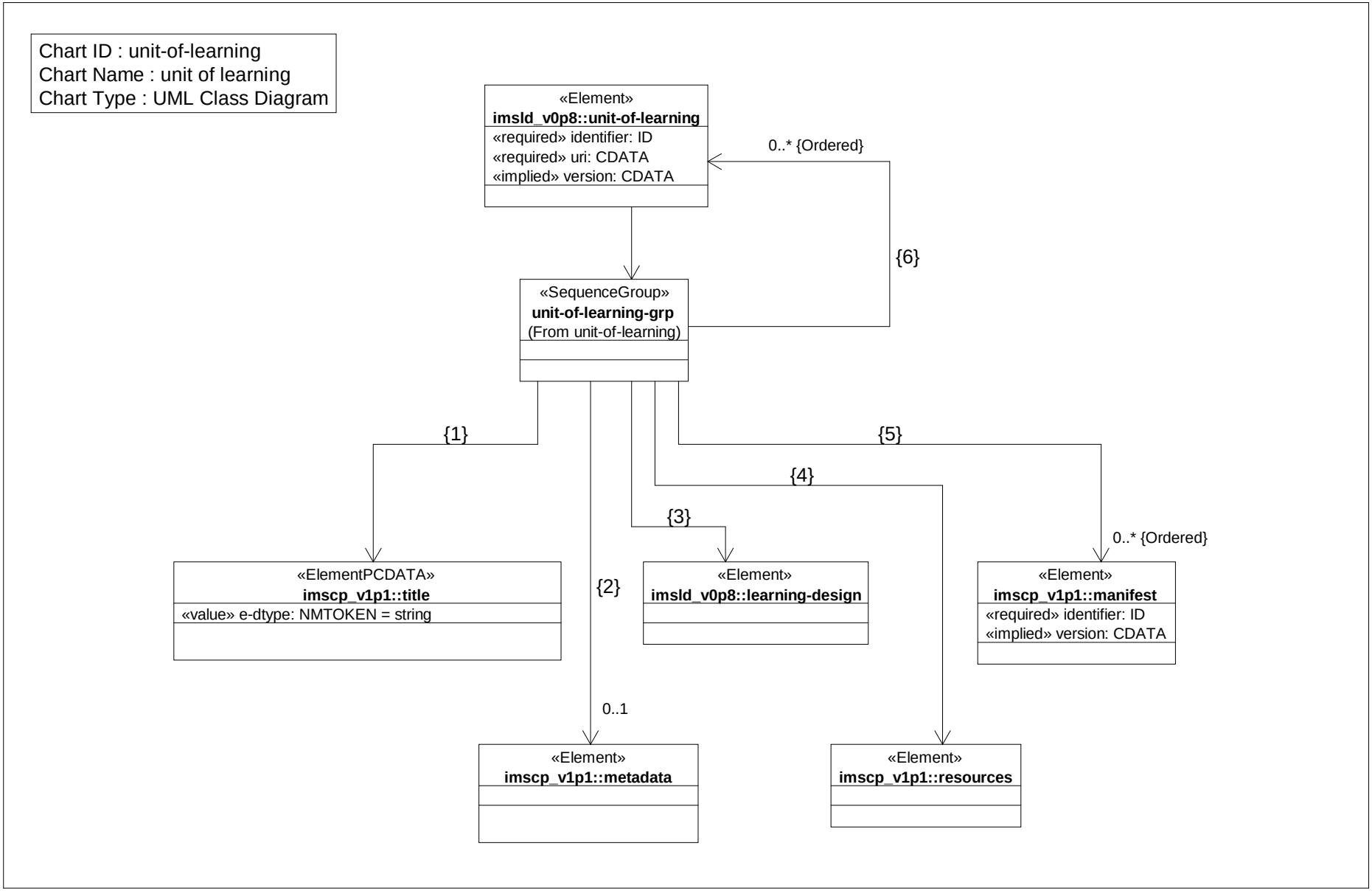
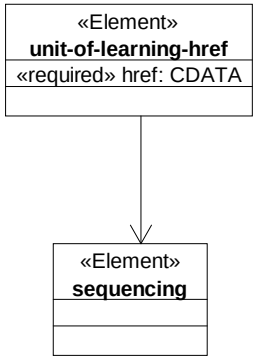


Chart ID : unit-of-learning-href
Chart Name : unit of learning href
Chart Type : UML Class Diagram



Model level B

Chart ID : IMSLD
Chart Name : IMS learning design
Chart Type : UML Package Diagram

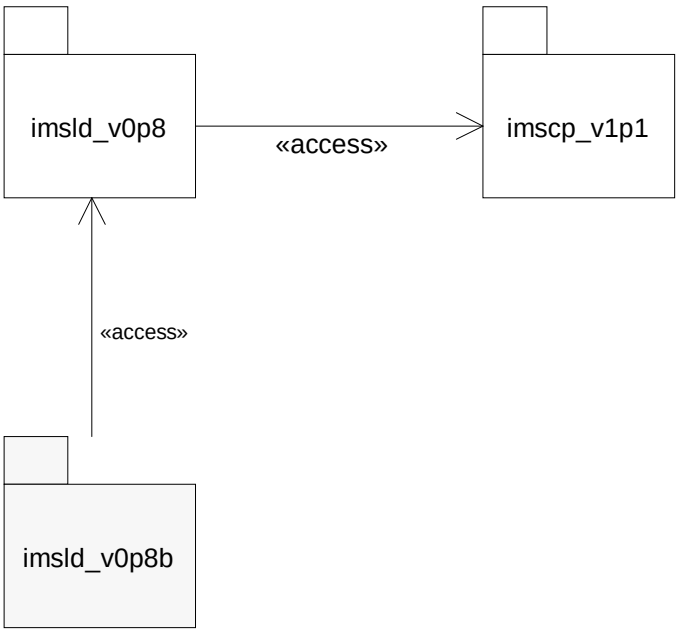


Chart ID : IMSLD-global
Chart Name : IMSLD global elements
Chart Type : UML Package Diagram

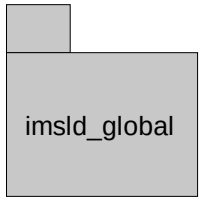
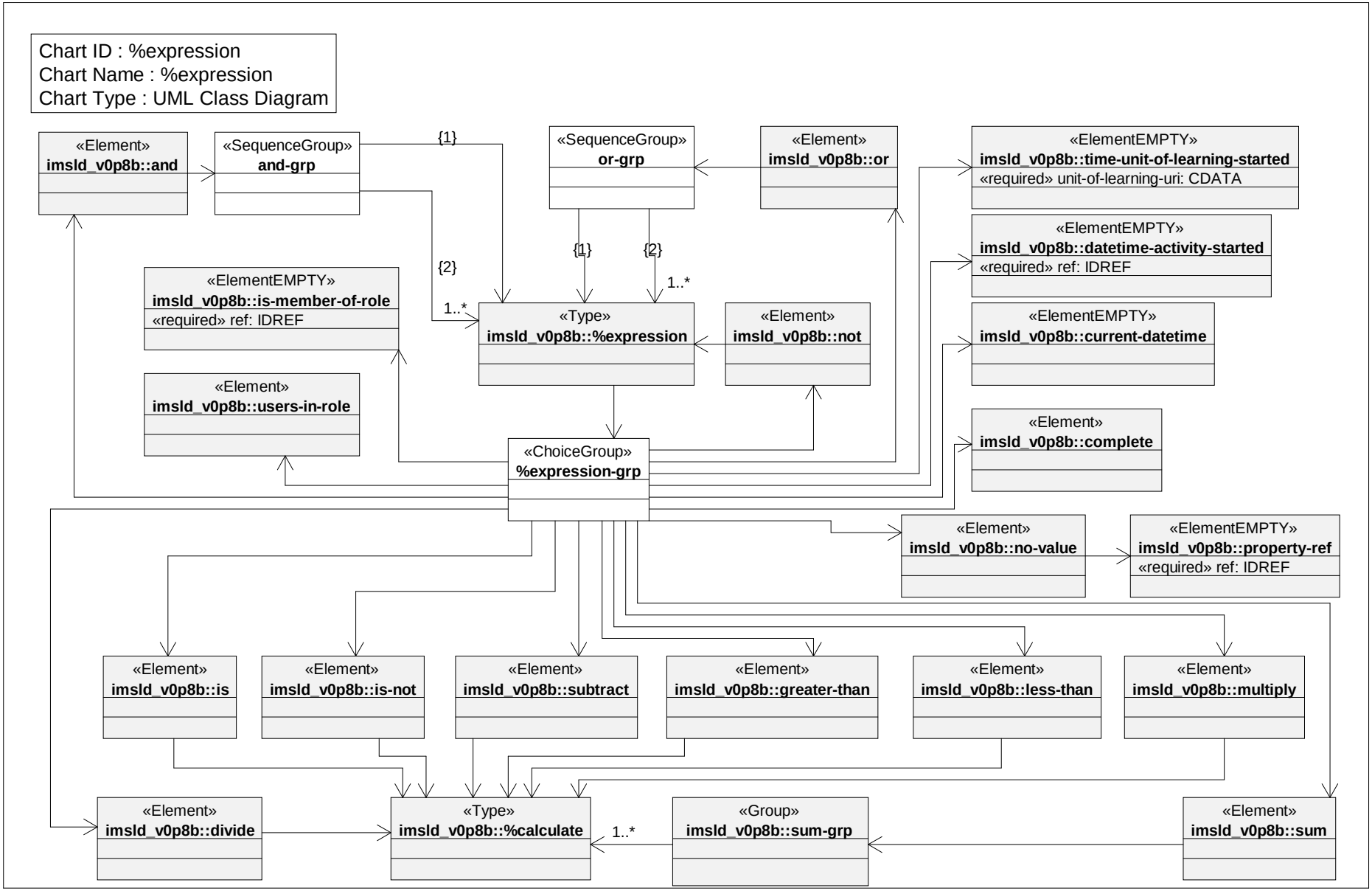


Chart ID : %calculate
Chart Name : %calculate
Chart Type : UML Class Diagram

Class Diagram: %expression



Class Diagram: item model

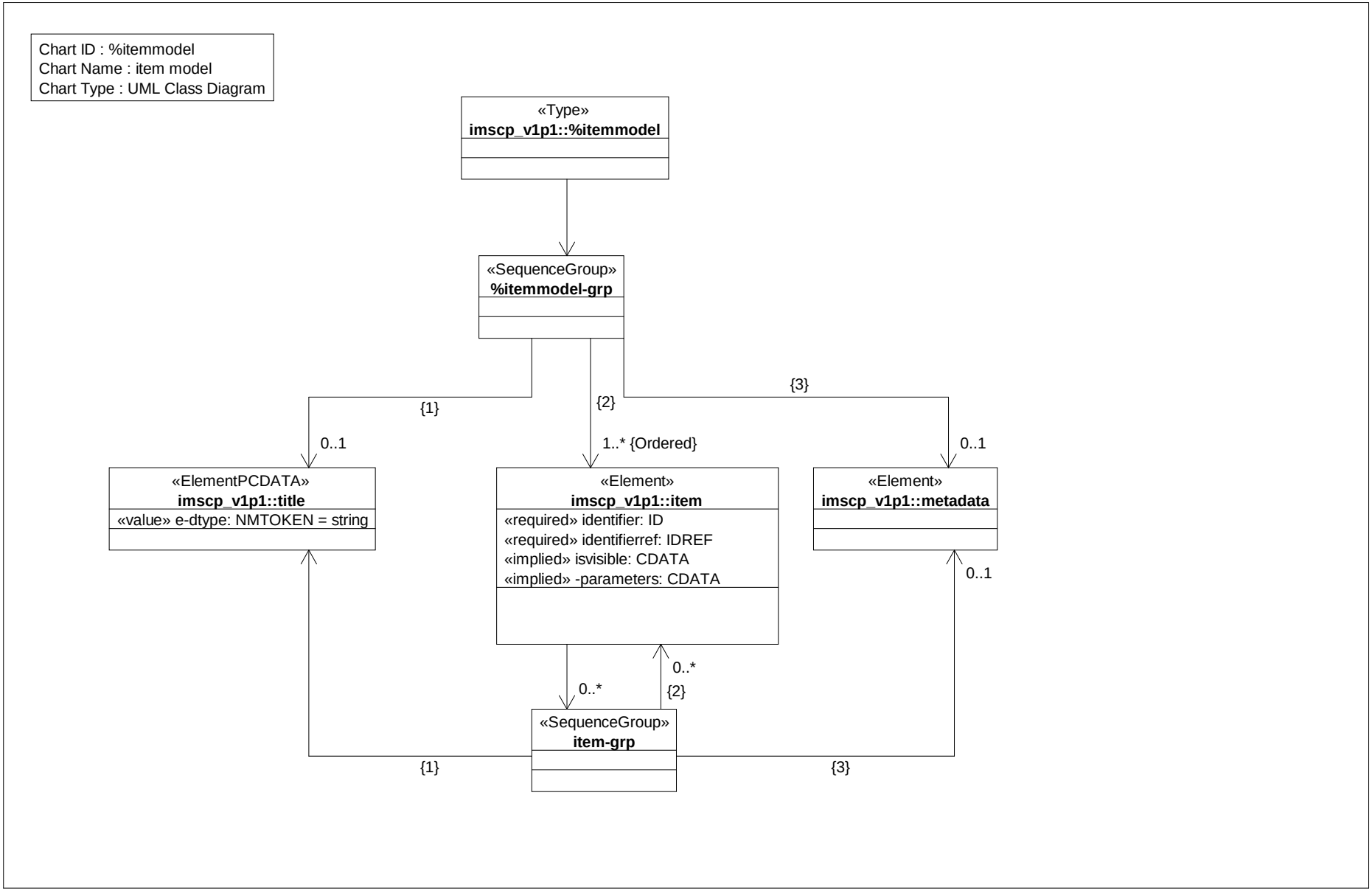


Chart ID : %thenmodel
Chart Name : %thenmodel
Chart Type : UML Class Diagram

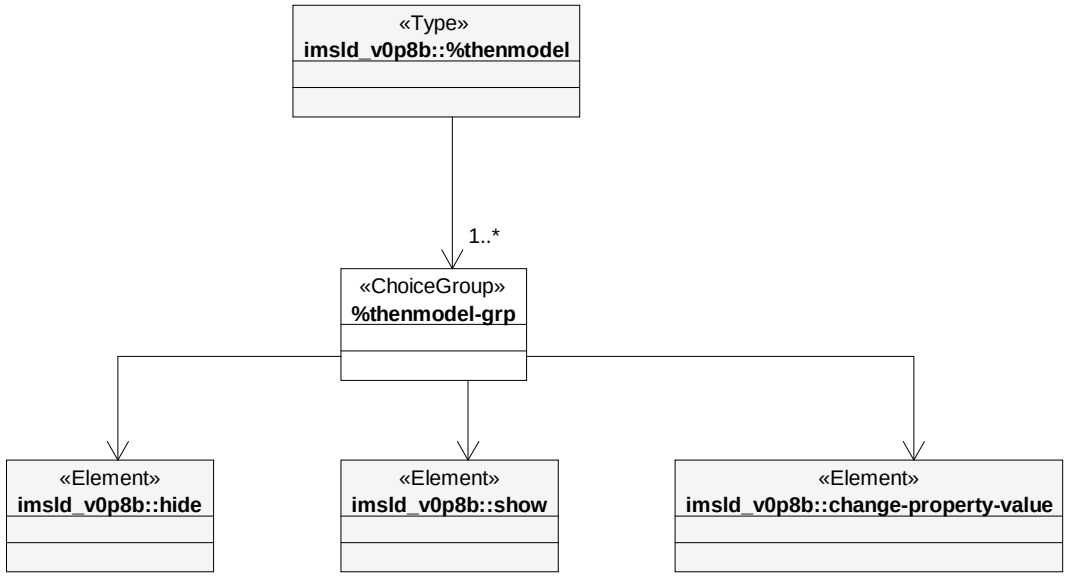
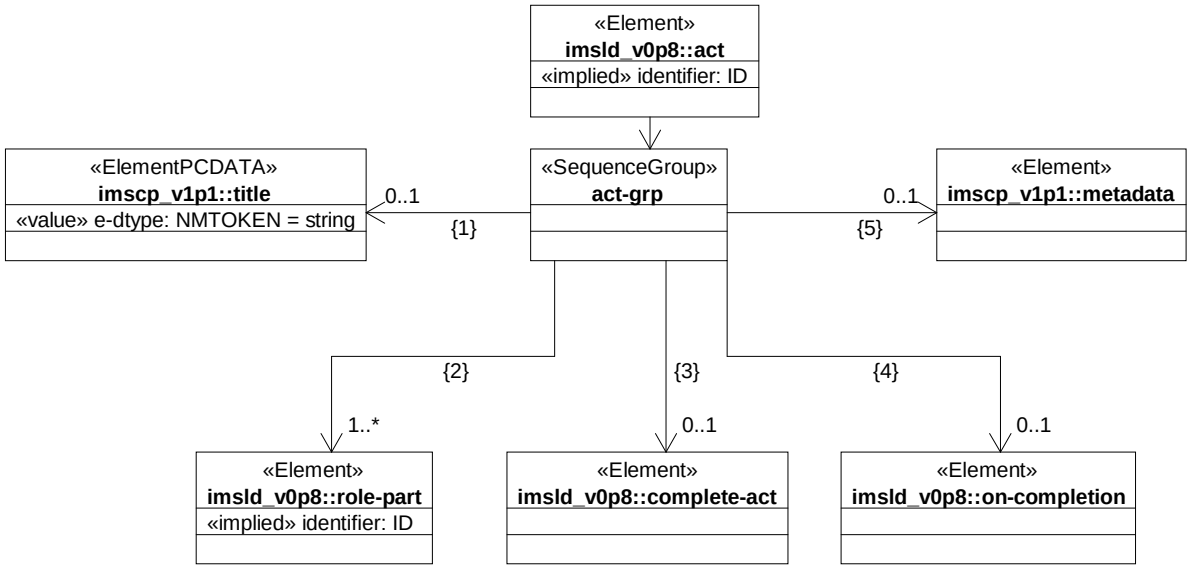


Chart ID : act
Chart Name : act
Chart Type : UML Class Diagram



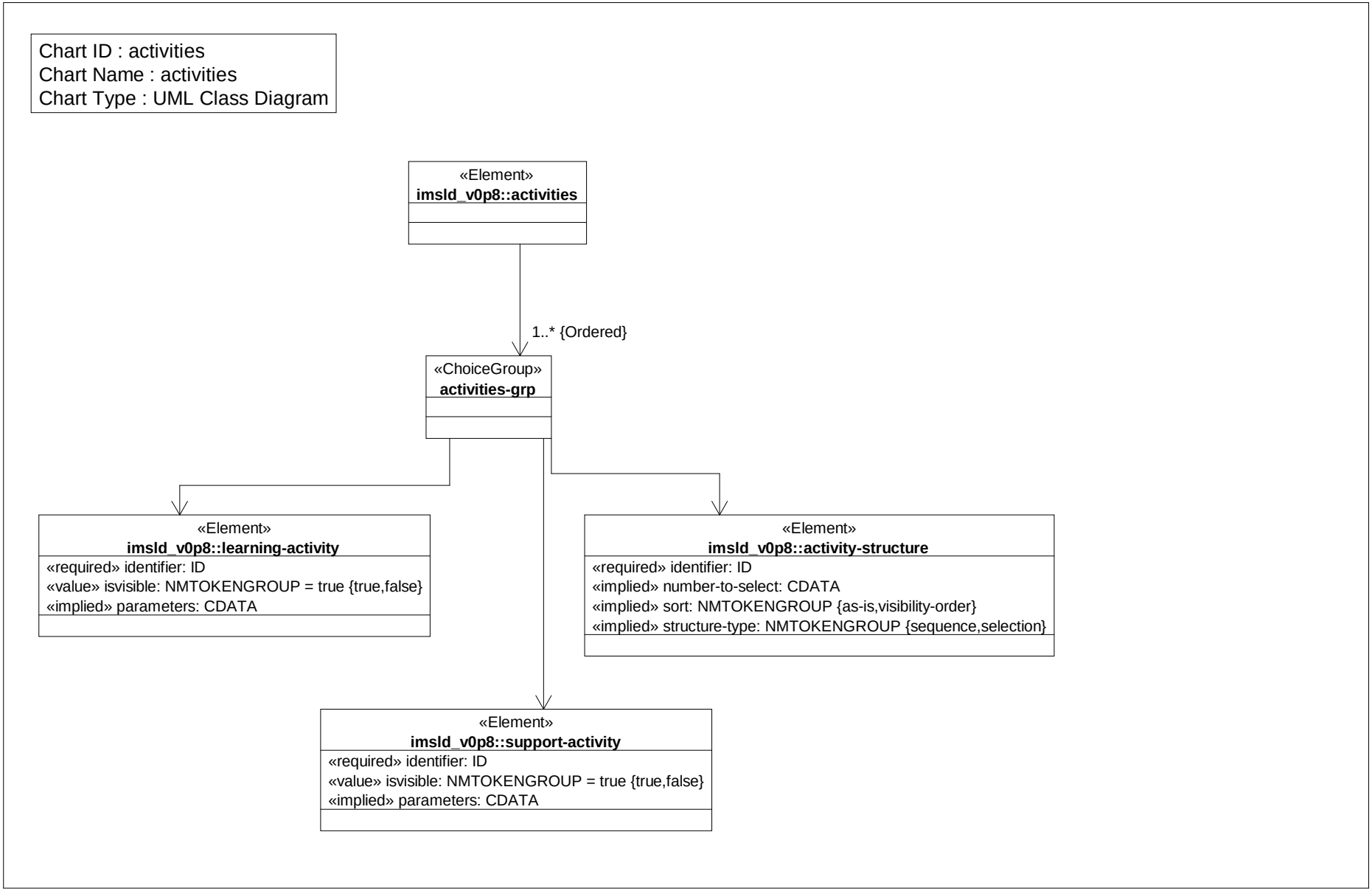


Chart ID : activities
Chart Name : activities
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::activities
1..* {Ordered}
«ChoiceGroup»
activities-grp
«Element»
imsld_v0p8::learning-activity
«required» identifier: ID
«value» isvisible: NMTOKENGROUP = true {true,false}
«implied» parameters: CDATA
«Element»
imsld_v0p8::activity-structure
«required» identifier: ID
«implied» number-to-select: CDATA
«implied» sort: NMTOKENGROUP {as-is,visibility-order}
«implied» structure-type: NMTOKENGROUP {sequence,selection}
«Element»
imsld_v0p8::support-activity
«required» identifier: ID
«value» isvisible: NMTOKENGROUP = true {true,false}
«implied» parameters: CDATA

Class Diagram: activty structure

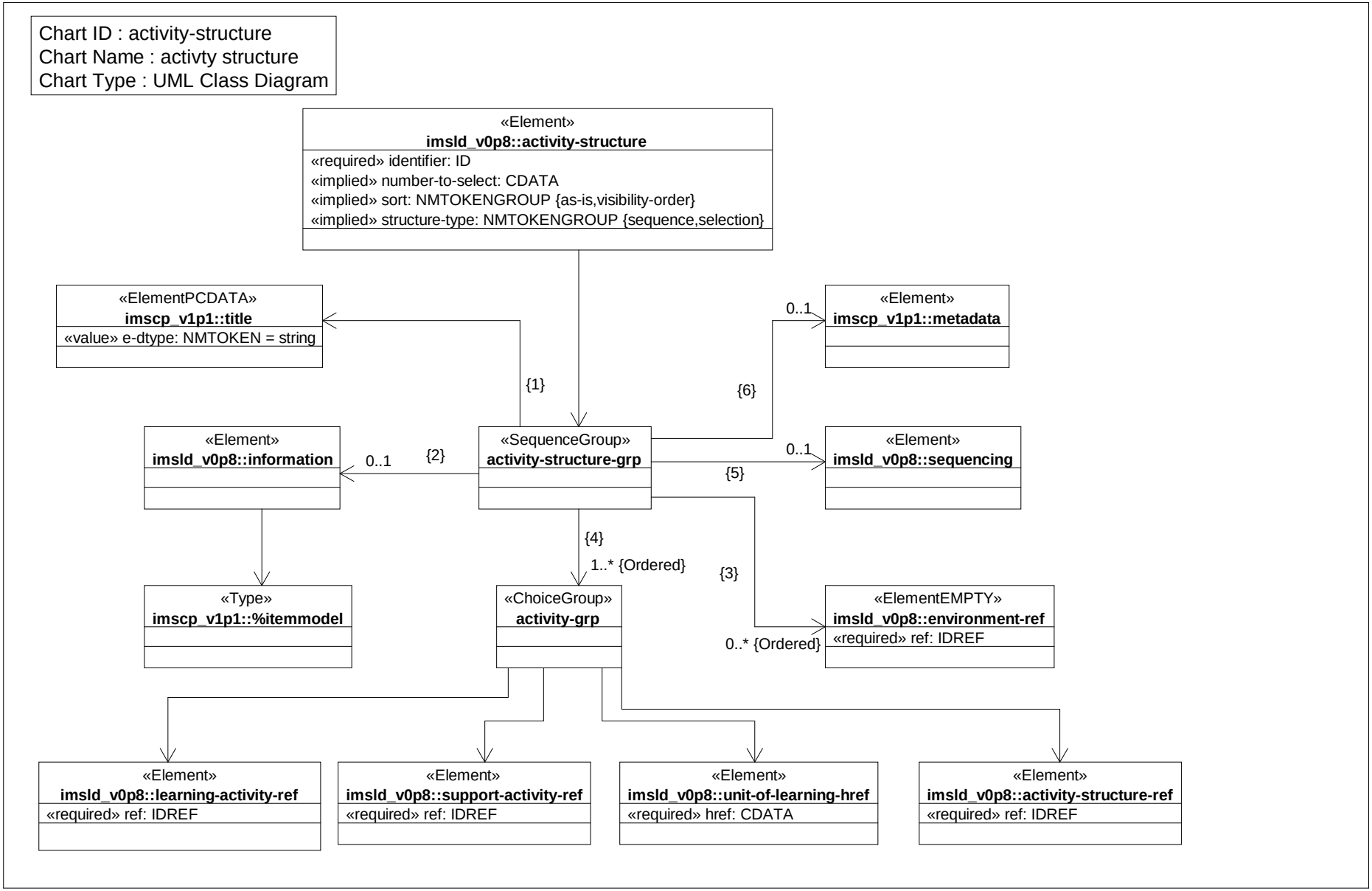


Chart ID : activity-structure-ref
Chart Name : activity structure ref
Chart Type : UML Class Diagram

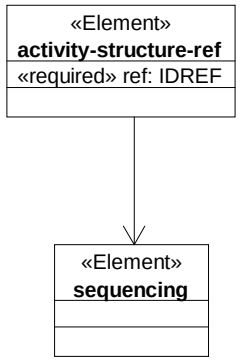
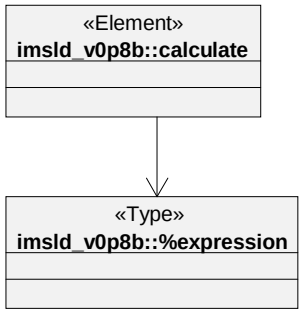
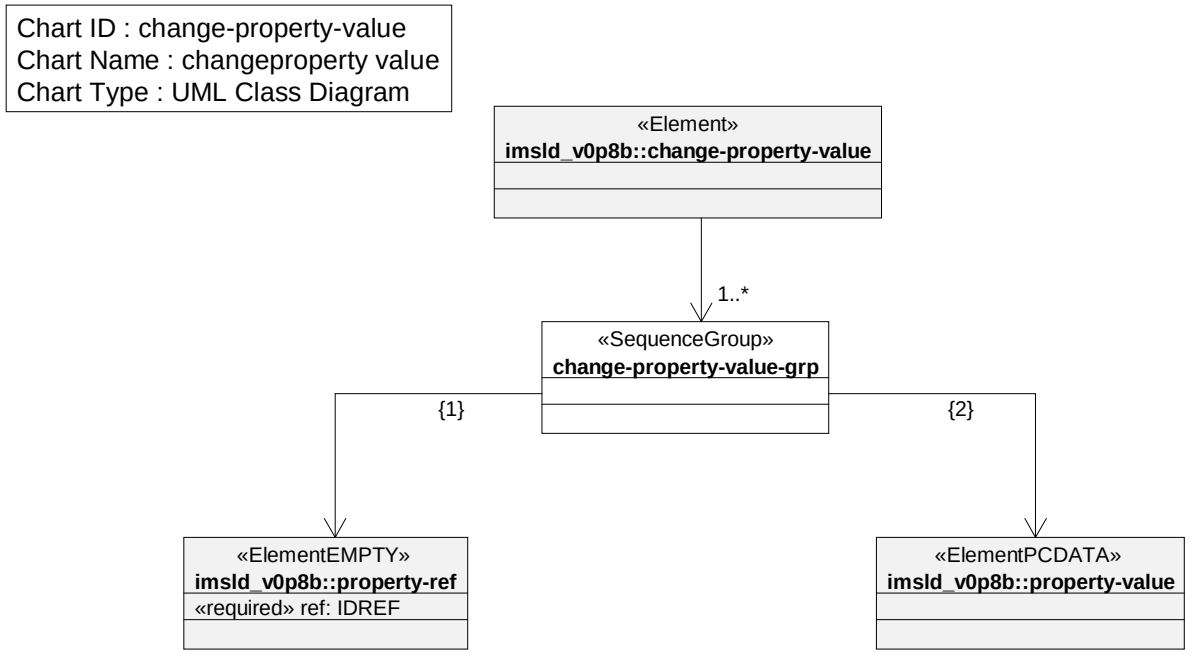


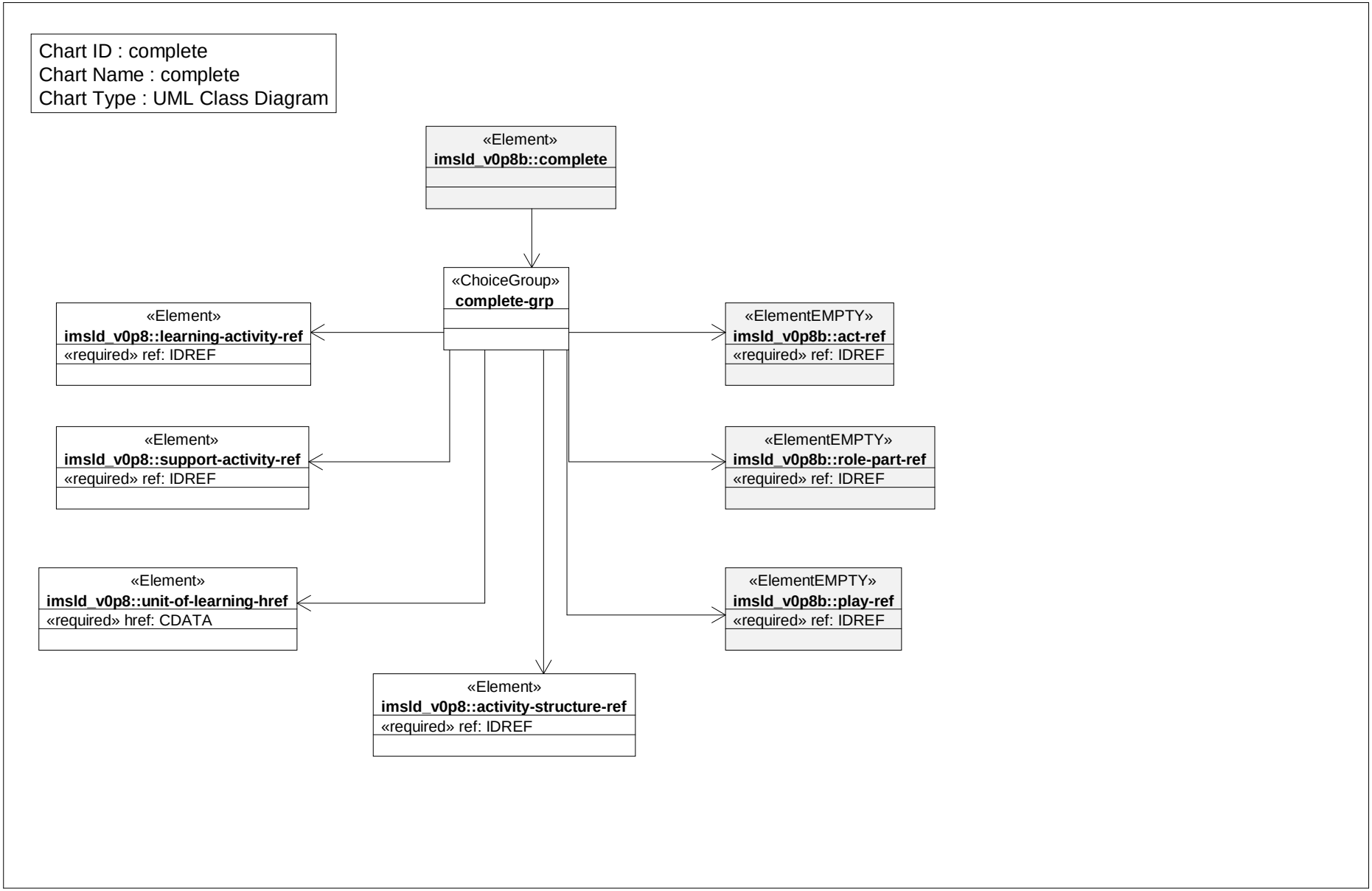
Chart ID : calculate
Chart Name : calculate
Chart Type : UML Class Diagram



Class Diagram: changeproperty value



Class Diagram: complete



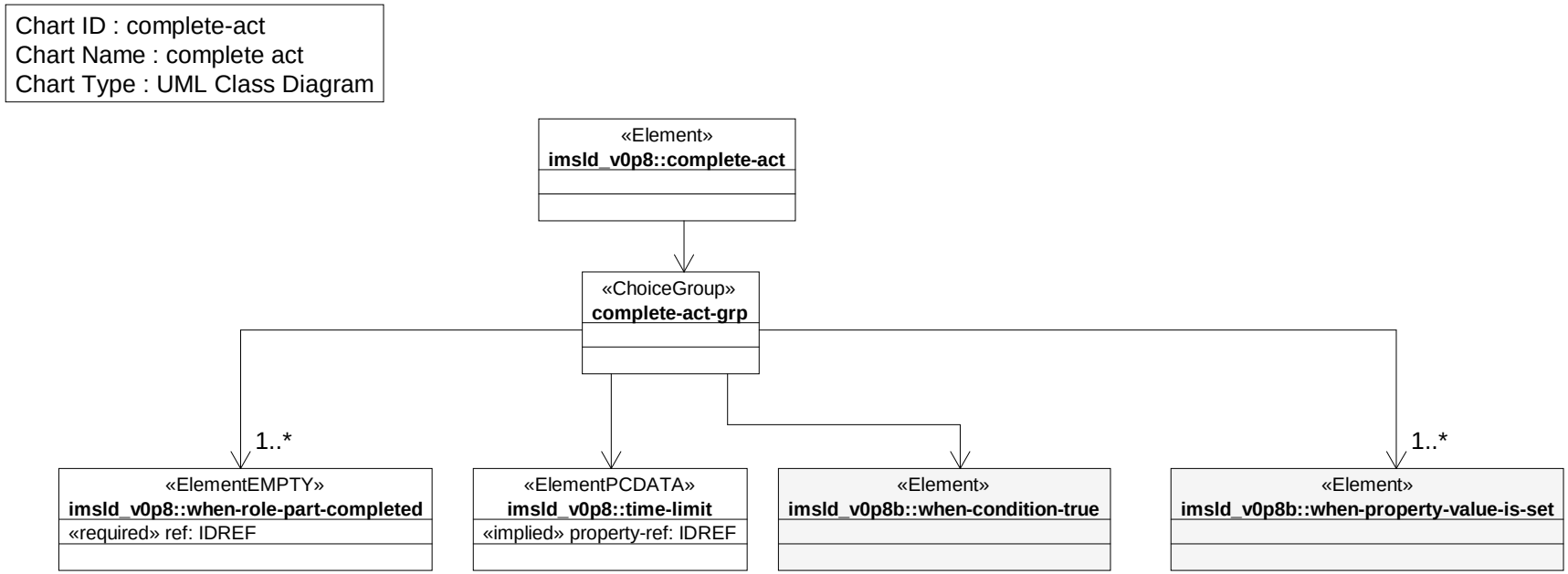


Chart ID : complete-act
Chart Name : complete act
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::complete-act
«ChoiceGroup»
complete-act-grp
1..*
1..*
«ElementEMPTY»
imsld_v0p8::when-role-part-completed
«required» ref: IDREF
«ElementPCDATA»
imsld_v0p8::time-limit
«implied» property-ref: IDREF
«Element»
imsld_v0p8b::when-condition-true
«Element»
imsld_v0p8b::when-property-value-is-set

Chart ID : complete-play
Chart Name : complete play
Chart Type : UML Class Diagram

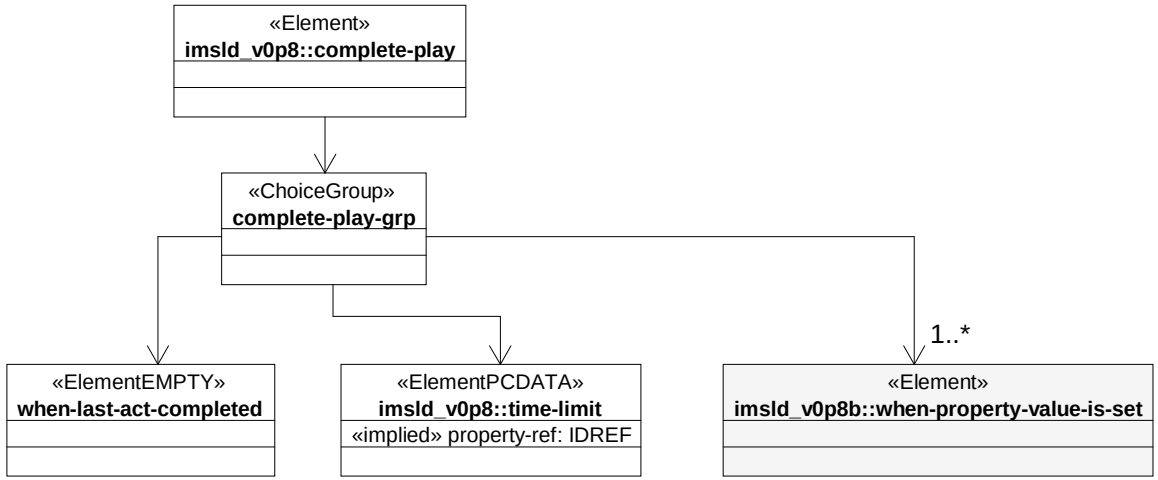
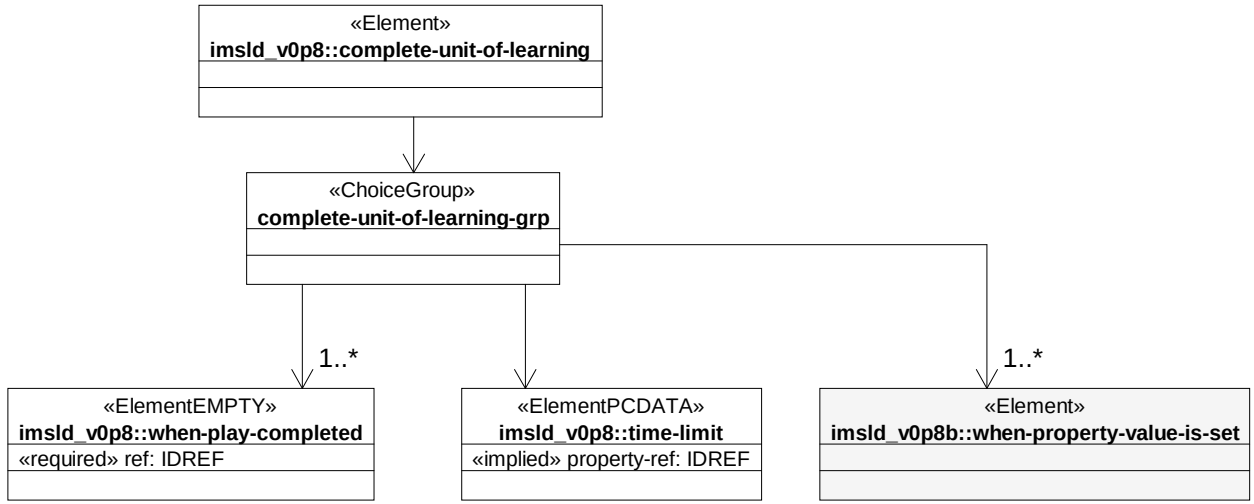
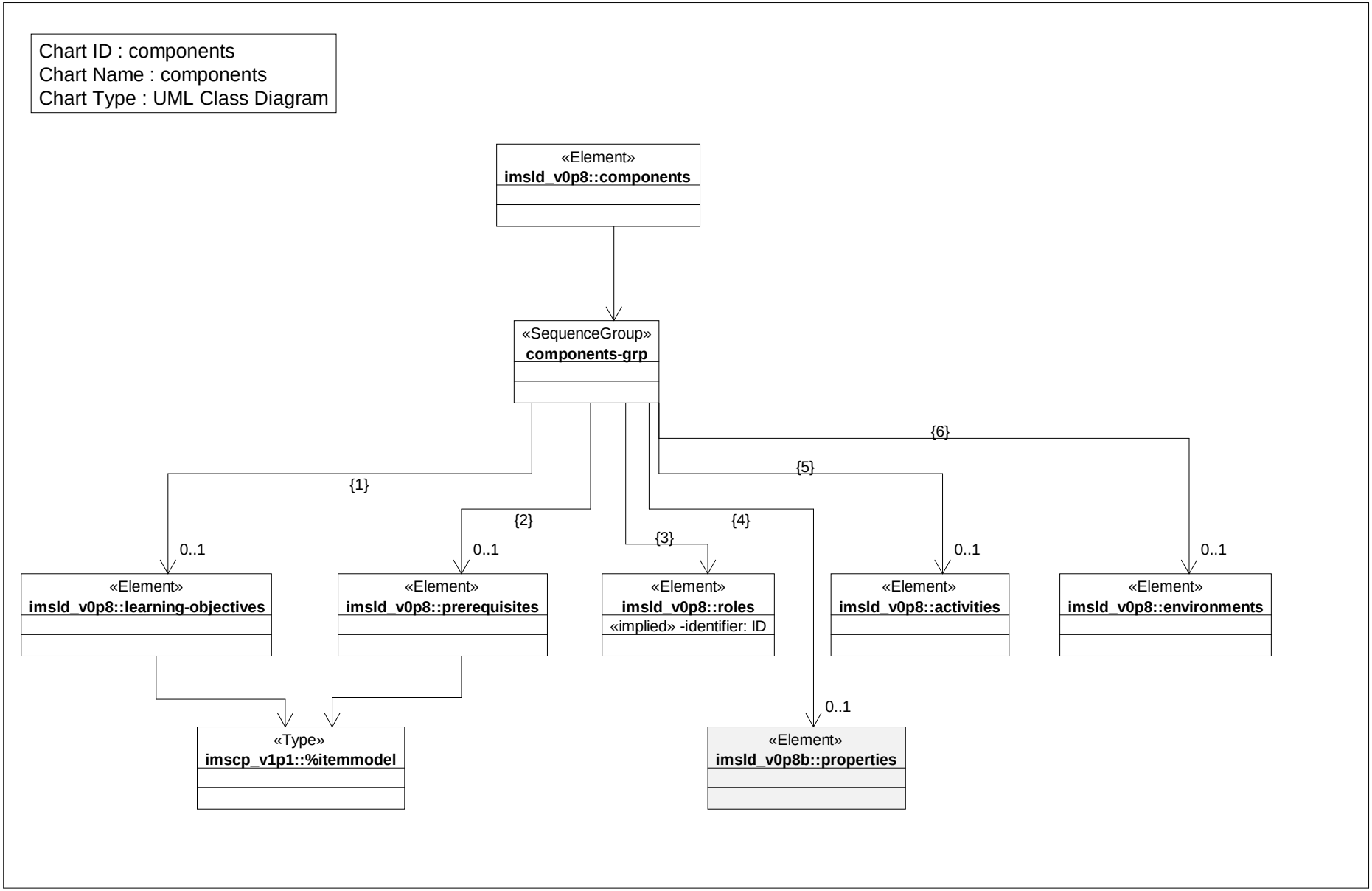


Chart ID : complete-unit-of-learning
Chart Name : complete unit of learning
Chart Type : UML Class Diagram



Class Diagram: components



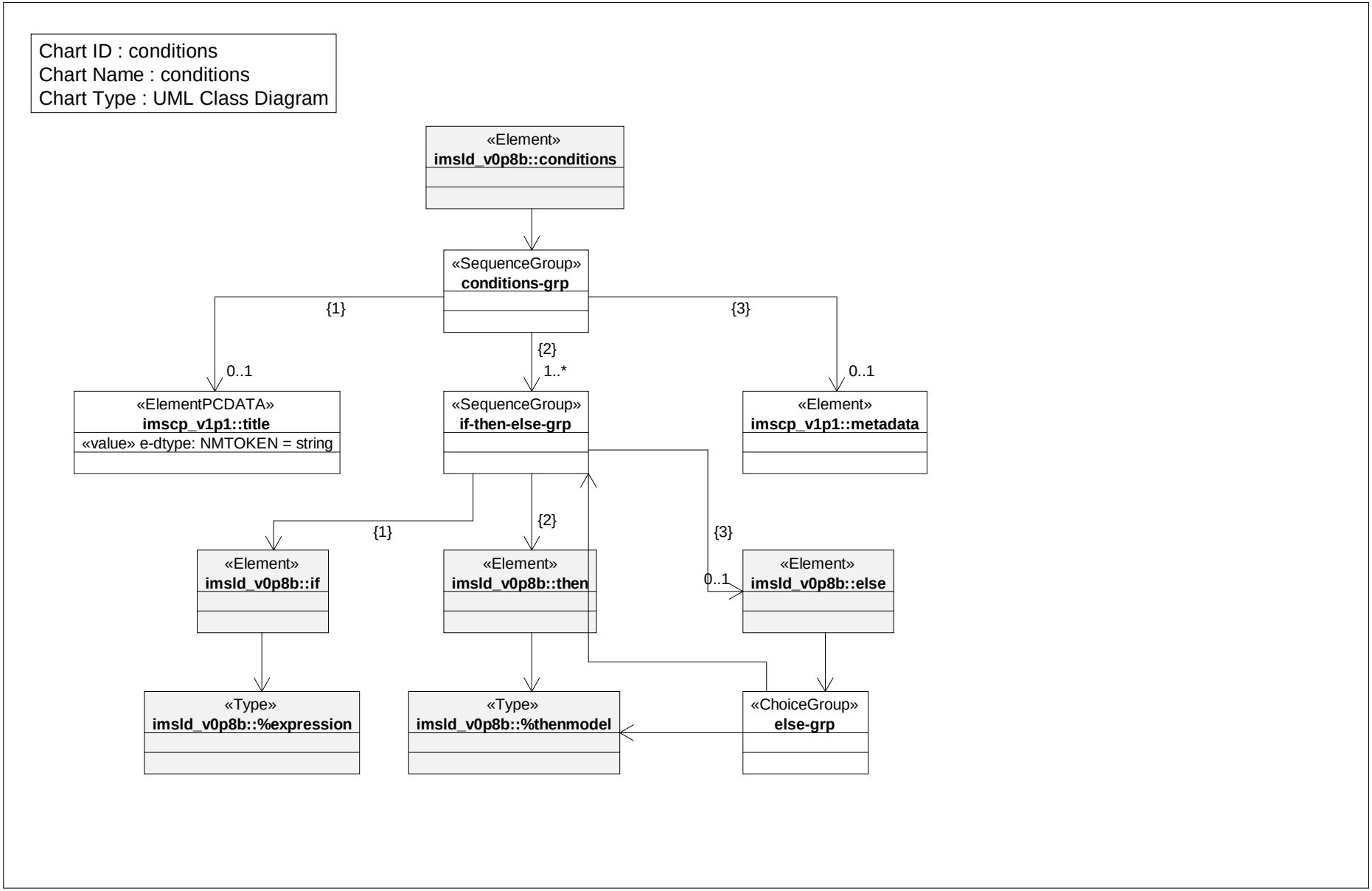
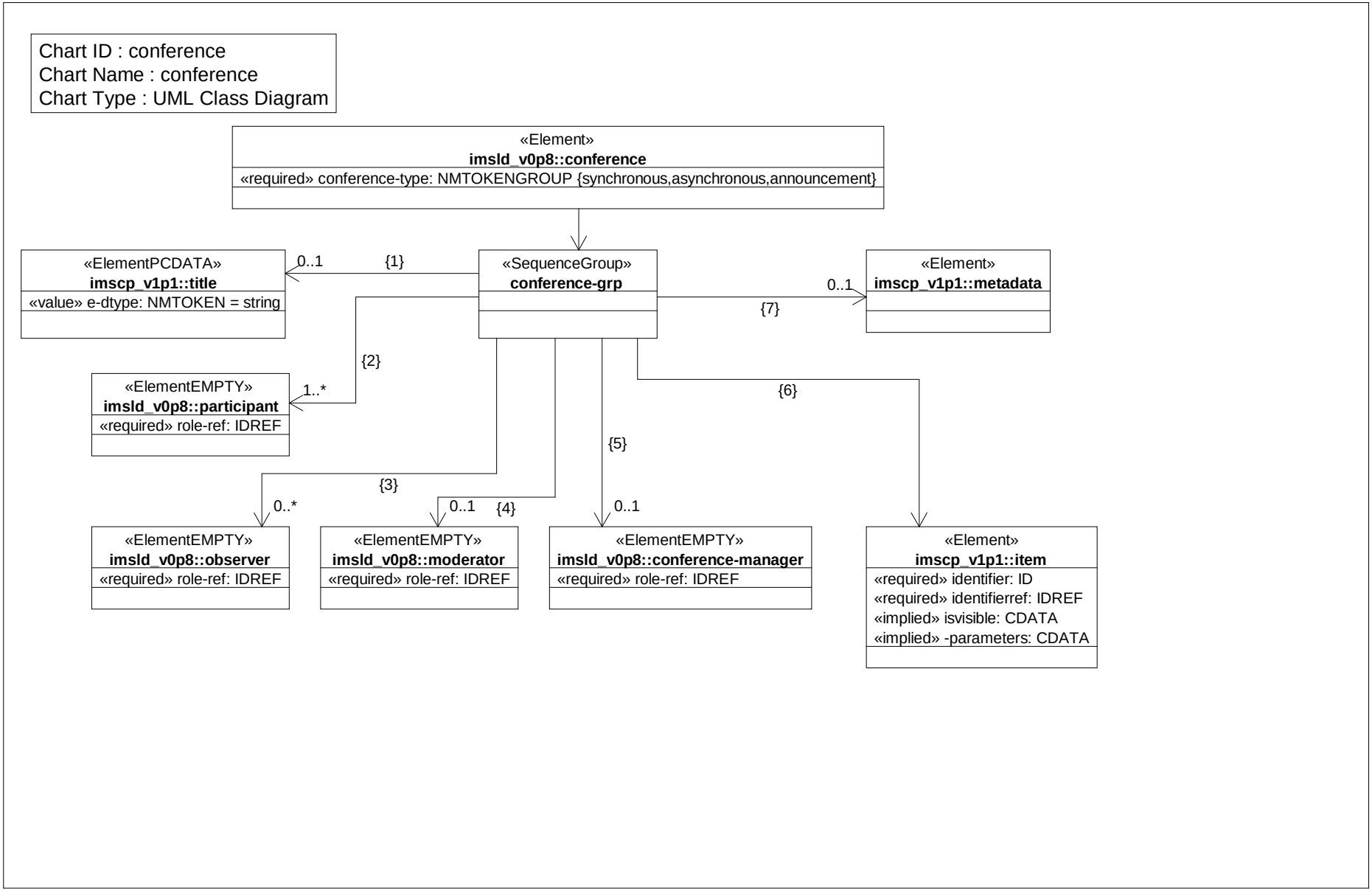
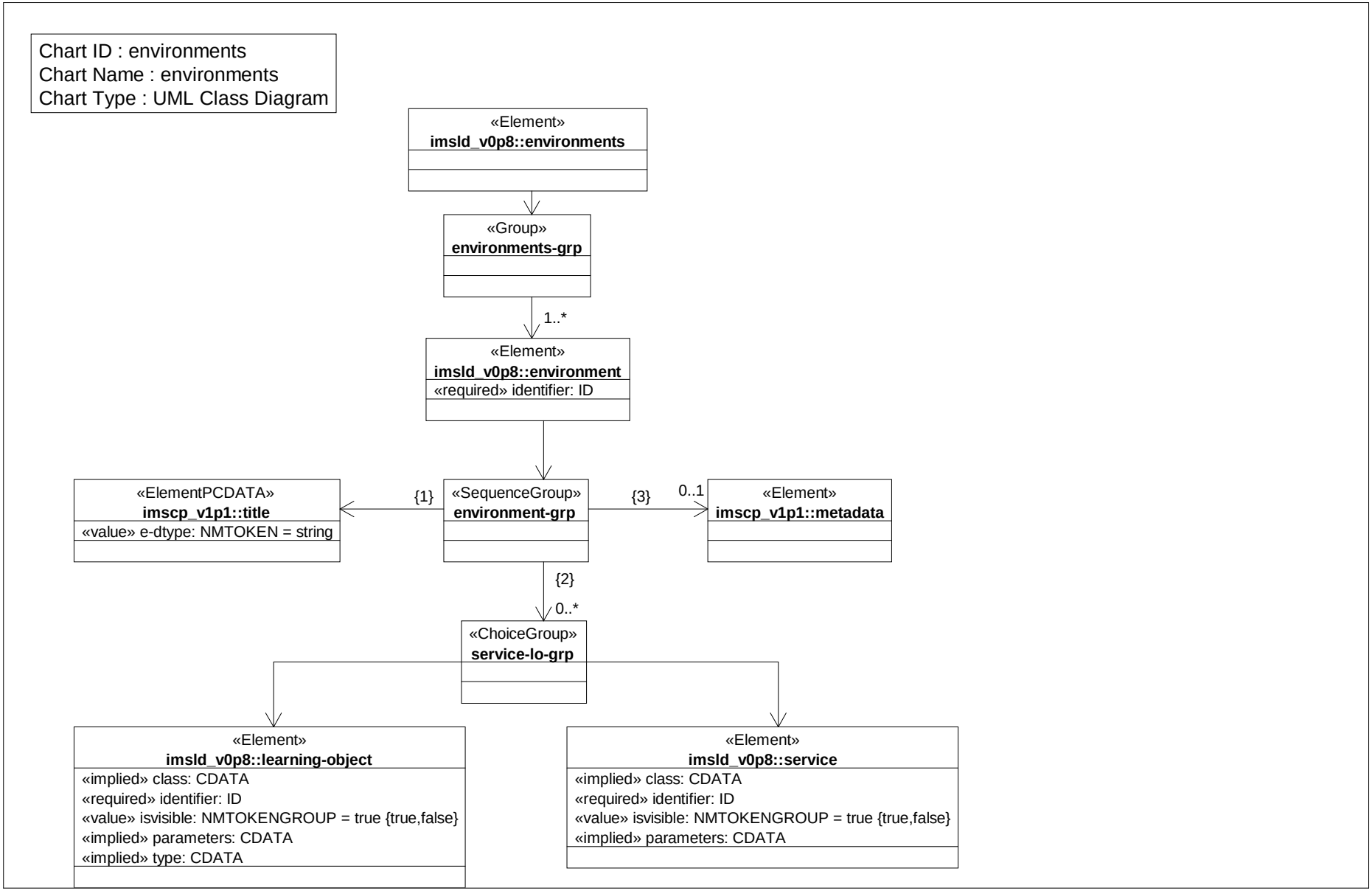


Chart ID : conditions
Chart Name : conditions
Chart Type : UML Class Diagram
«Element»
imsld_v0p8b::conditions
«SequenceGroup»
conditions-grp
{1}
{2}
{3}
0..1
1..*
0..1
«ElementPCDATA»
imscp_v1p1::title
«value» e-dtype: NMTOKEN = string
«SequenceGroup»
if-then-else-grp
«Element»
imscp_v1p1::metadata
{1}
{2}
{3}
0..1
«Element»
imsld_v0p8b::if
«Element»
imsld_v0p8b::then
«Element»
imsld_v0p8b::else
«Type»
imsld_v0p8b::%expression
«Type»
imsld_v0p8b::%thenmodel
«ChoiceGroup»
else-grp

Class Diagram: conference



Class Diagram: environments



Class Diagram: global definition

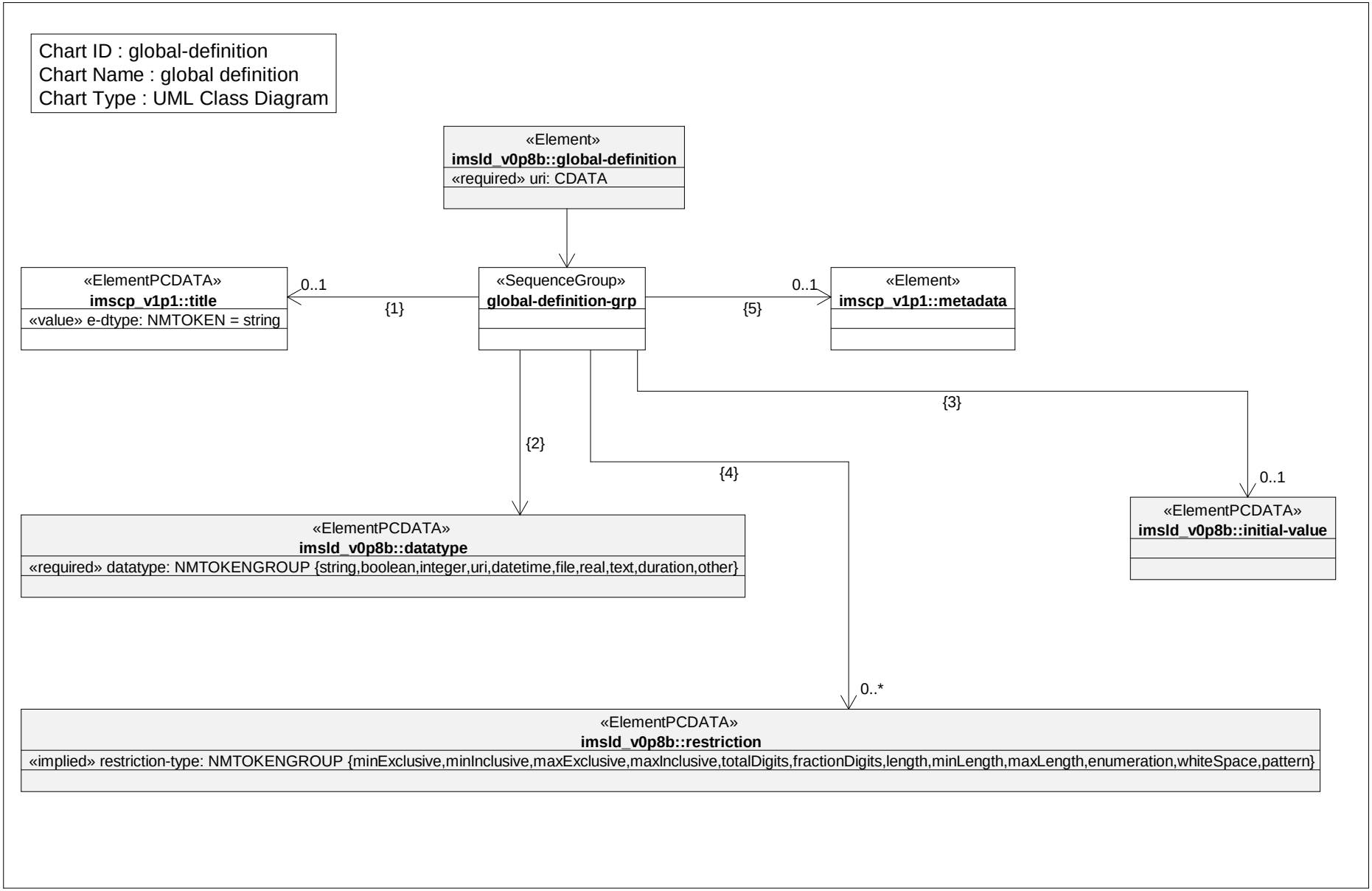


Chart ID : global-elements
Chart Name : global elements
Chart Type : UML Class Diagram

«ElementEMPTY»
imsld_global::view-property-group
«implied» href: CDATA
«value» property-of: NMTOKENGROUP = self {self,supported-person}
«implied» ref: IDREF
«value» view: NMTOKENGROUP = value {value,title-value}

«ElementEMPTY»
imsld_global::view-property
«implied» href: CDATA
«value» property-of: NMTOKENGROUP = self {self,supported-person}
«implied» ref: IDREF
«value» view: NMTOKENGROUP = value {value,title-value}

«ElementEMPTY»
imsld_global::set-property
«implied» href: CDATA
«implied» max-transactions: CDATA
«value» property-of: NMTOKENGROUP = self {self,supported-person}
«implied» ref: IDREF
«implied» transaction-type: CDATA
«value» view: NMTOKENGROUP = value {value,title-value}

«ElementEMPTY»
imsld_global::set-property-group
«implied» href: CDATA
«implied» max-transactions: CDATA
«value» property-of: NMTOKENGROUP = self {self,supported-person}
«implied» ref: IDREF
«implied» transaction-type: CDATA
«value» view: NMTOKENGROUP = value {value,title-value}

Chart ID : globpers-property
Chart Name : global personal property
Chart Type : UML Class Diagram

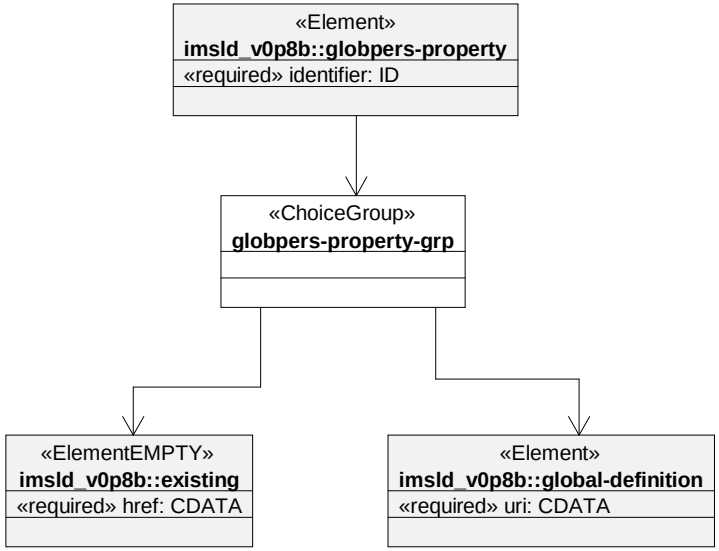
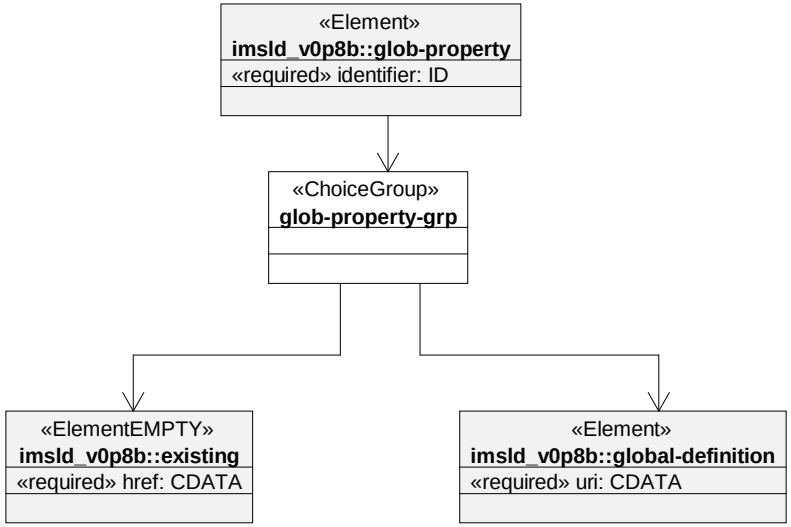
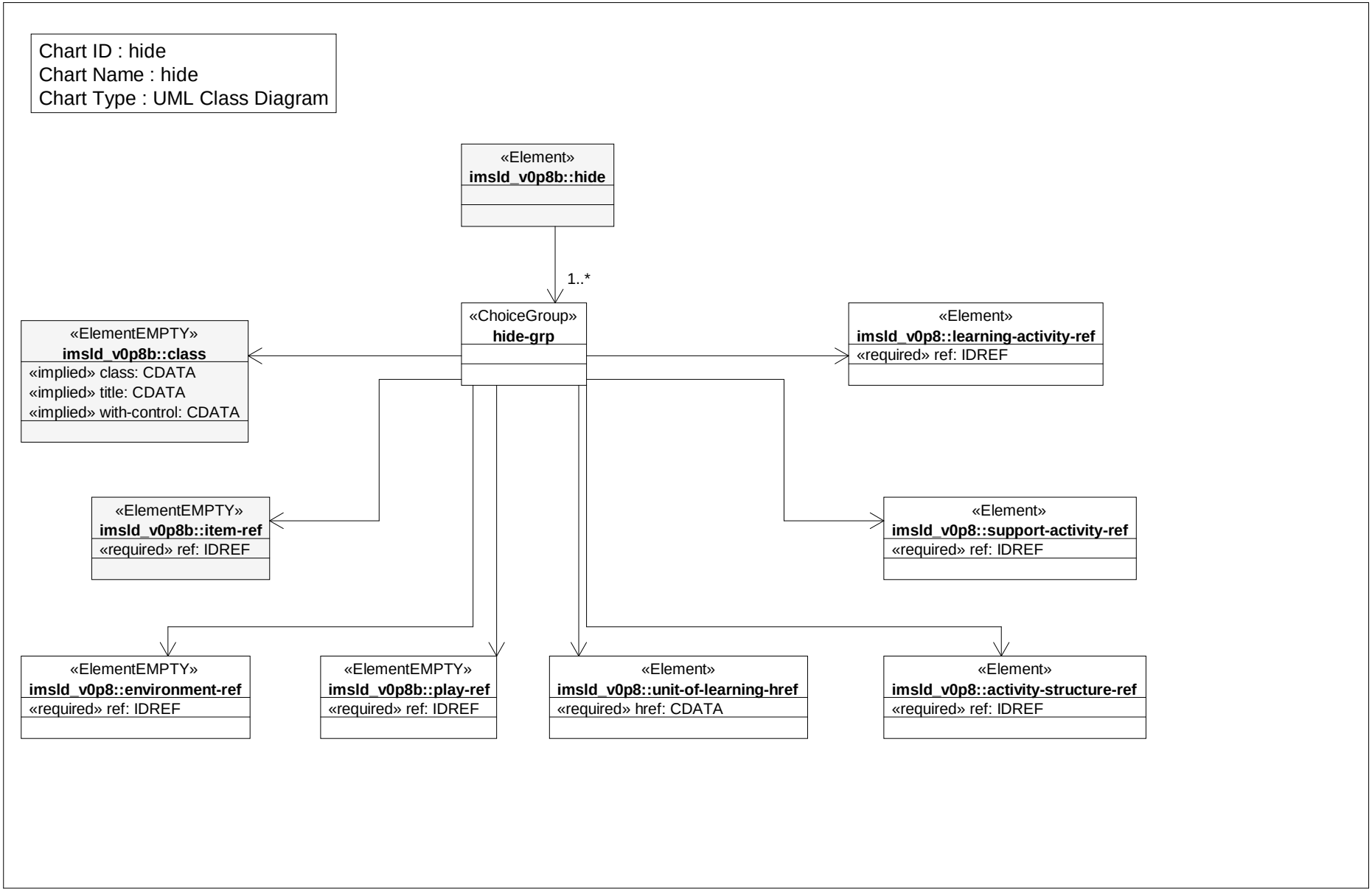


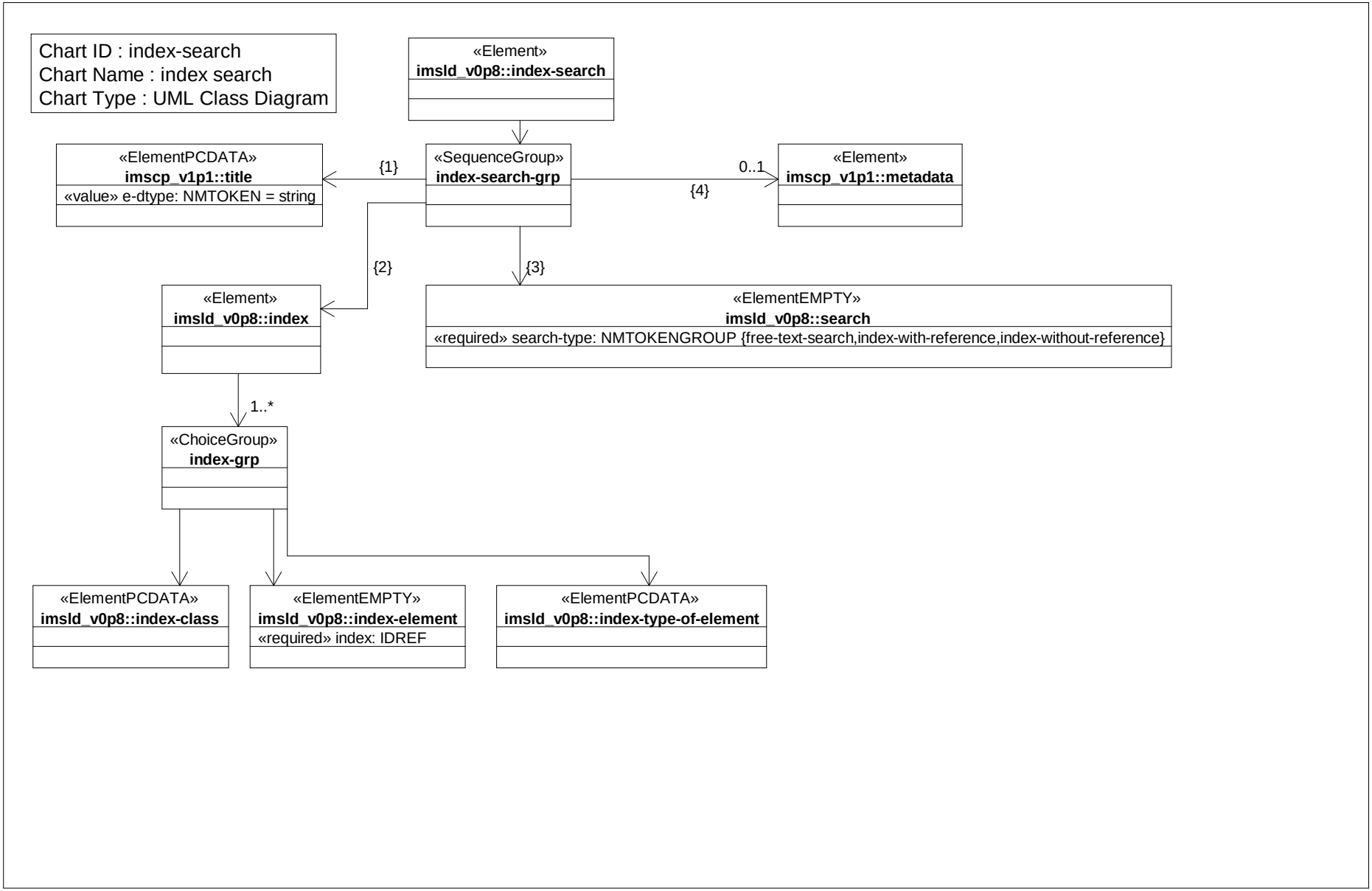
Chart ID : glob-property
Chart Name : global property
Chart Type : UML Class Diagram



Class Diagram: hide



Class Diagram: index search



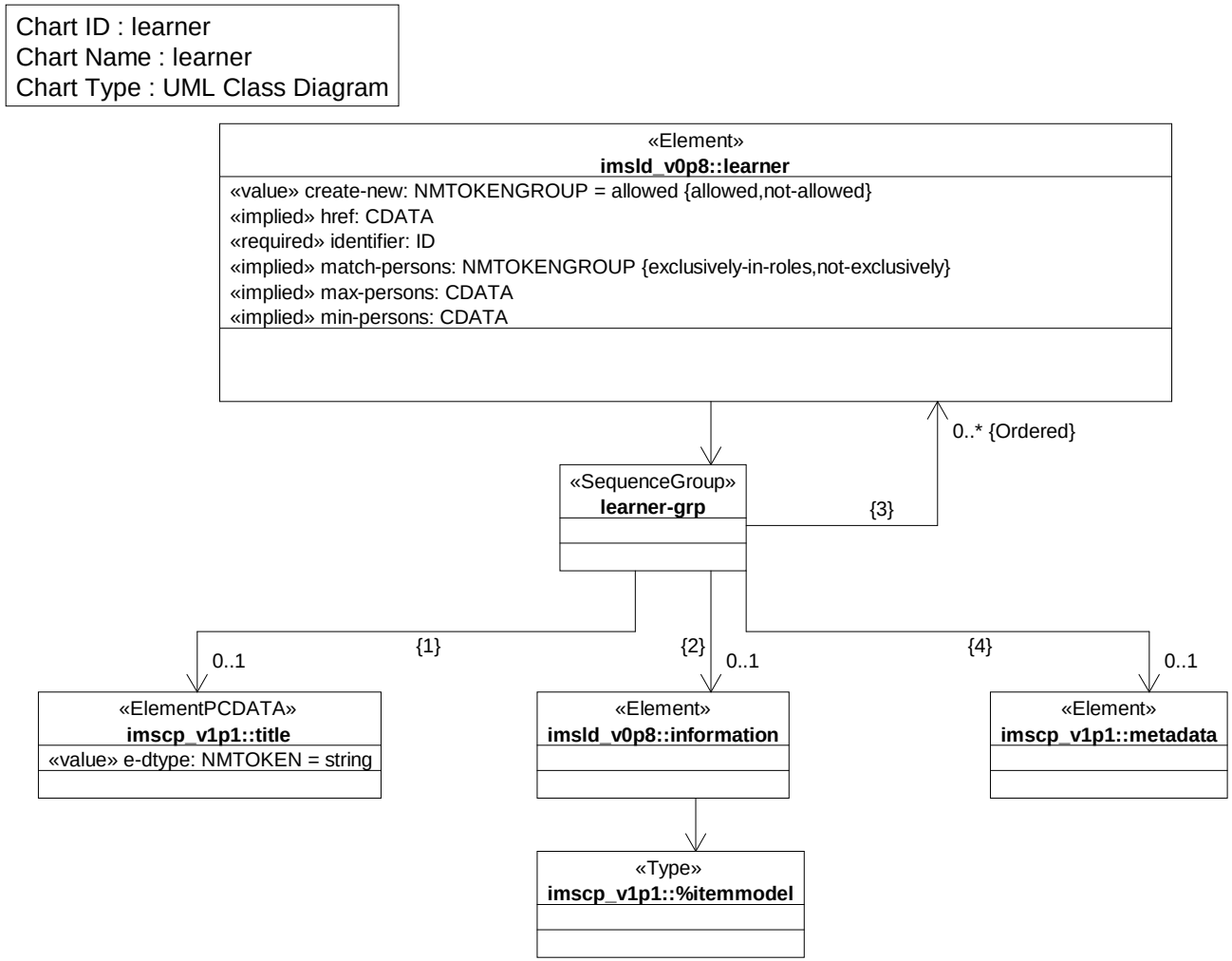


Chart ID : learner
Chart Name : learner
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::learner
«value» create-new: NMTOKENGROUP = allowed {allowed,not-allowed}
«implied» href: CDATA
«required» identifier: ID
«implied» match-persons: NMTOKENGROUP {exclusively-in-roles,not-exclusively}
«implied» max-persons: CDATA
«implied» min-persons: CDATA
0..* {Ordered}
«SequenceGroup»
learner-grp
{3}
{1}
0..1
{2}
0..1
{4}
0..1
«ElementPCDATA»
imscp_v1p1::title
«value» e-dtype: NMTOKEN = string
«Element»
imsld_v0p8::information
«Element»
imscp_v1p1::metadata
«Type»
imscp_v1p1::%itemmodel

Class Diagram: learning activity

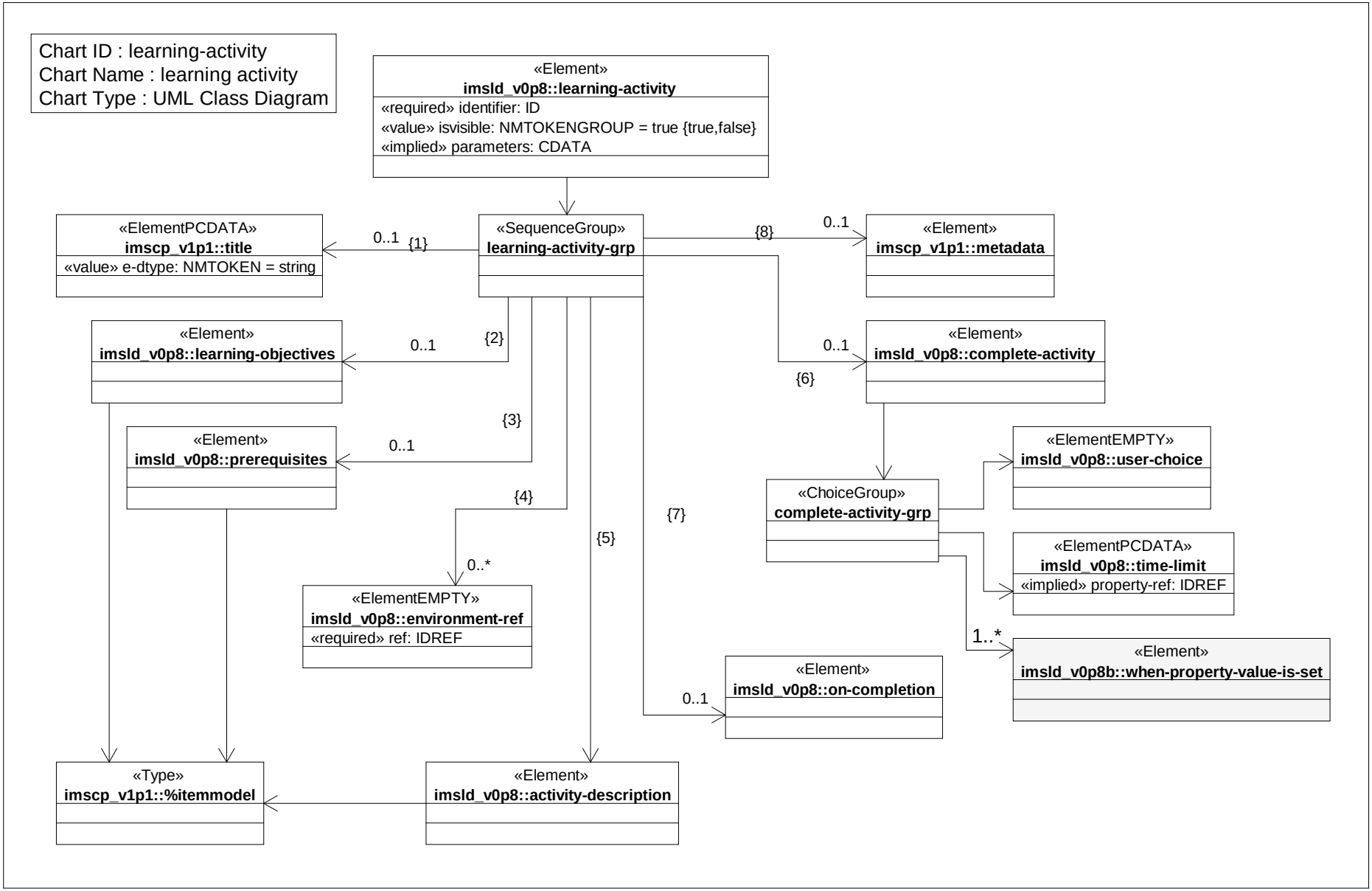


Chart ID : learning-activity-ref
Chart Name : learning activity ref
Chart Type : UML Class Diagram

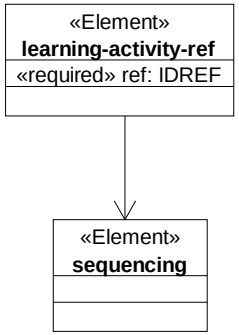


Chart ID : learning-design
Chart Name : learning design
Chart Type : UML Class Diagram

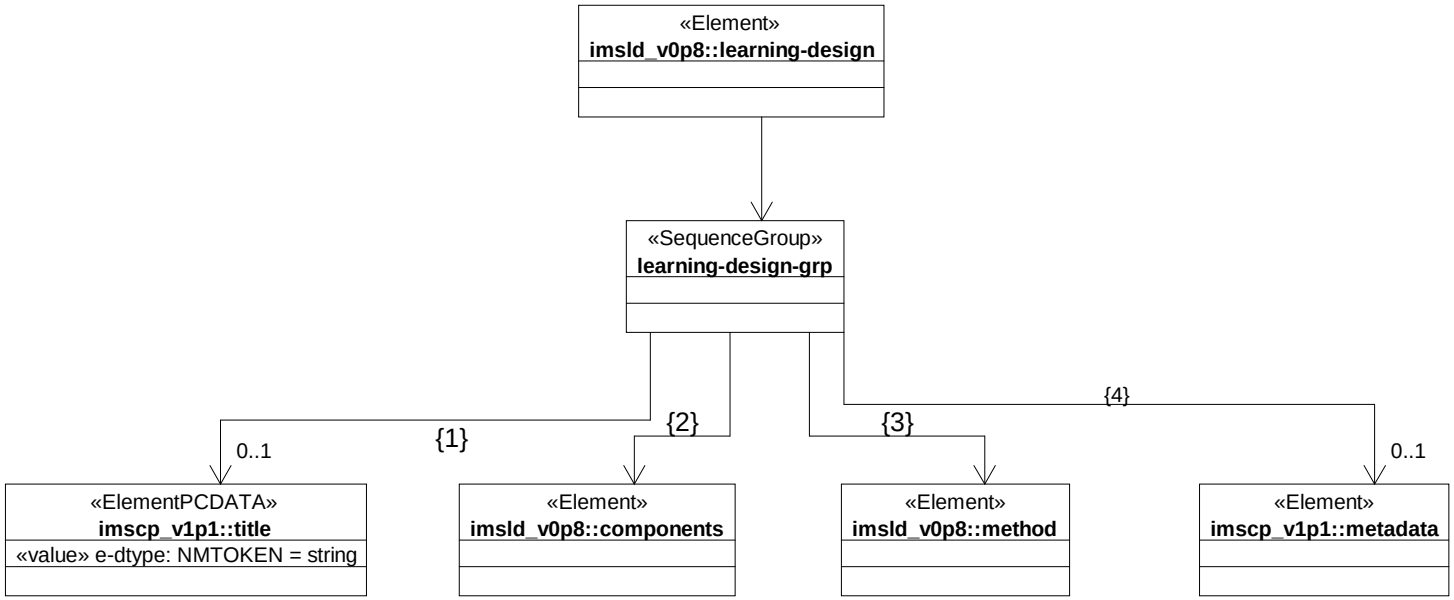
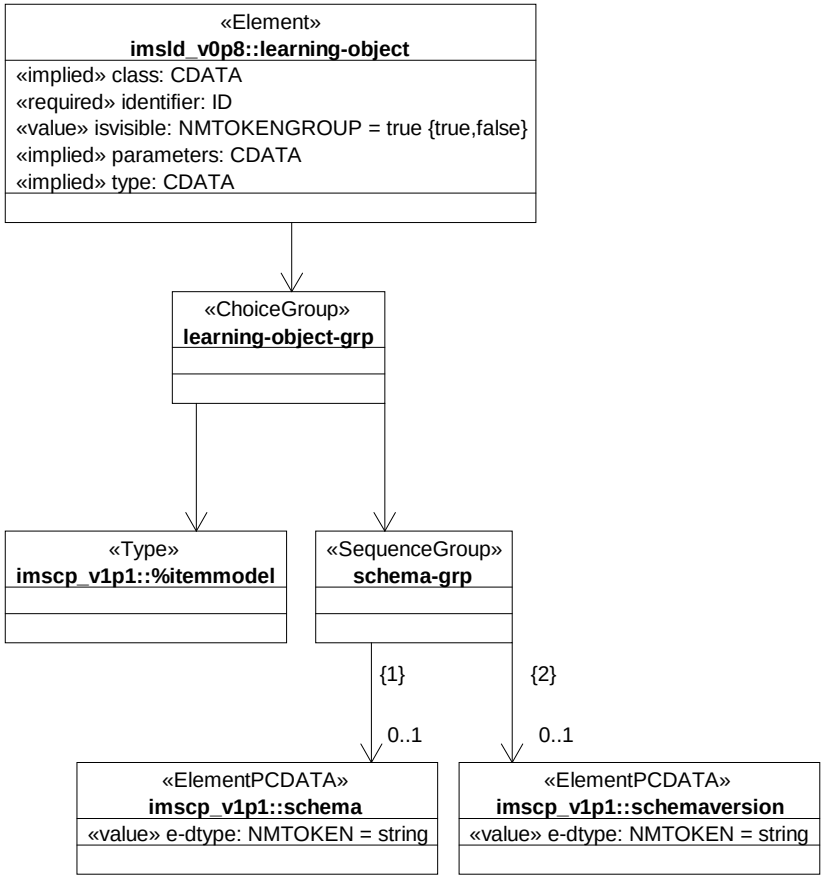
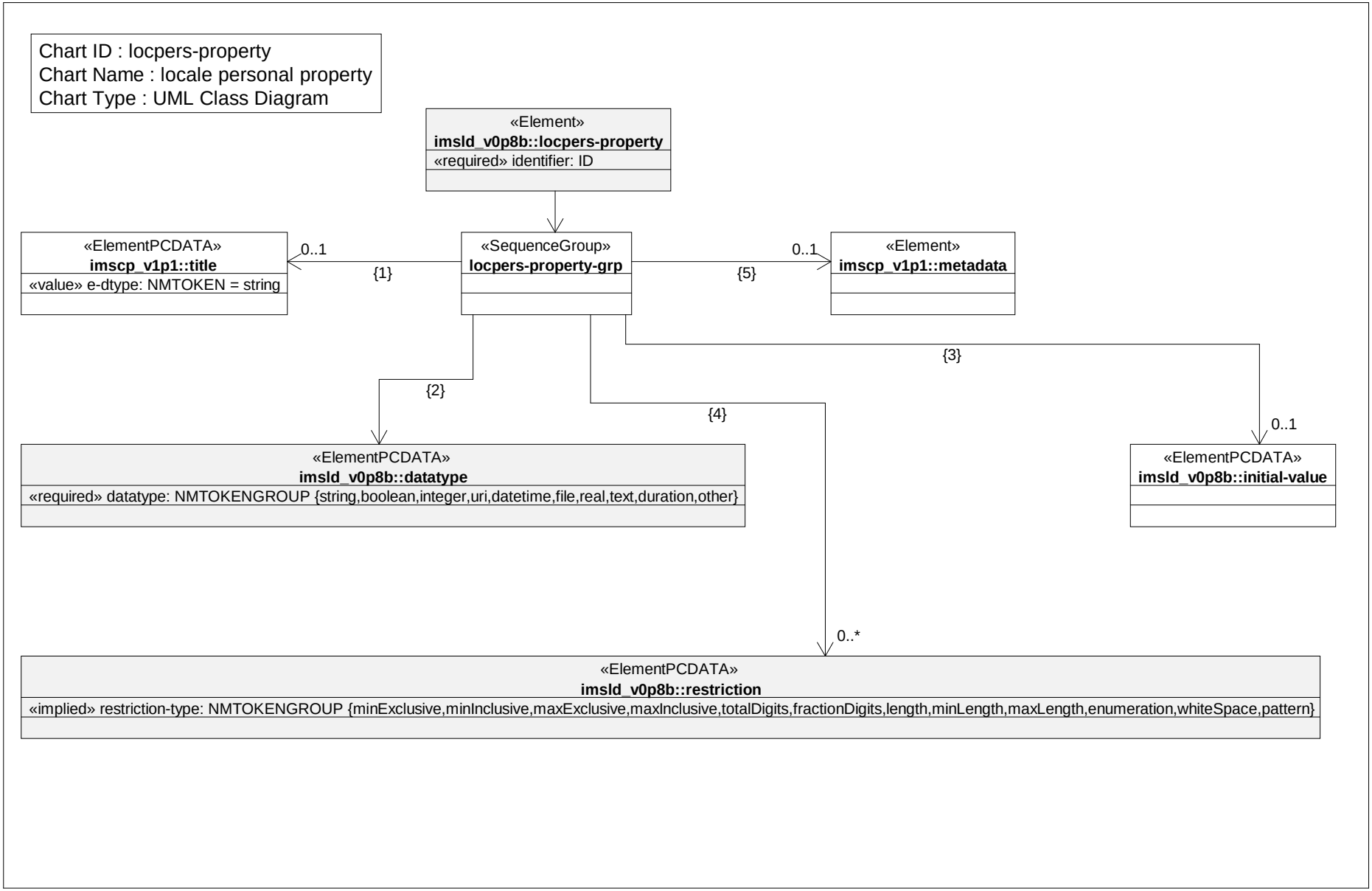


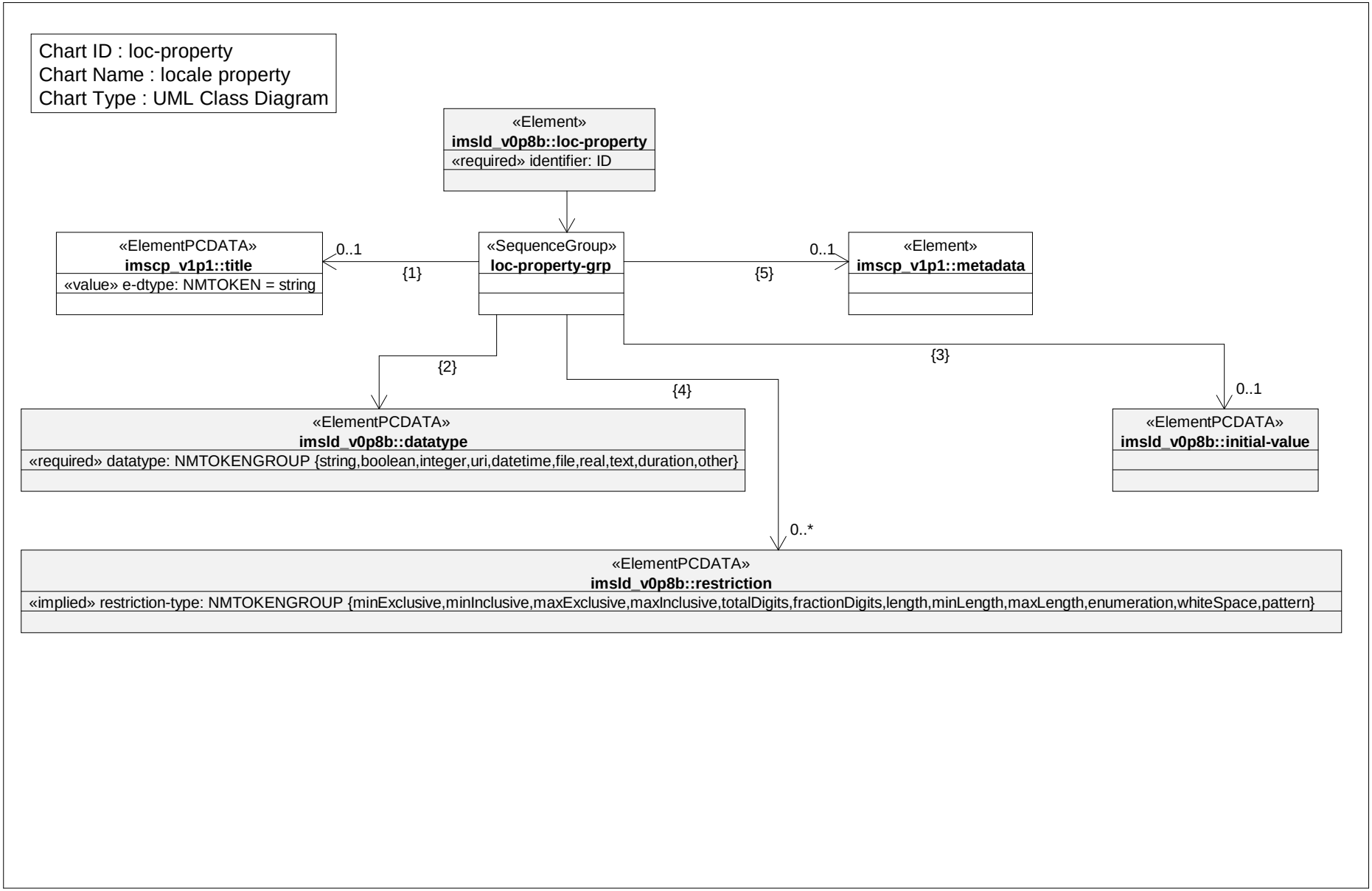
Chart ID : learning-object
Chart Name : learning object
Chart Type : UML Class Diagram



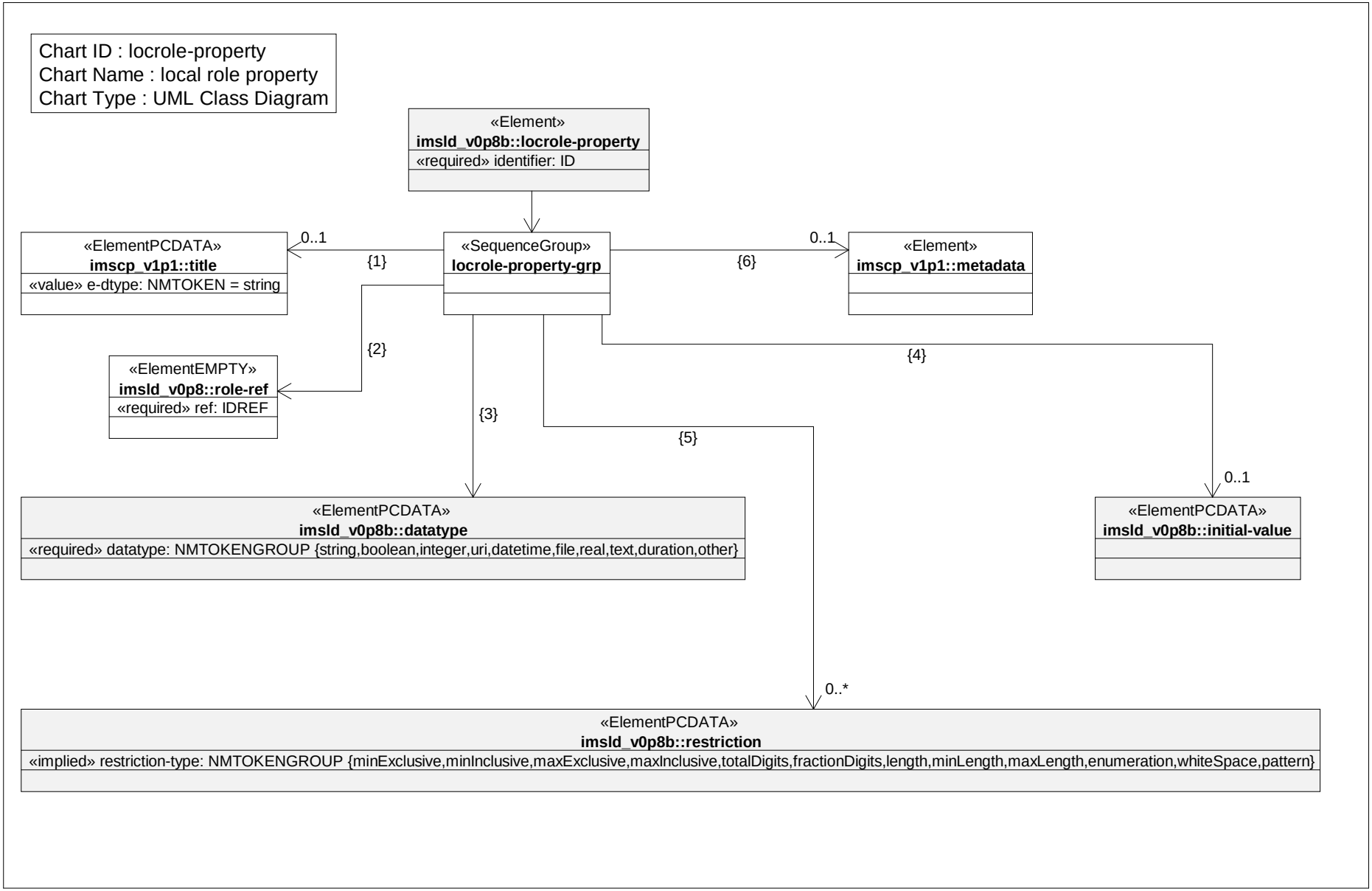
Class Diagram: locale personal property



Class Diagram: locale property



Class Diagram: local role property



Class Diagram: manifest

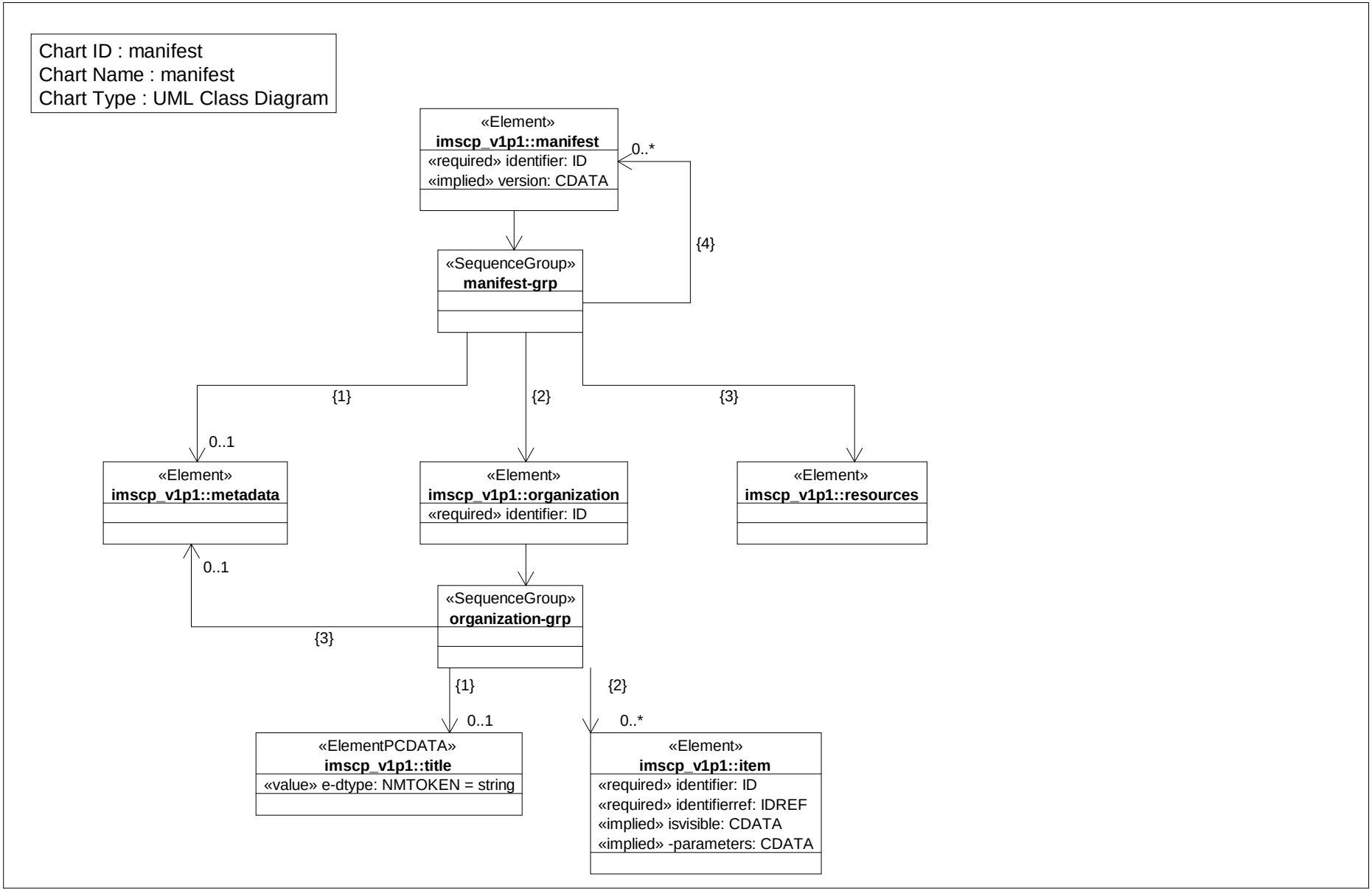
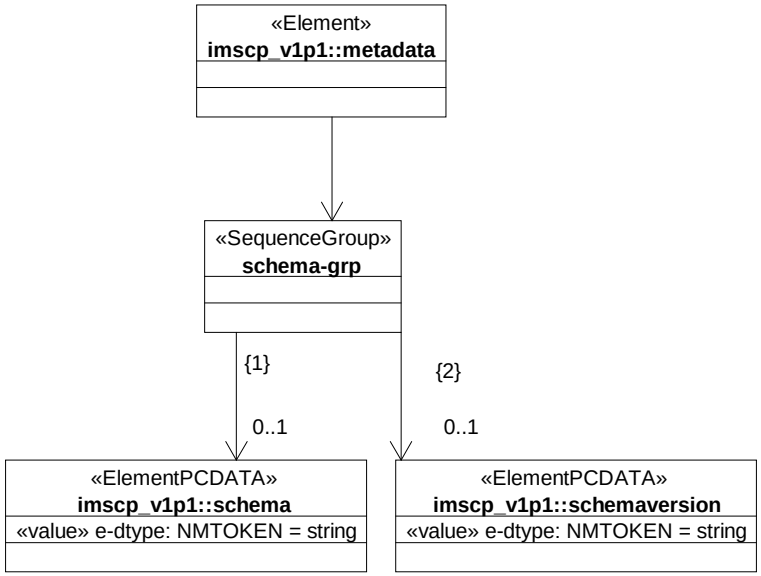


Chart ID : metadata
Chart Name : metadata
Chart Type : UML Class Diagram



Class Diagram: method

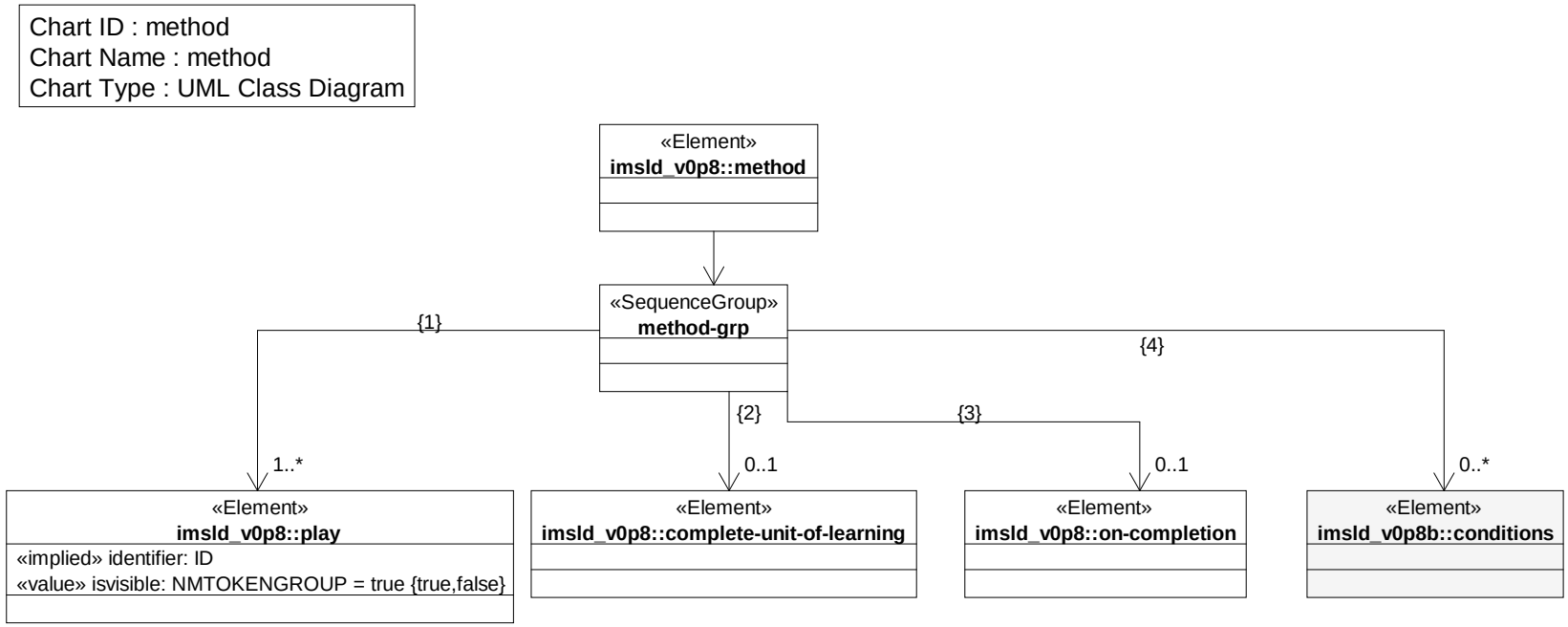


Chart ID : monitor
Chart Name : monitor
Chart Type : UML Class Diagram

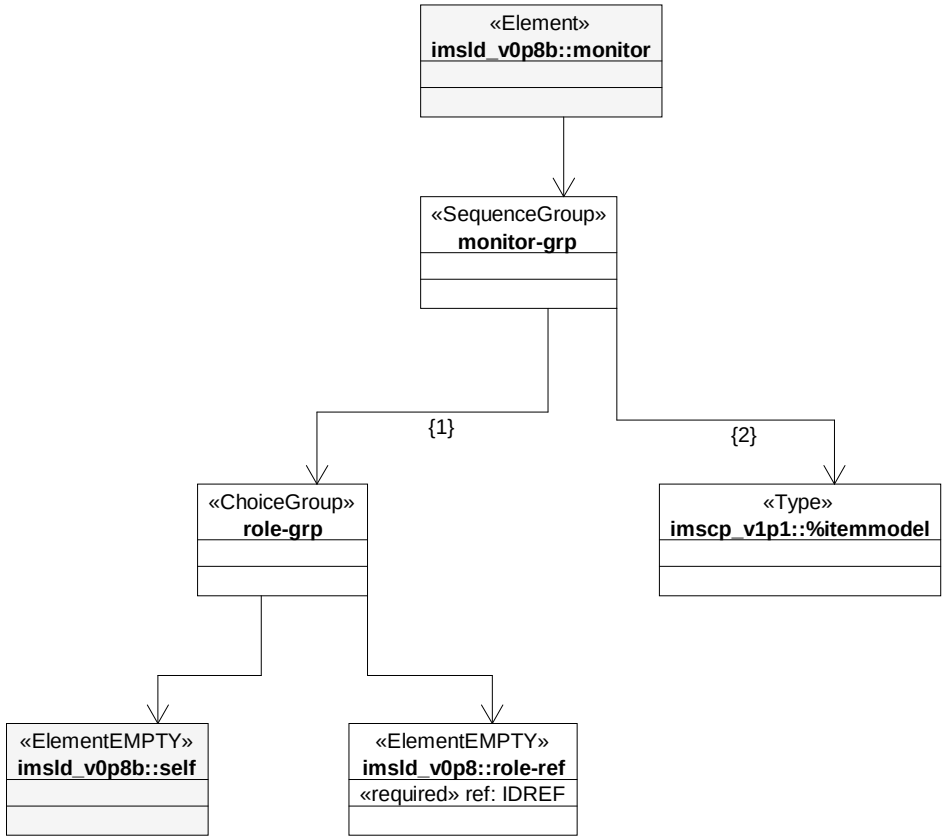
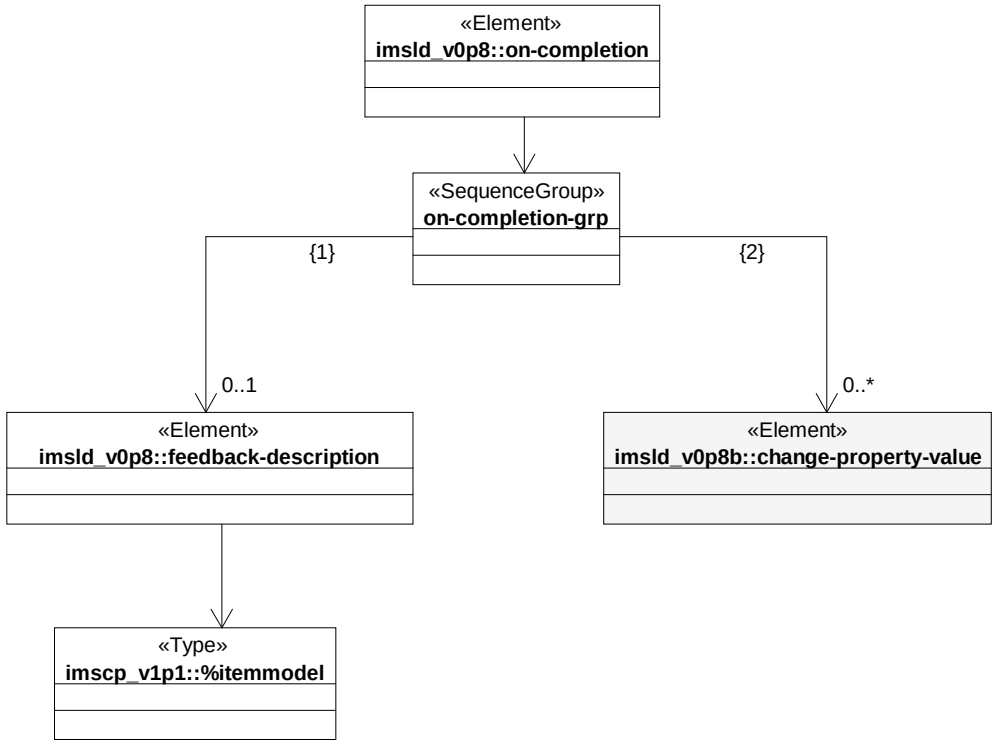
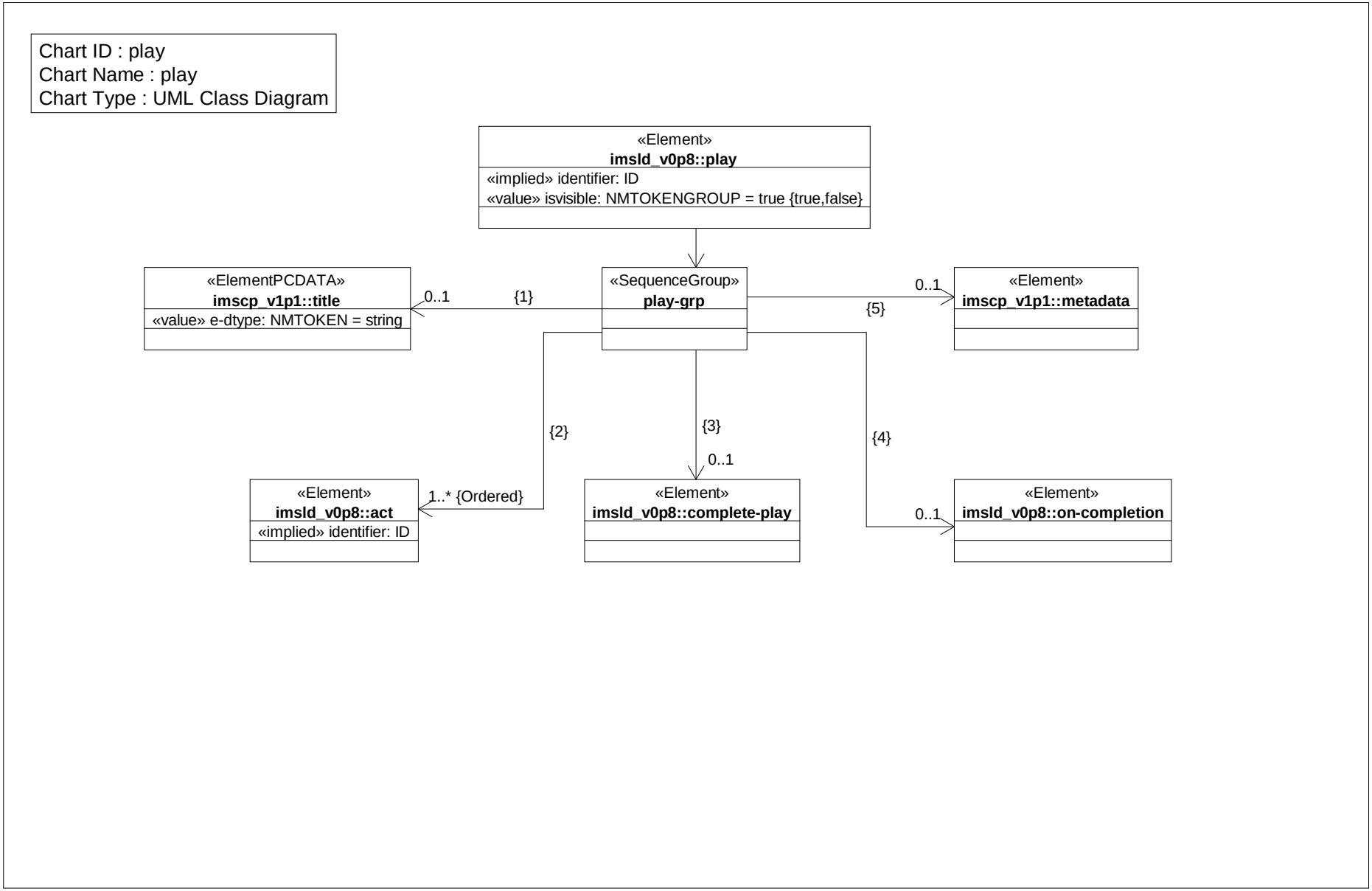


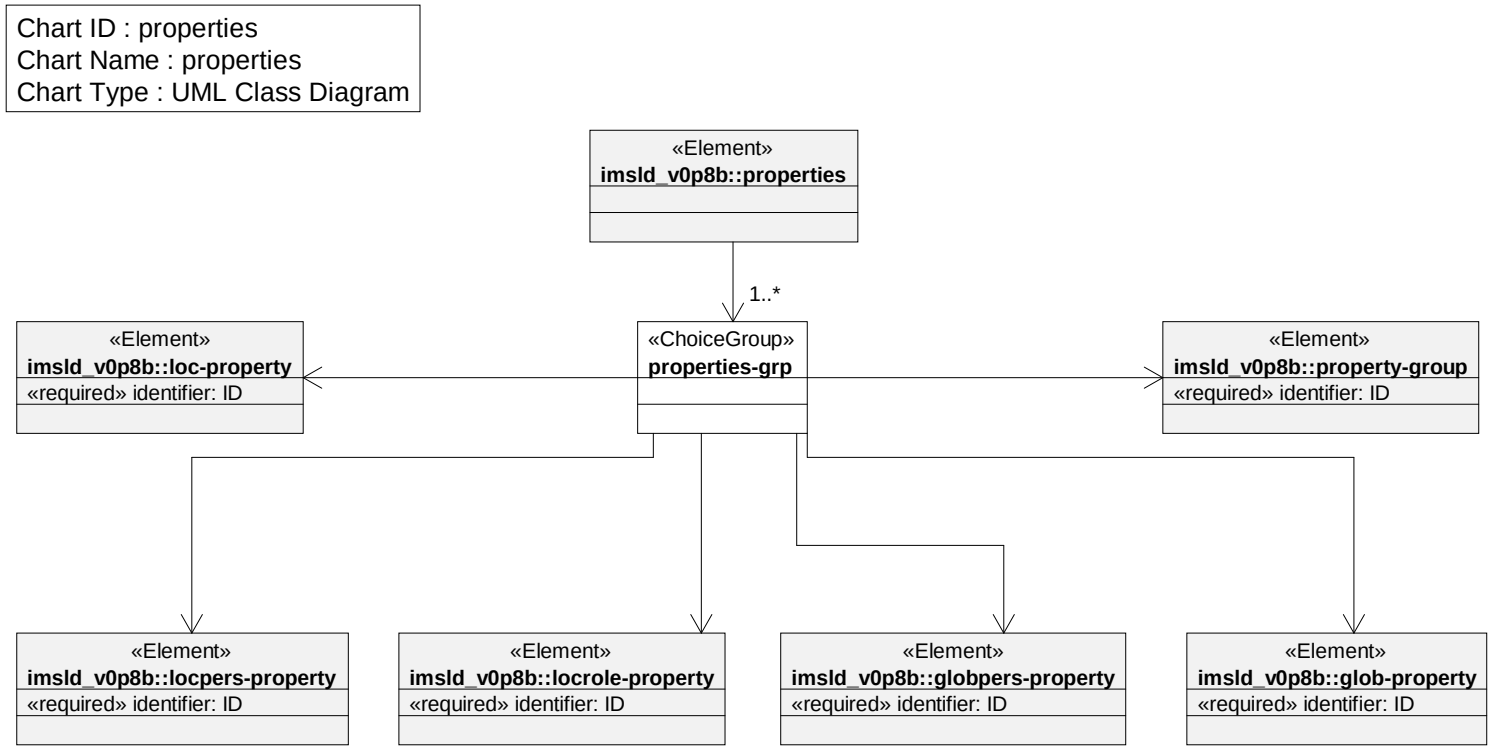
Chart ID : on-completion
Chart Name : on completion
Chart Type : UML Class Diagram



Class Diagram: play



Class Diagram: properties



Class Diagram: property group

Chart ID : property-group
Chart Name : property group
Chart Type : UML Class Diagram

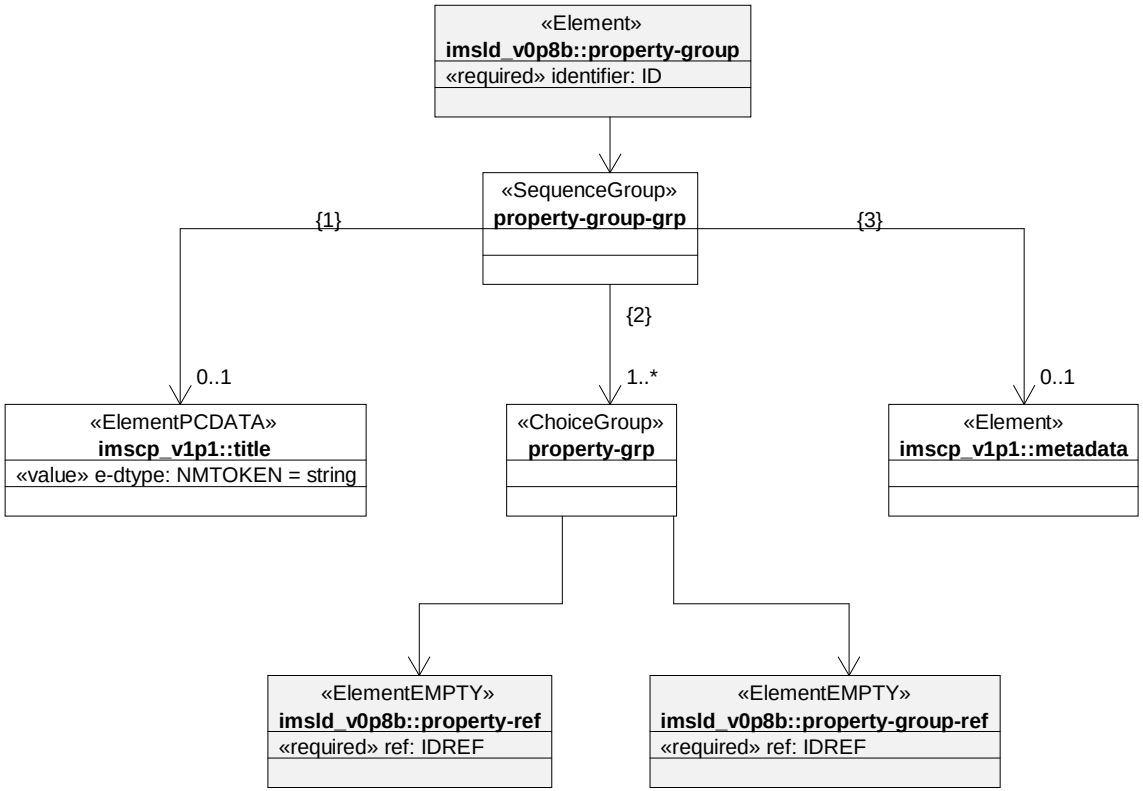
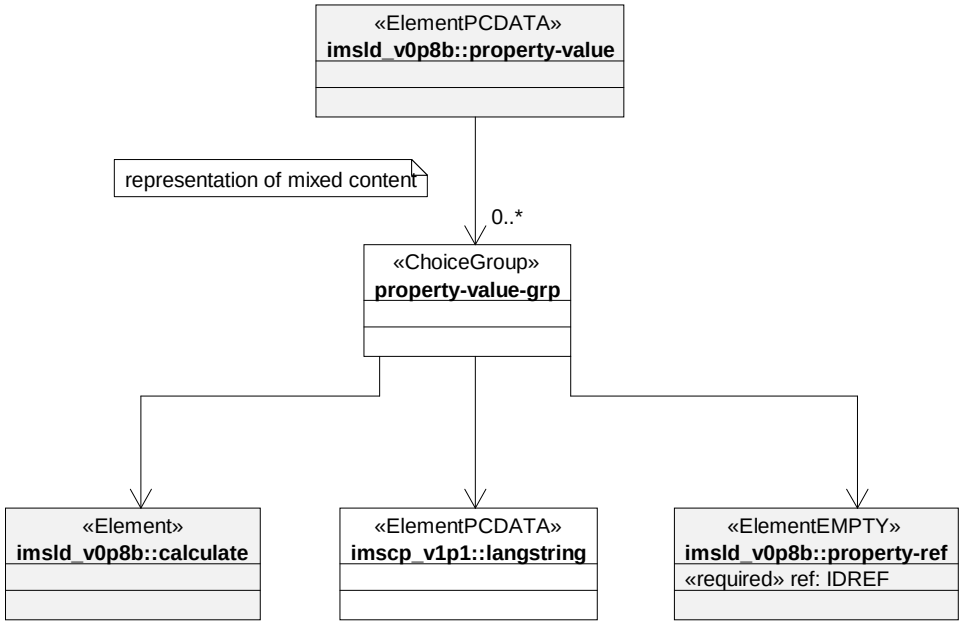
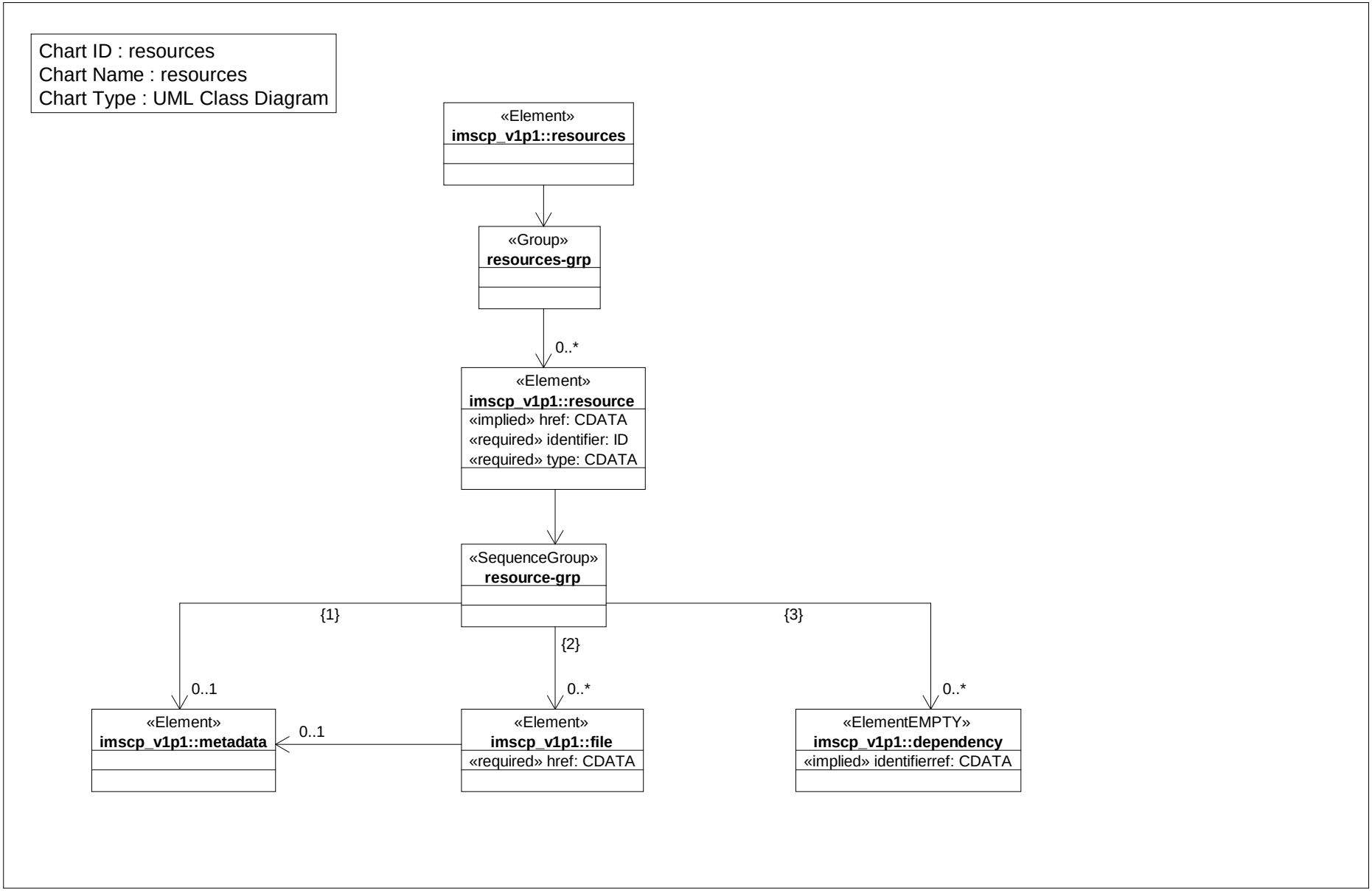


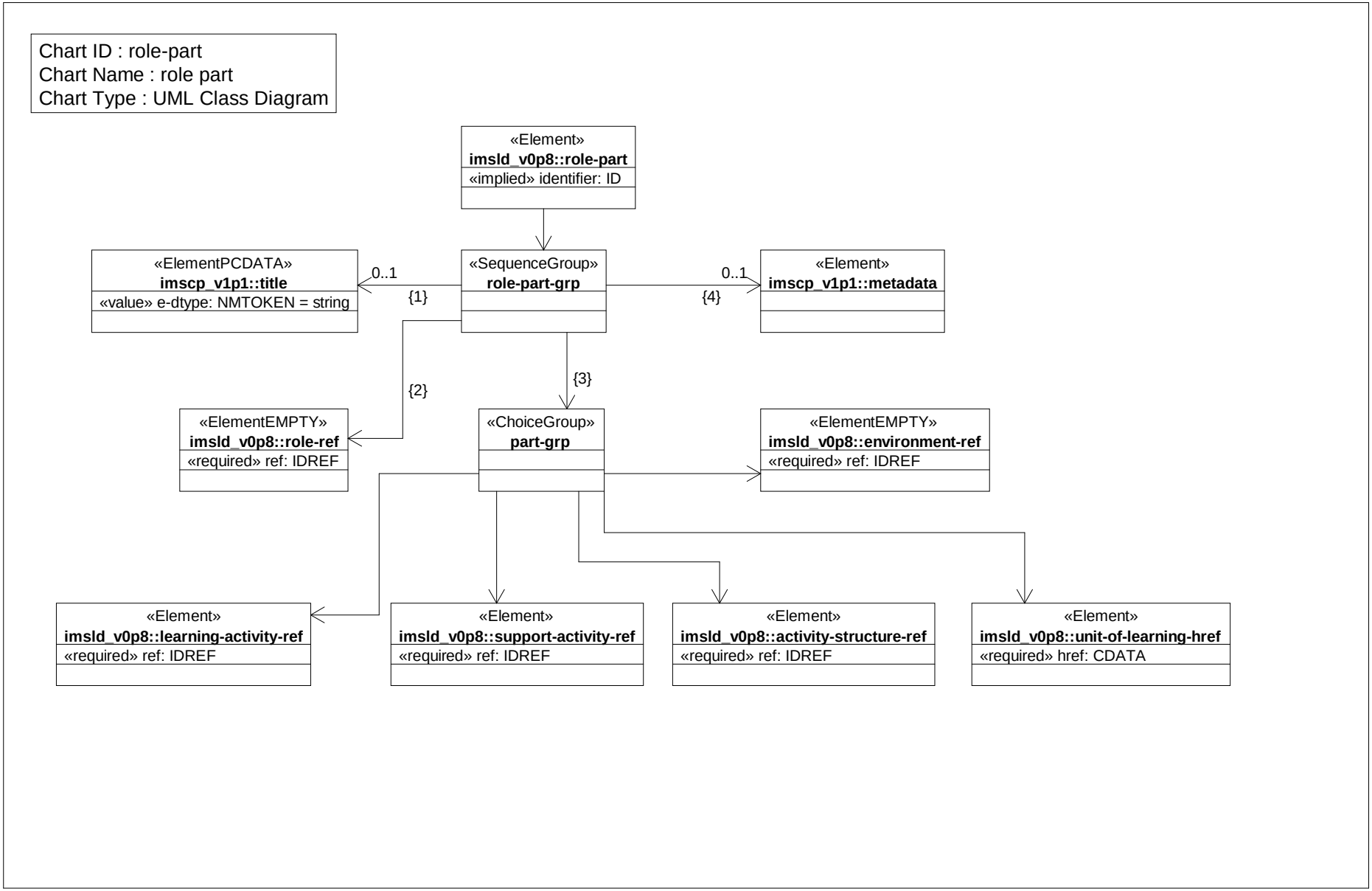
Chart ID : property-value
Chart Name : property value
Chart Type : UML Class Diagram



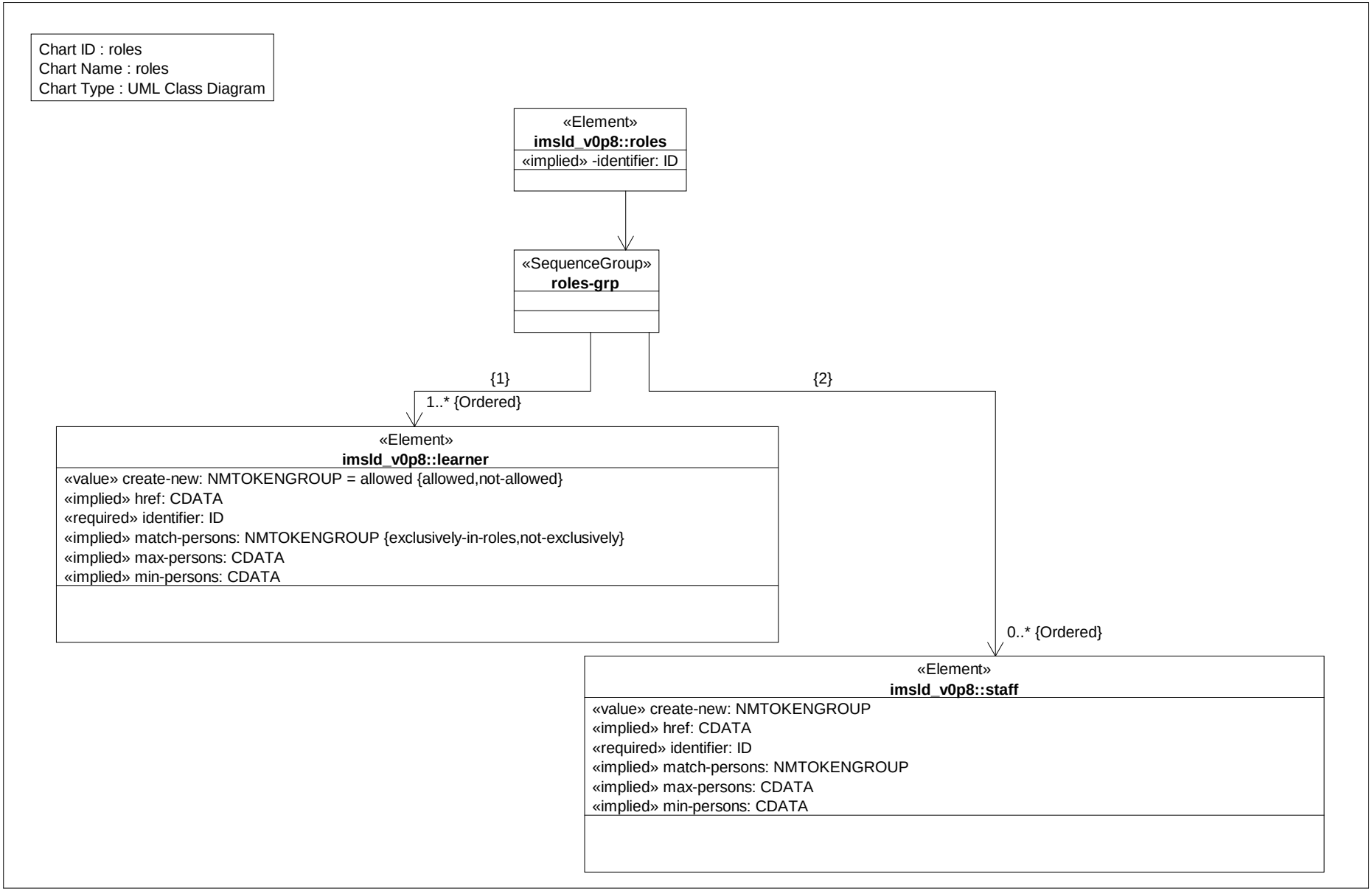
Class Diagram: resources



Class Diagram: role part



Class Diagram: roles



Class Diagram: send mail

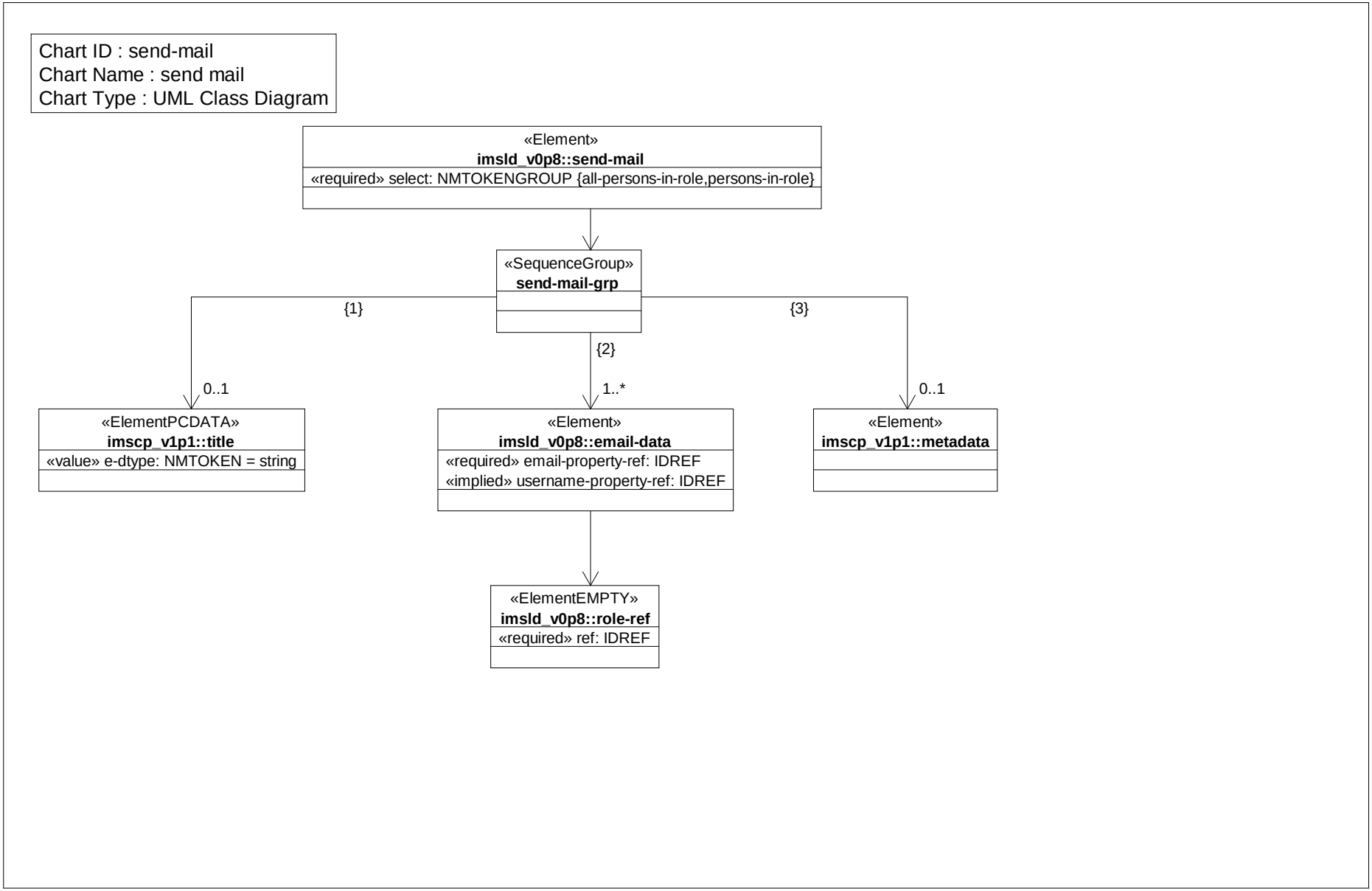
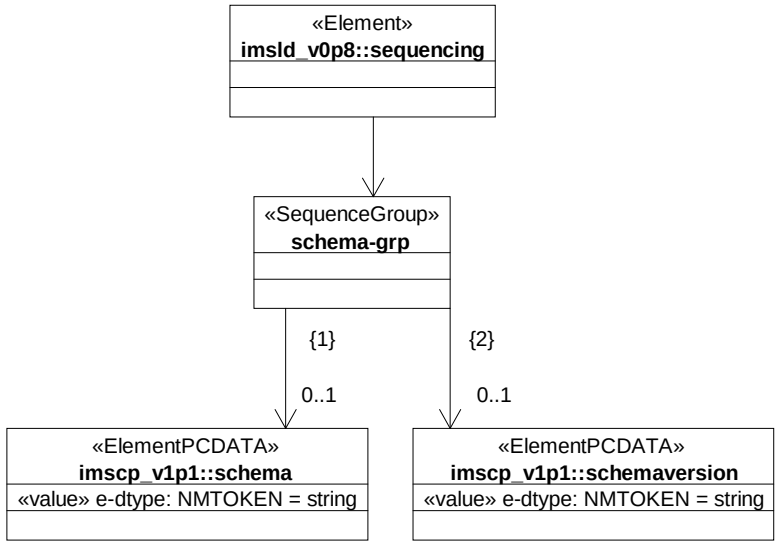
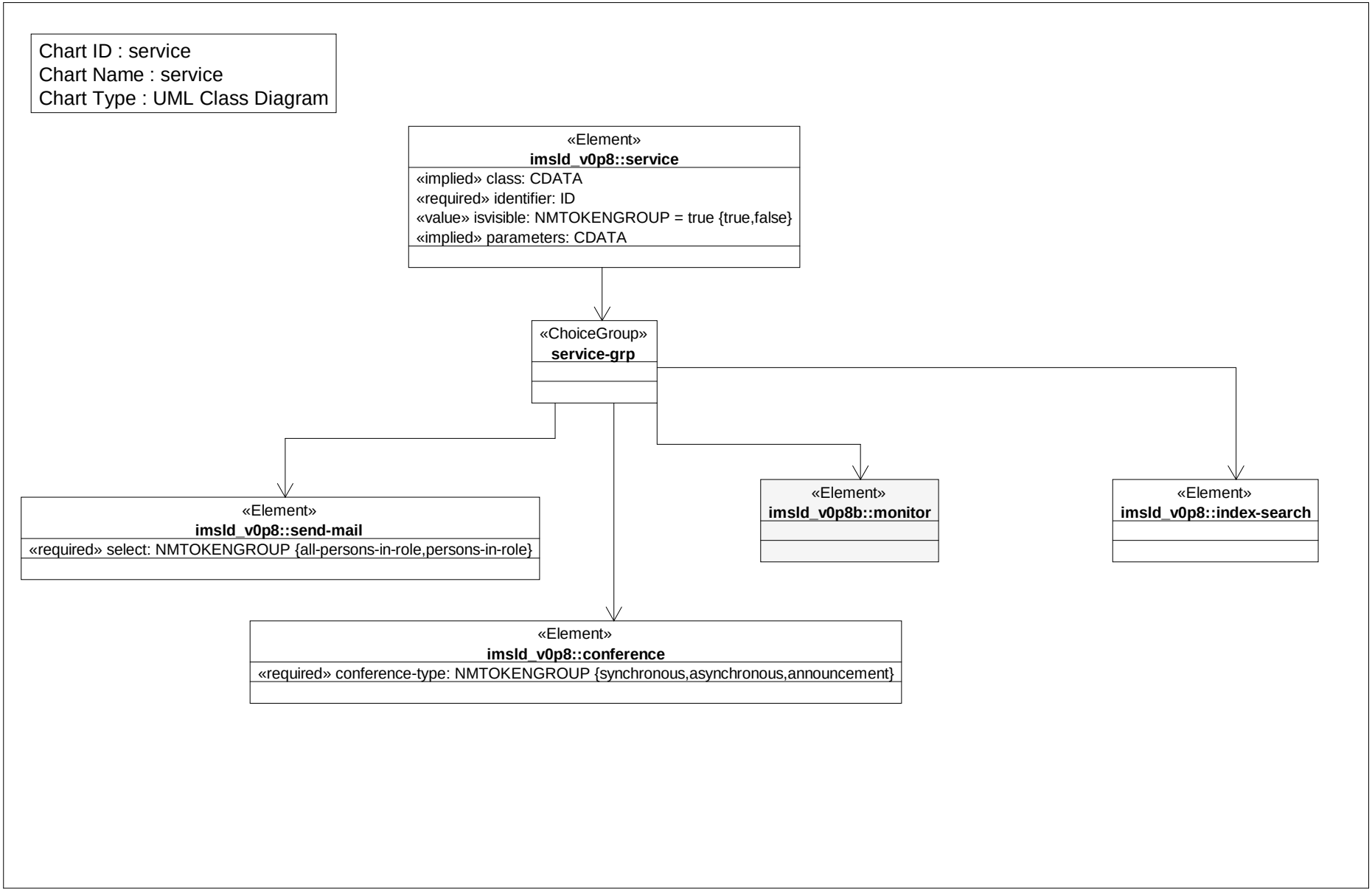


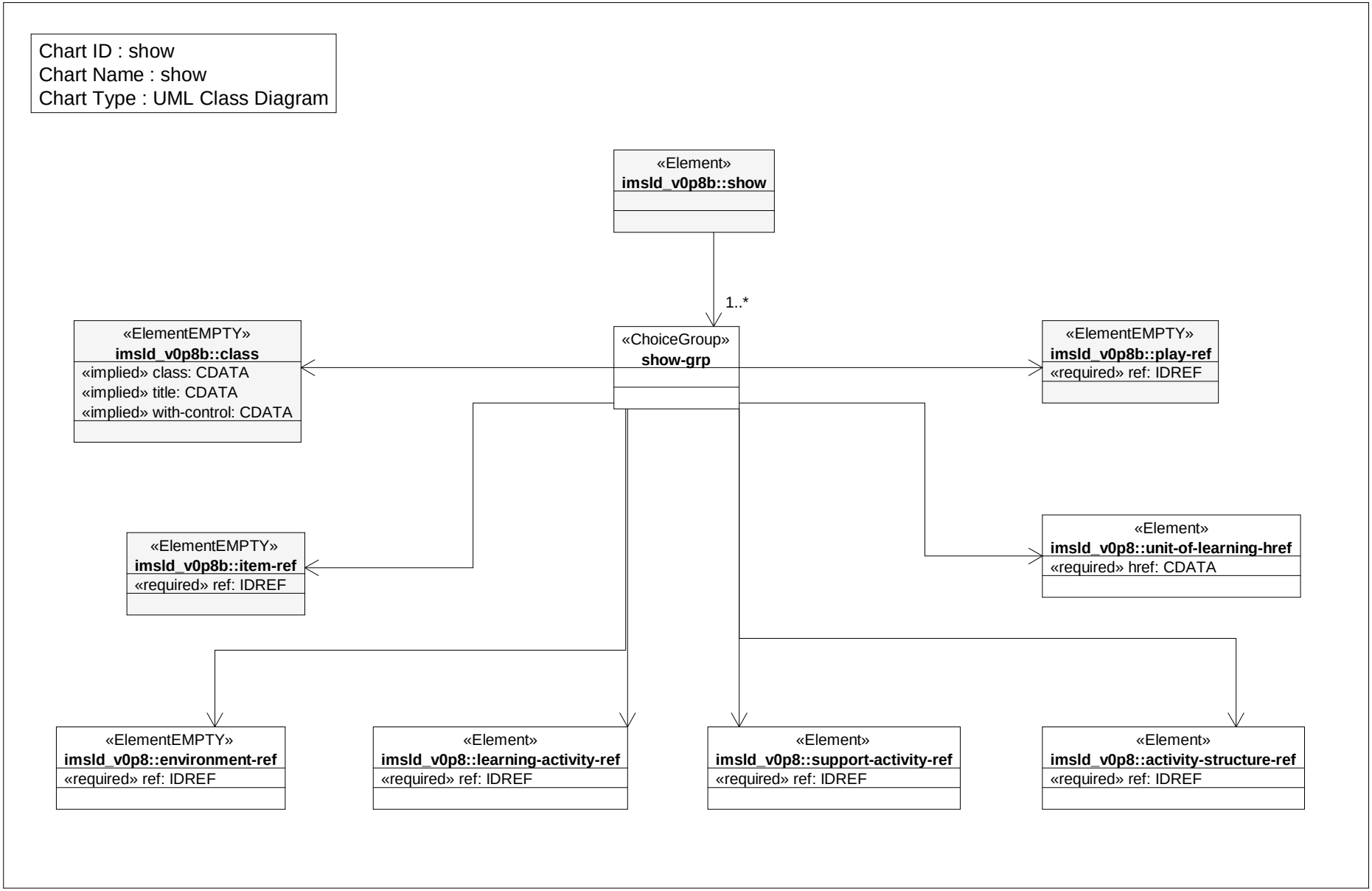
Chart ID : sequencing
Chart Name : sequencing
Chart Type : UML Class Diagram



Class Diagram: service



Class Diagram: show



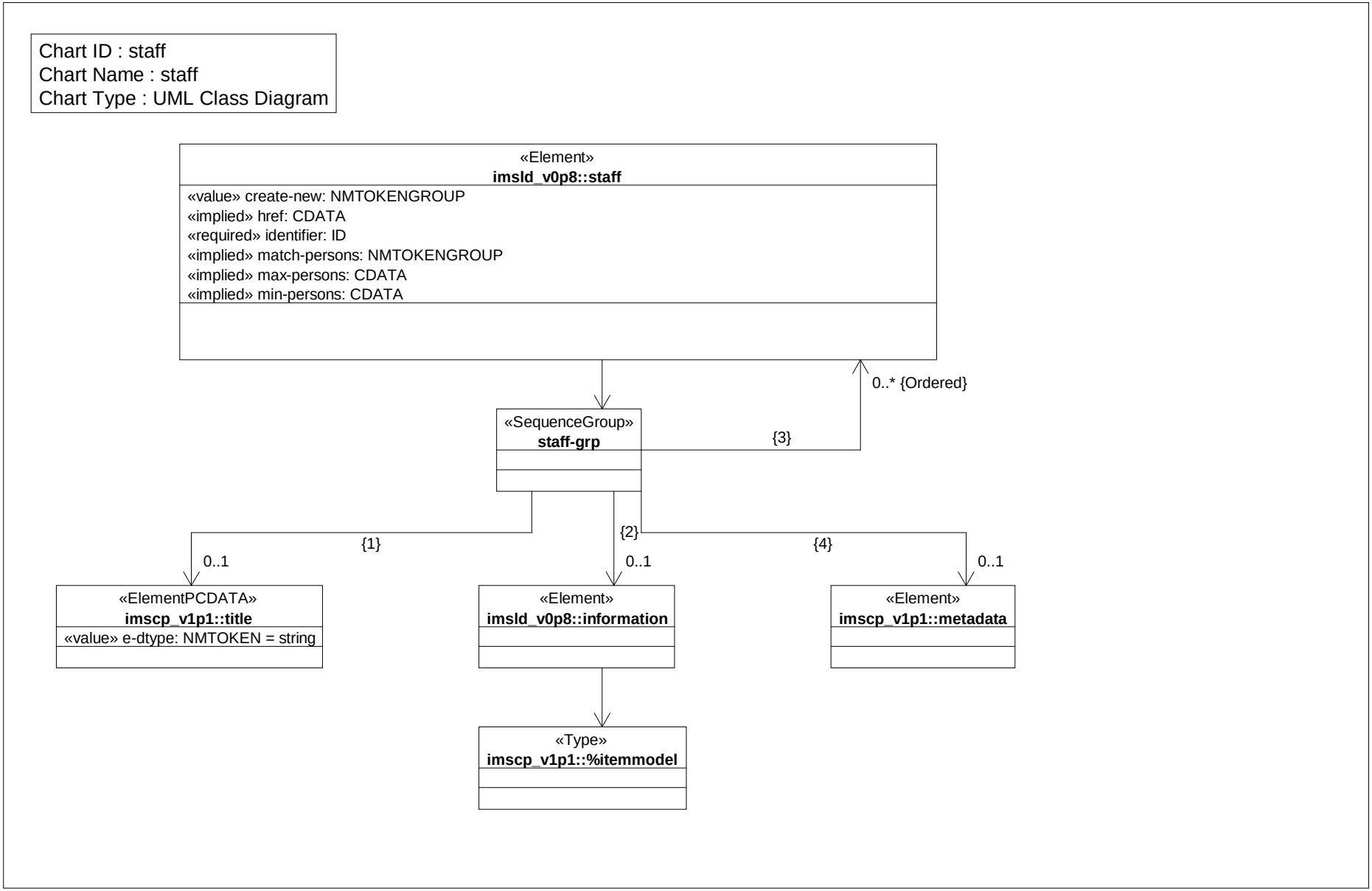


Chart ID : staff
Chart Name : staff
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::staff
«value» create-new: NMTOKENGROUP
«implied» href: CDATA
«required» identifier: ID
«implied» match-persons: NMTOKENGROUP
«implied» max-persons: CDATA
«implied» min-persons: CDATA
0..* {Ordered}
«SequenceGroup»
staff-grp
{3}
{1}
{2}
{4}
0..1
0..1
0..1
«ElementPCDATA»
imscp_v1p1::title
«value» e-dtype: NMTOKEN = string
«Element»
imsld_v0p8::information
«Element»
imscp_v1p1::metadata
«Type»
imscp_v1p1::%itemmodel

Class Diagram: support activity

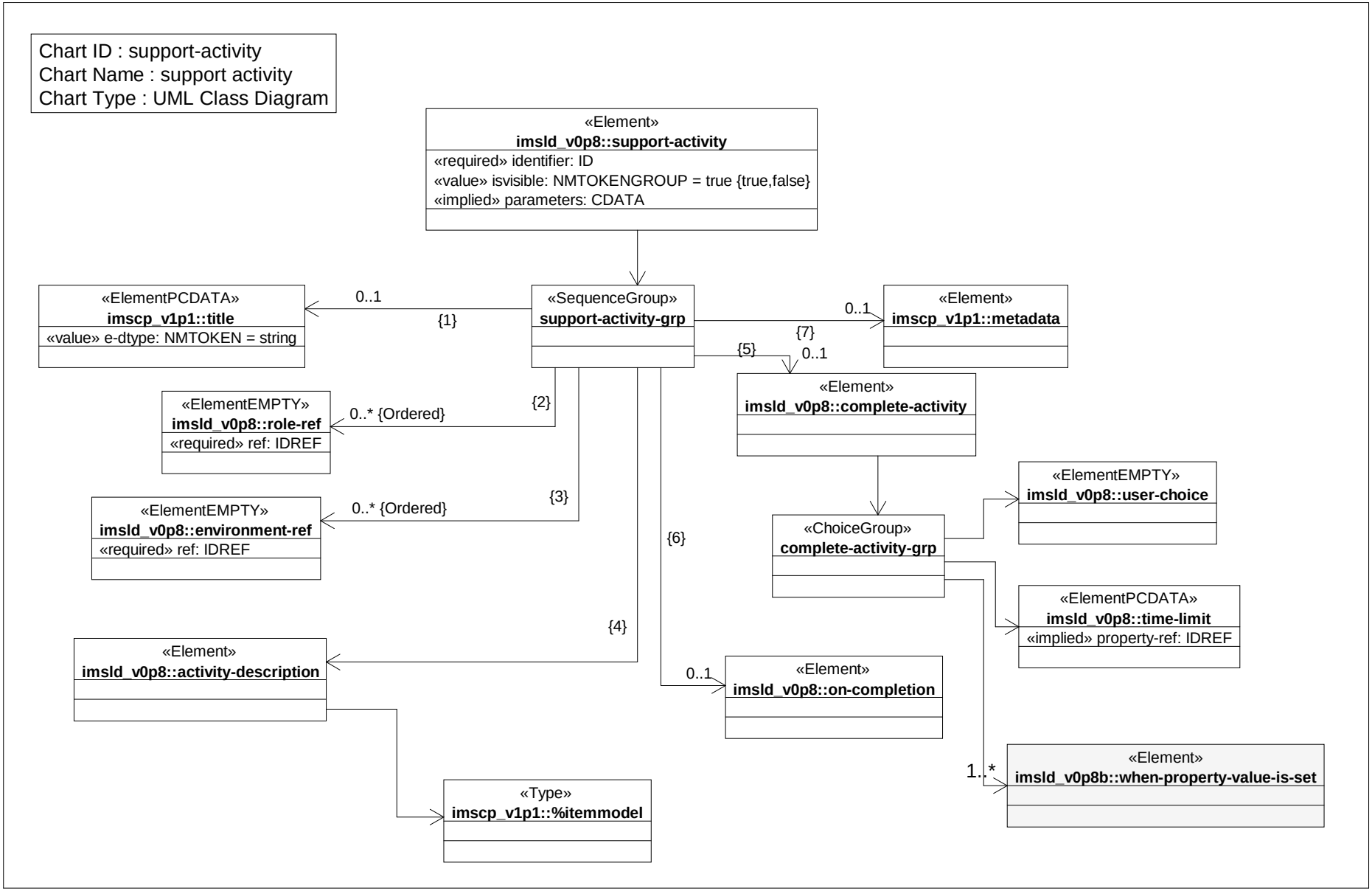
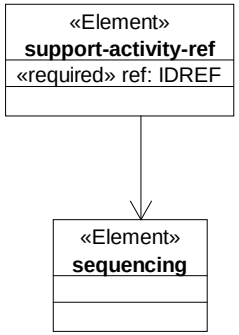


Chart ID : support-activity-ref
Chart Name : support activity ref
Chart Type : UML Class Diagram



Class Diagram: unit of learning

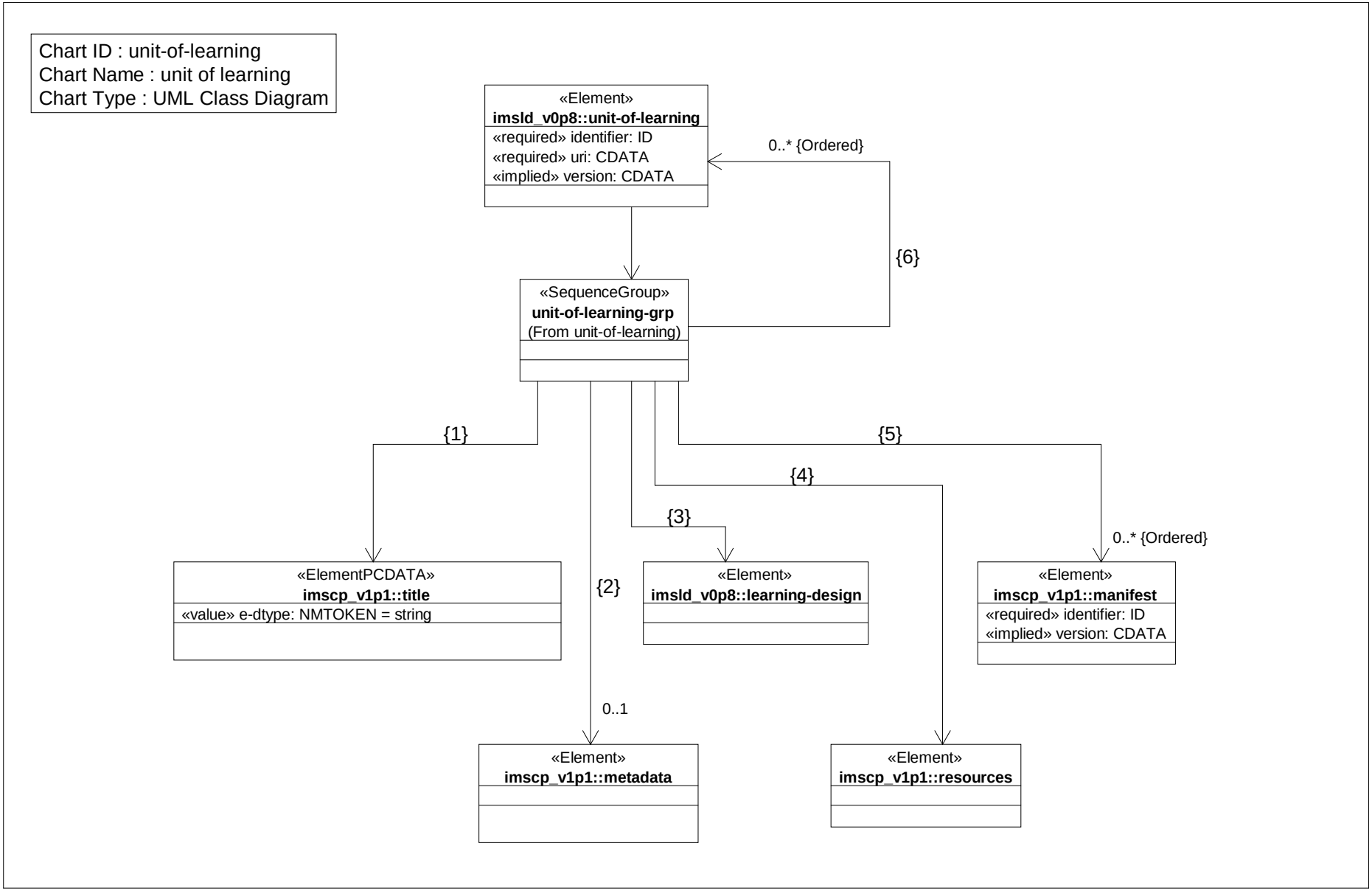


Chart ID : unit-of-learning-href
Chart Name : unit of learning href
Chart Type : UML Class Diagram

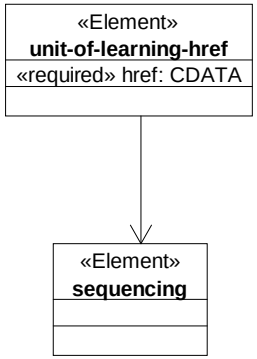


Chart ID : users-in-role
Chart Name : users in role
Chart Type : UML Class Diagram

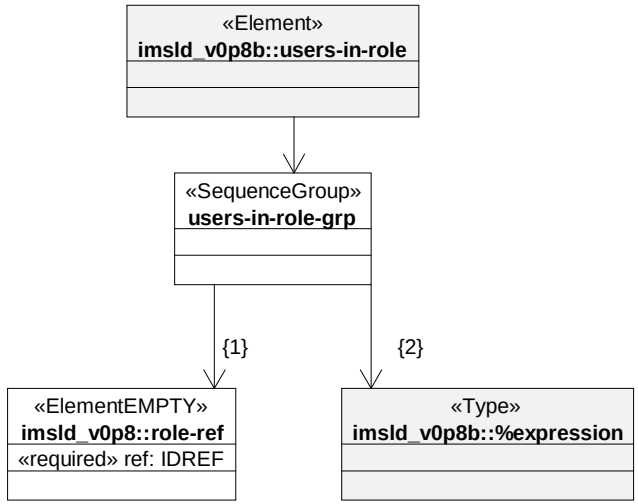
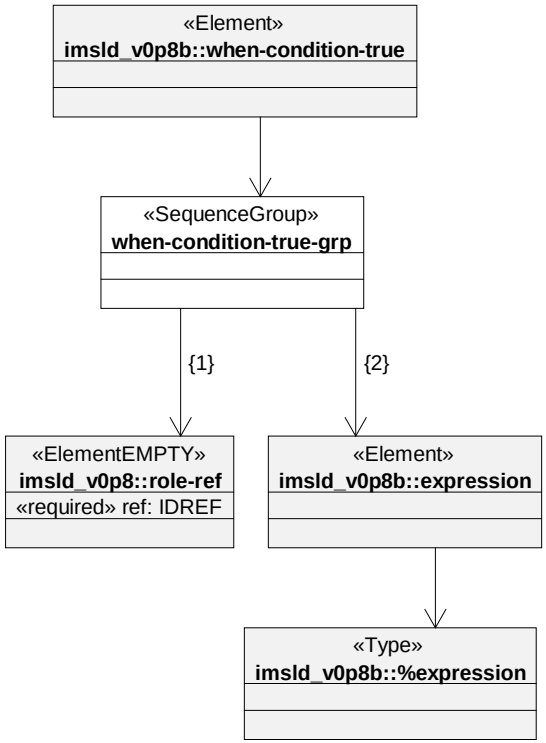
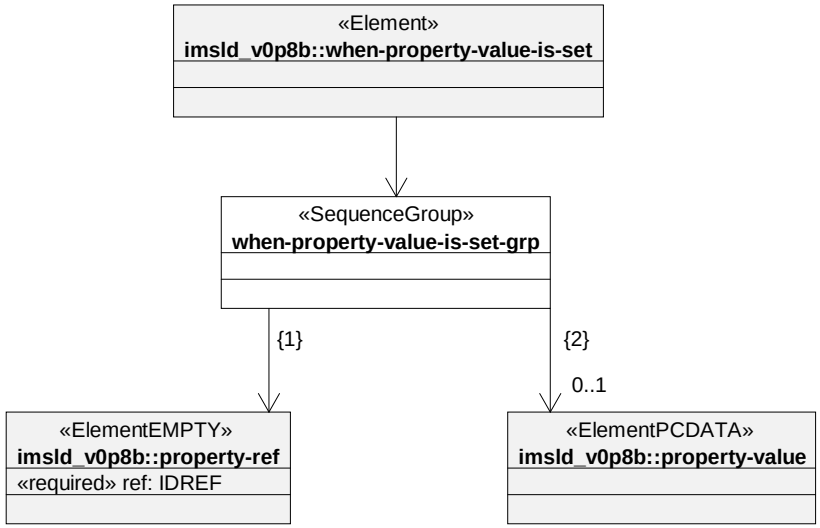


Chart ID : when-condition-true
Chart Name : when condition true
Chart Type : UML Class Diagram



Class Diagram: when propety value is set

Chart ID : when-property-value-is-set
Chart Name : when propety value is set
Chart Type : UML Class Diagram



Model level C

Chart ID : IMSLD
Chart Name : IMS learning design
Chart Type : UML Package Diagram

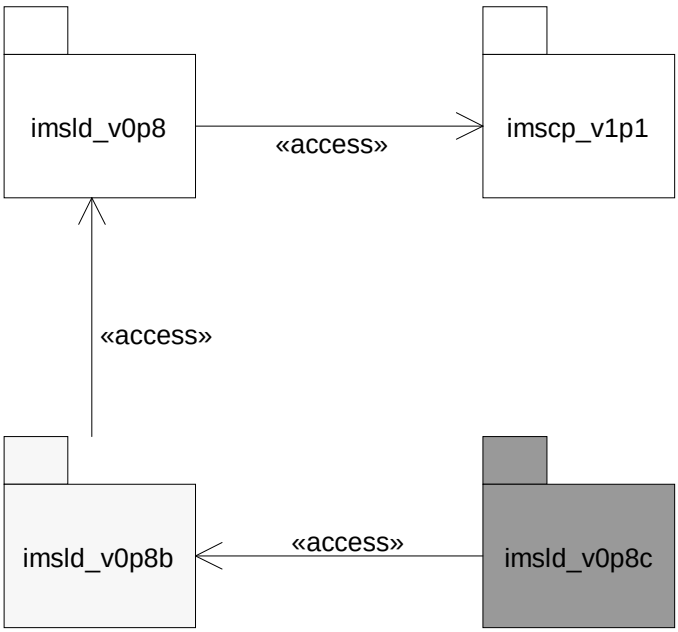


Chart ID : IMSLD-global
Chart Name : IMSLD global elements
Chart Type : UML Package Diagram

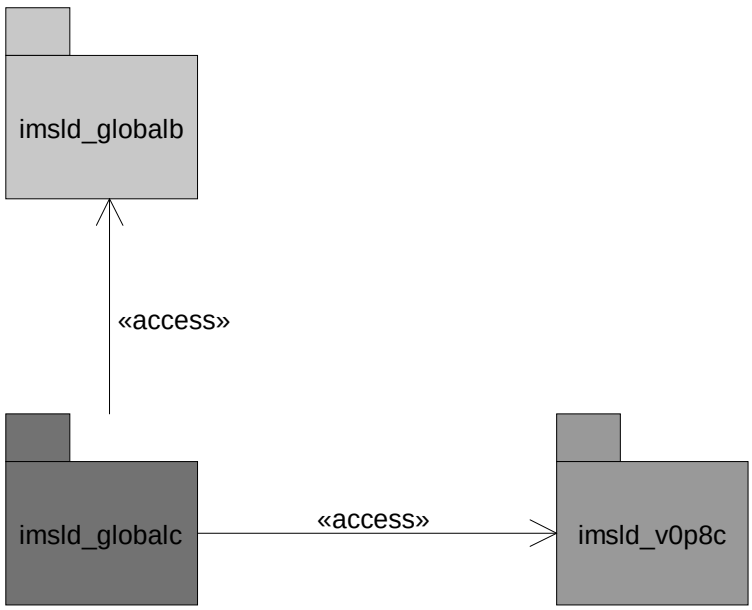
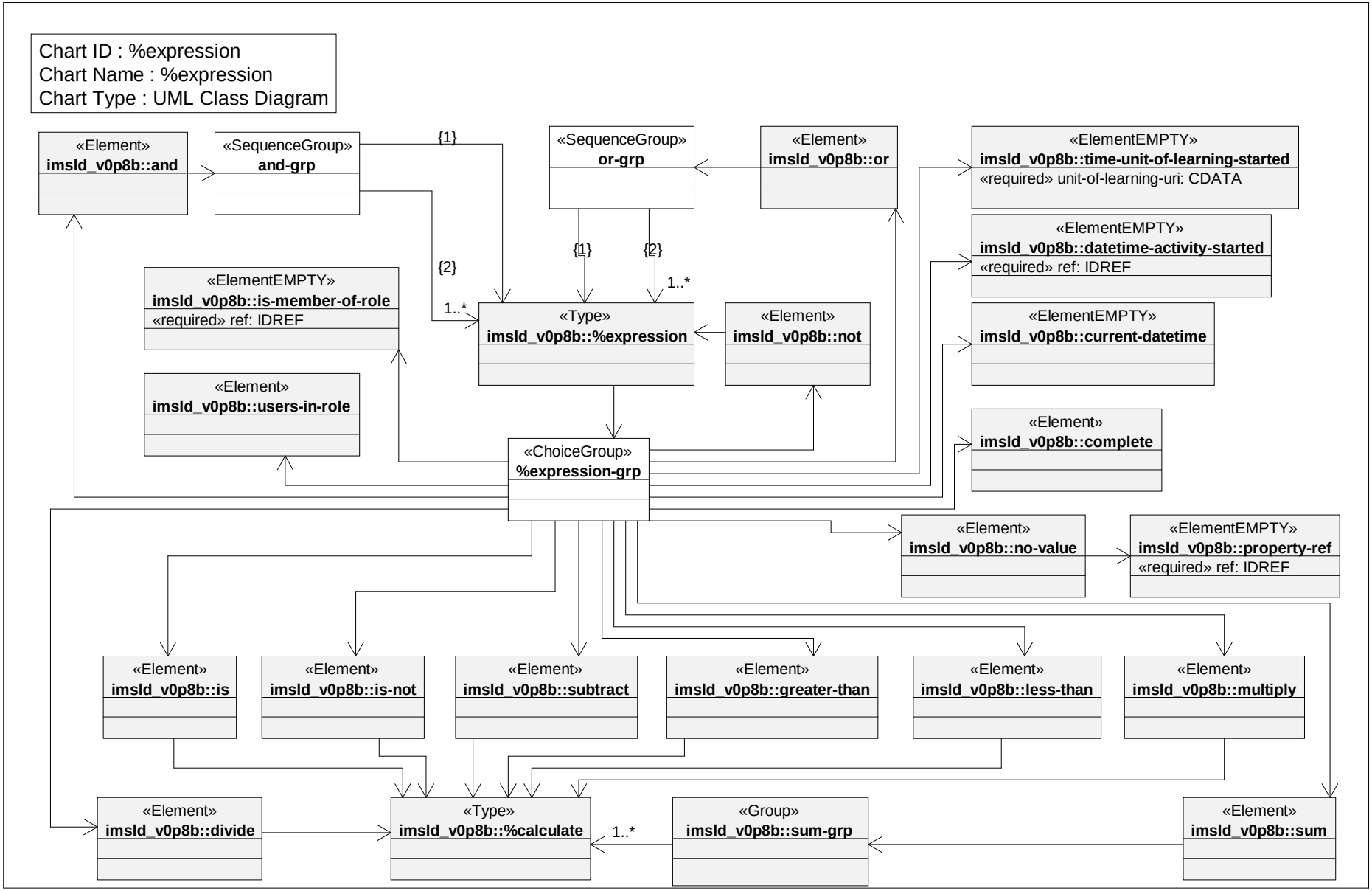
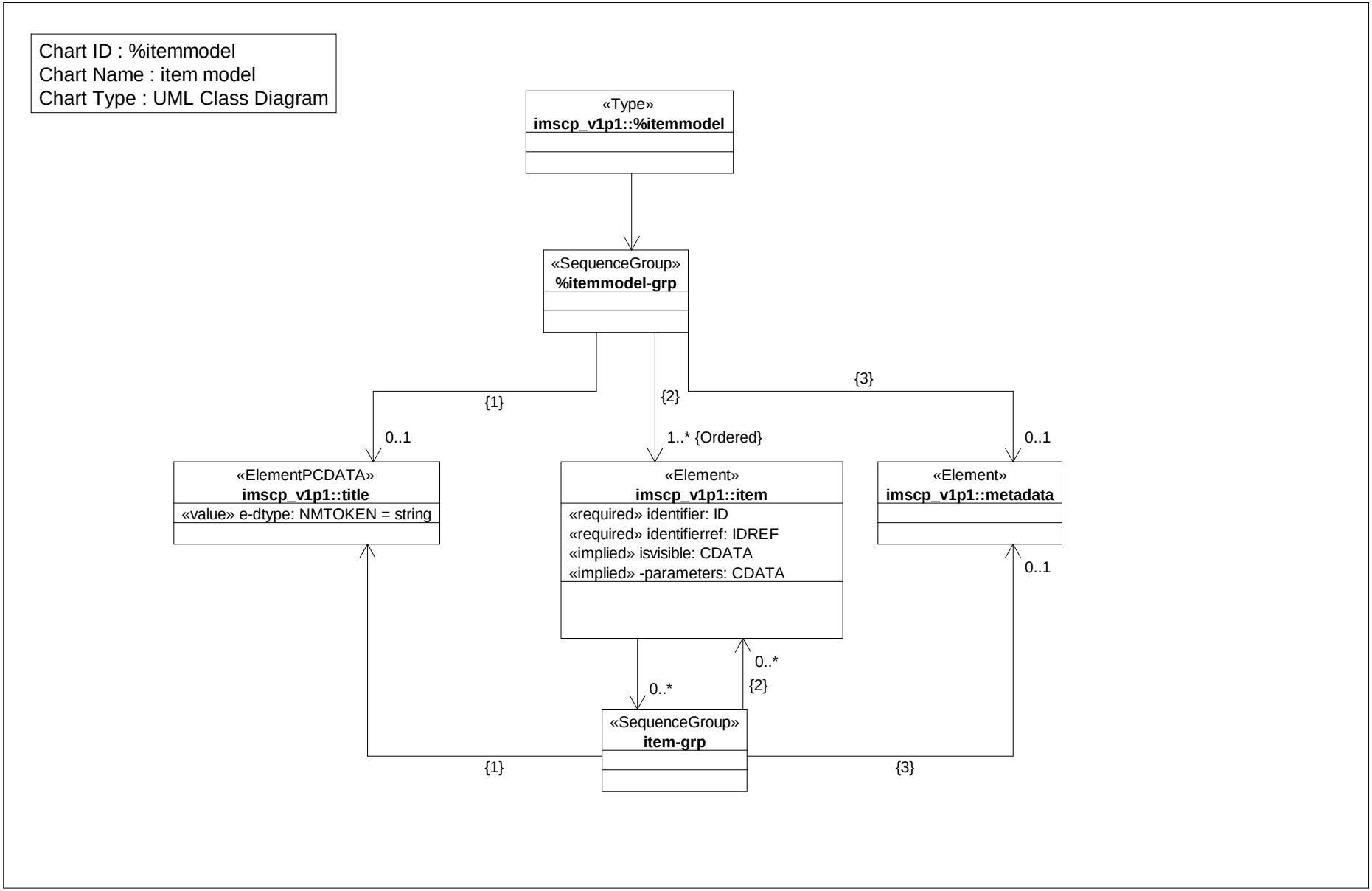


Chart ID : %calculate
Chart Name : %calculate
Chart Type : UML Class Diagram

Class Diagram: %expression



Class Diagram: item model



Class Diagram: %thenmodel

Chart ID : %thenmodel
Chart Name : %thenmodel
Chart Type : UML Class Diagram

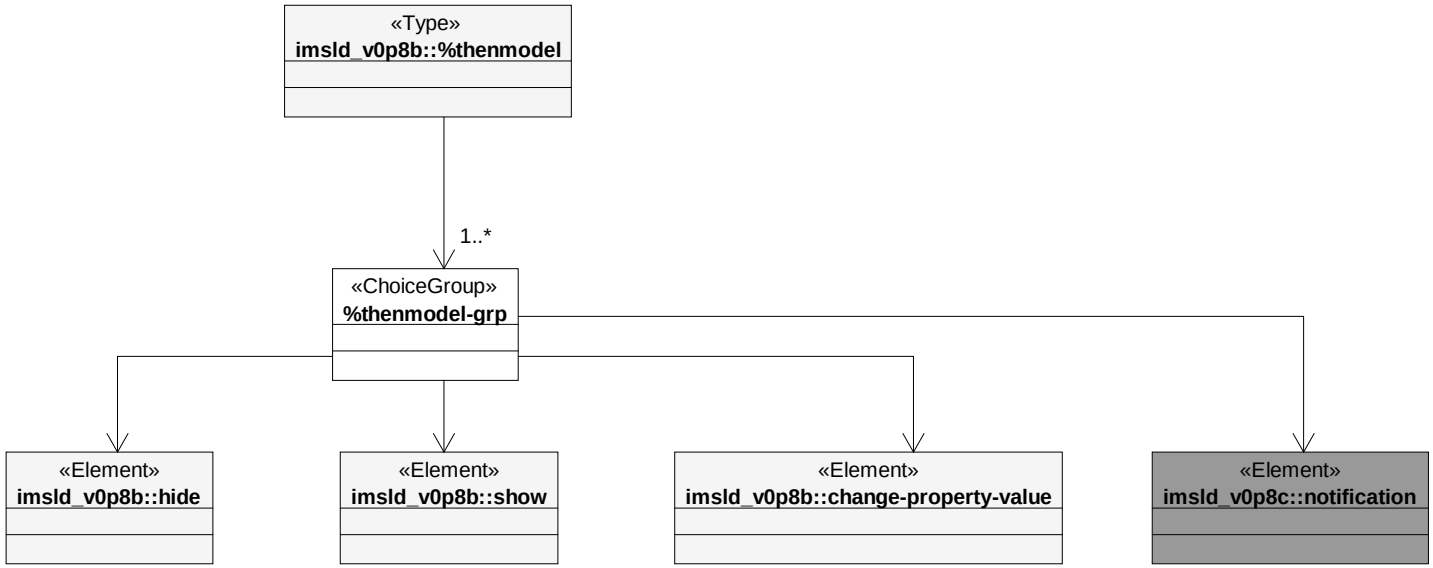
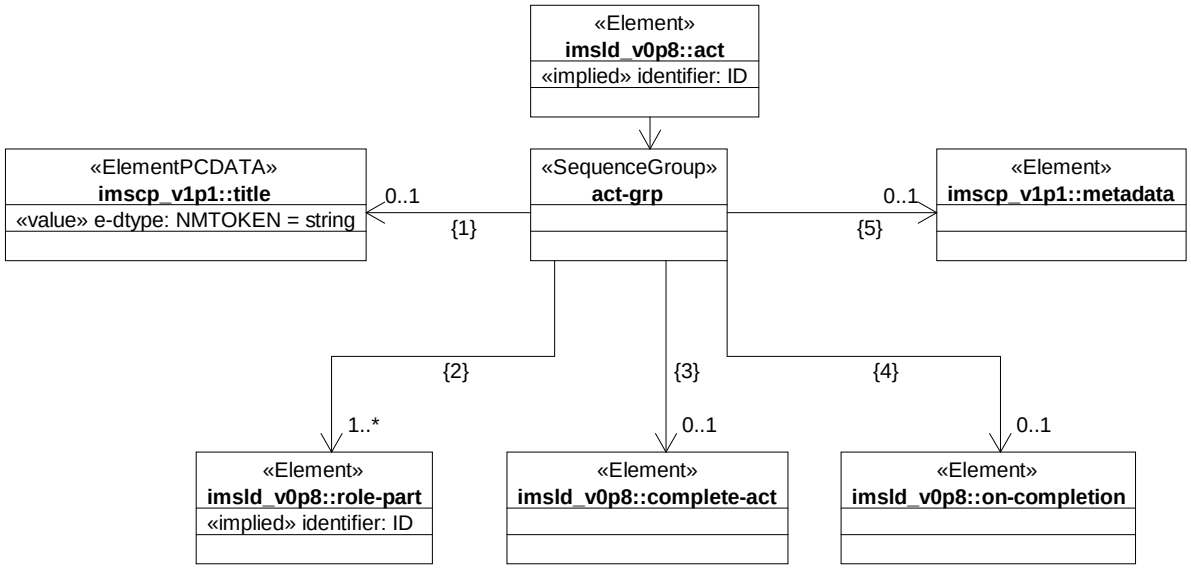


Chart ID : act
Chart Name : act
Chart Type : UML Class Diagram



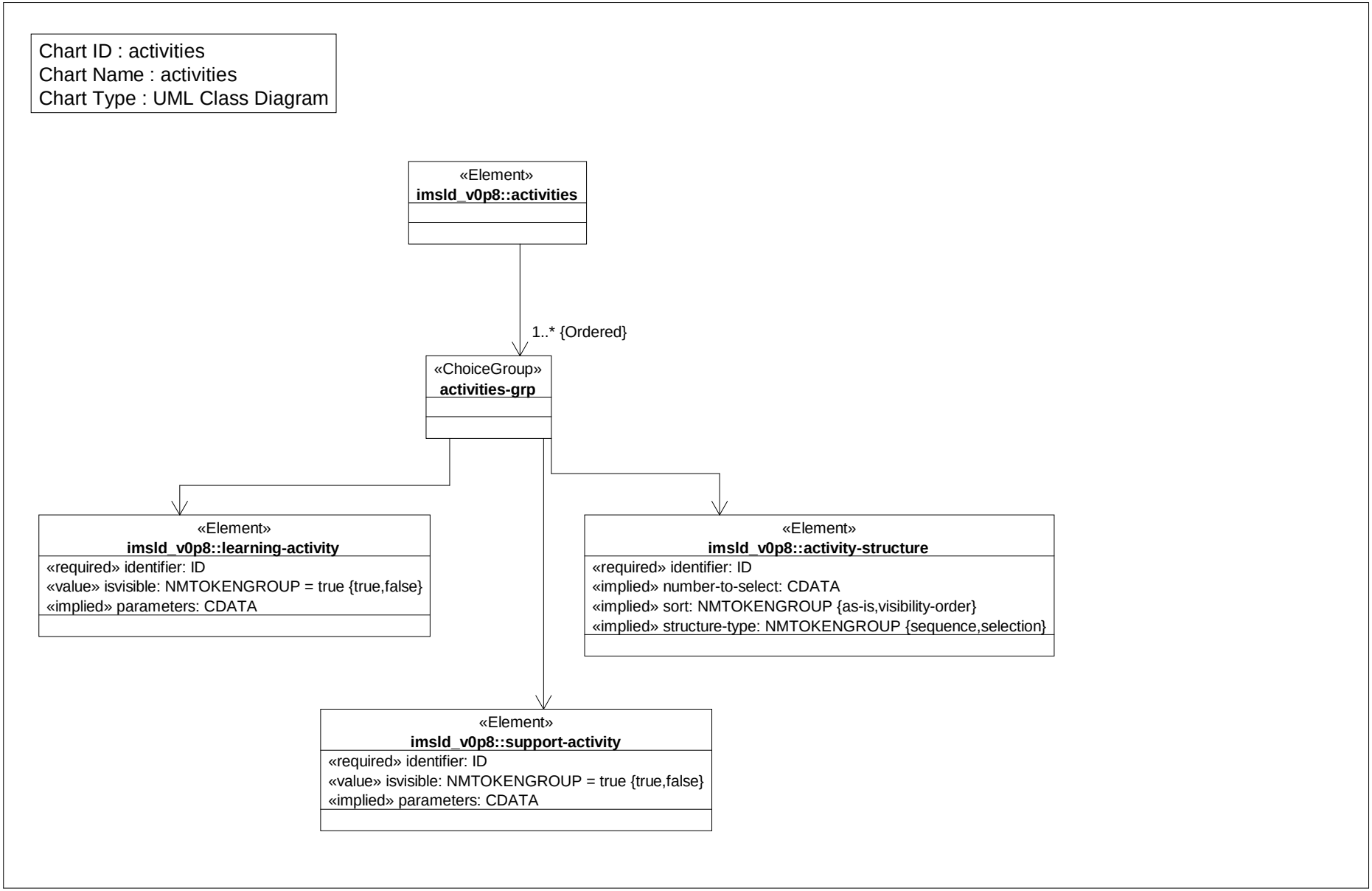


Chart ID : activities
Chart Name : activities
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::activities
1..* {Ordered}
«ChoiceGroup»
activities-grp
«Element»
imsld_v0p8::learning-activity
«required» identifier: ID
«value» isvisible: NMTOKENGROUP = true {true,false}
«implied» parameters: CDATA
«Element»
imsld_v0p8::activity-structure
«required» identifier: ID
«implied» number-to-select: CDATA
«implied» sort: NMTOKENGROUP {as-is,visibility-order}
«implied» structure-type: NMTOKENGROUP {sequence,selection}
«Element»
imsld_v0p8::support-activity
«required» identifier: ID
«value» isvisible: NMTOKENGROUP = true {true,false}
«implied» parameters: CDATA

Class Diagram: activty structure

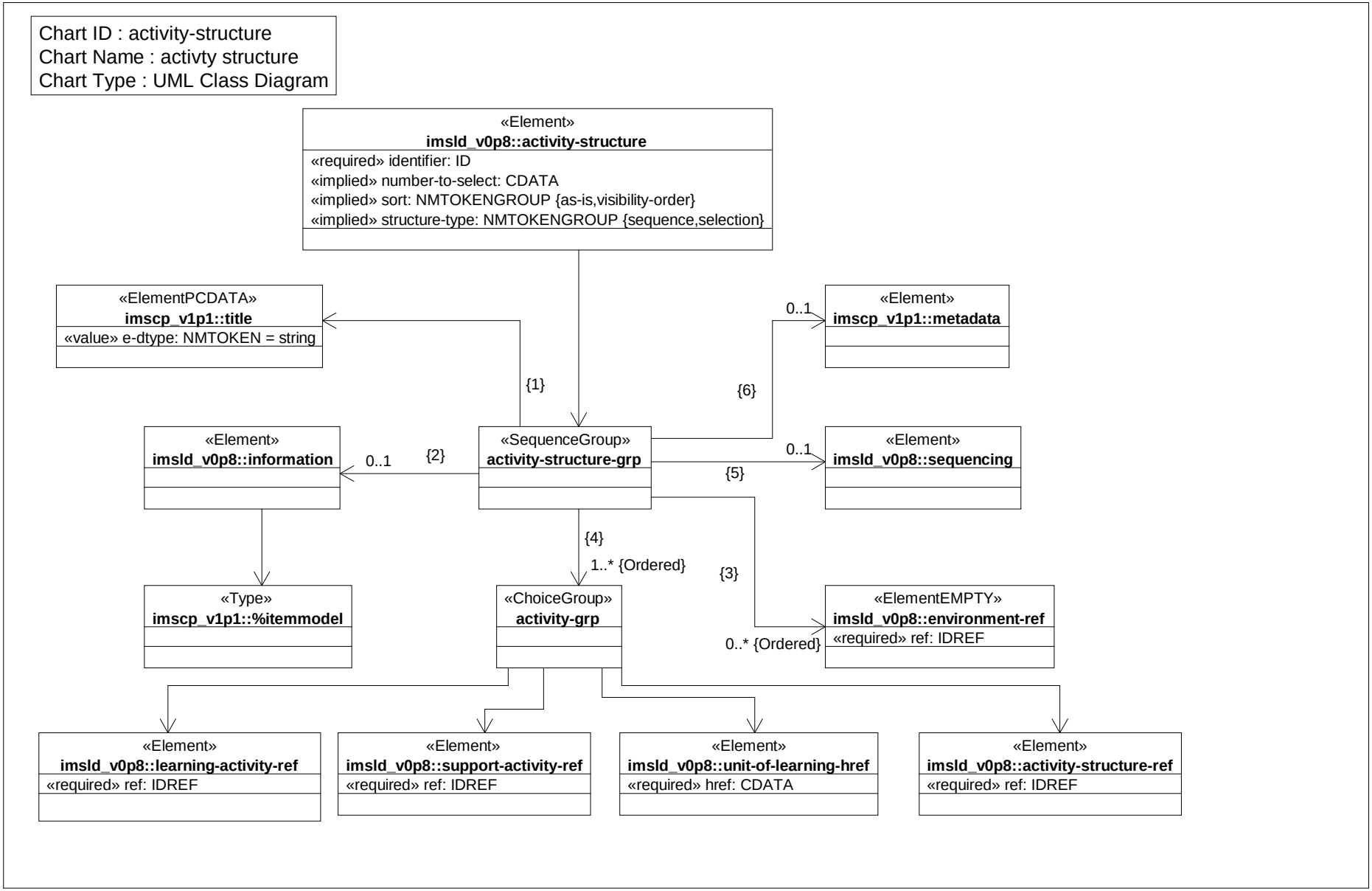


Chart ID : activity-structure-ref
Chart Name : activity structure ref
Chart Type : UML Class Diagram

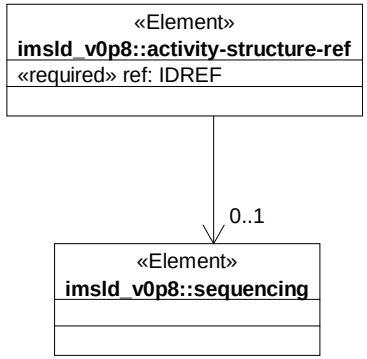
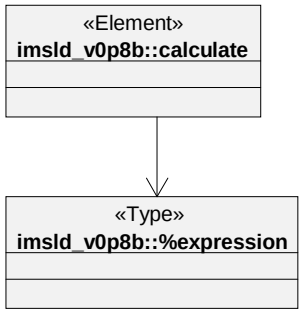
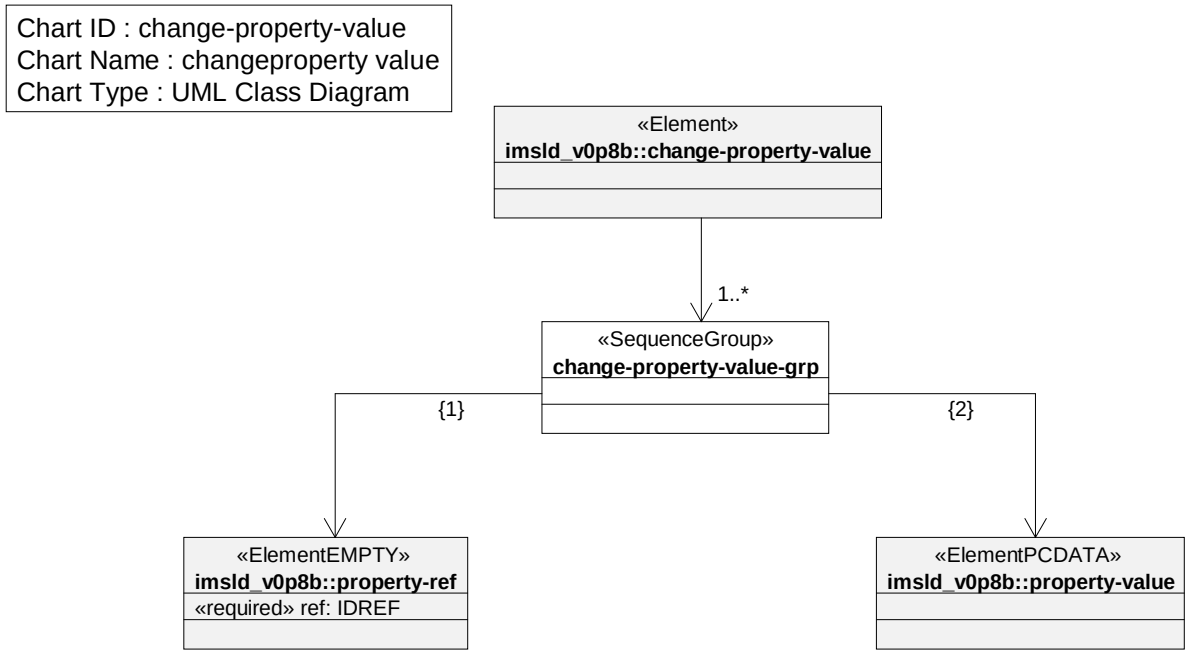


Chart ID : calculate
Chart Name : calculate
Chart Type : UML Class Diagram



Class Diagram: changeproperty value



Class Diagram: complete

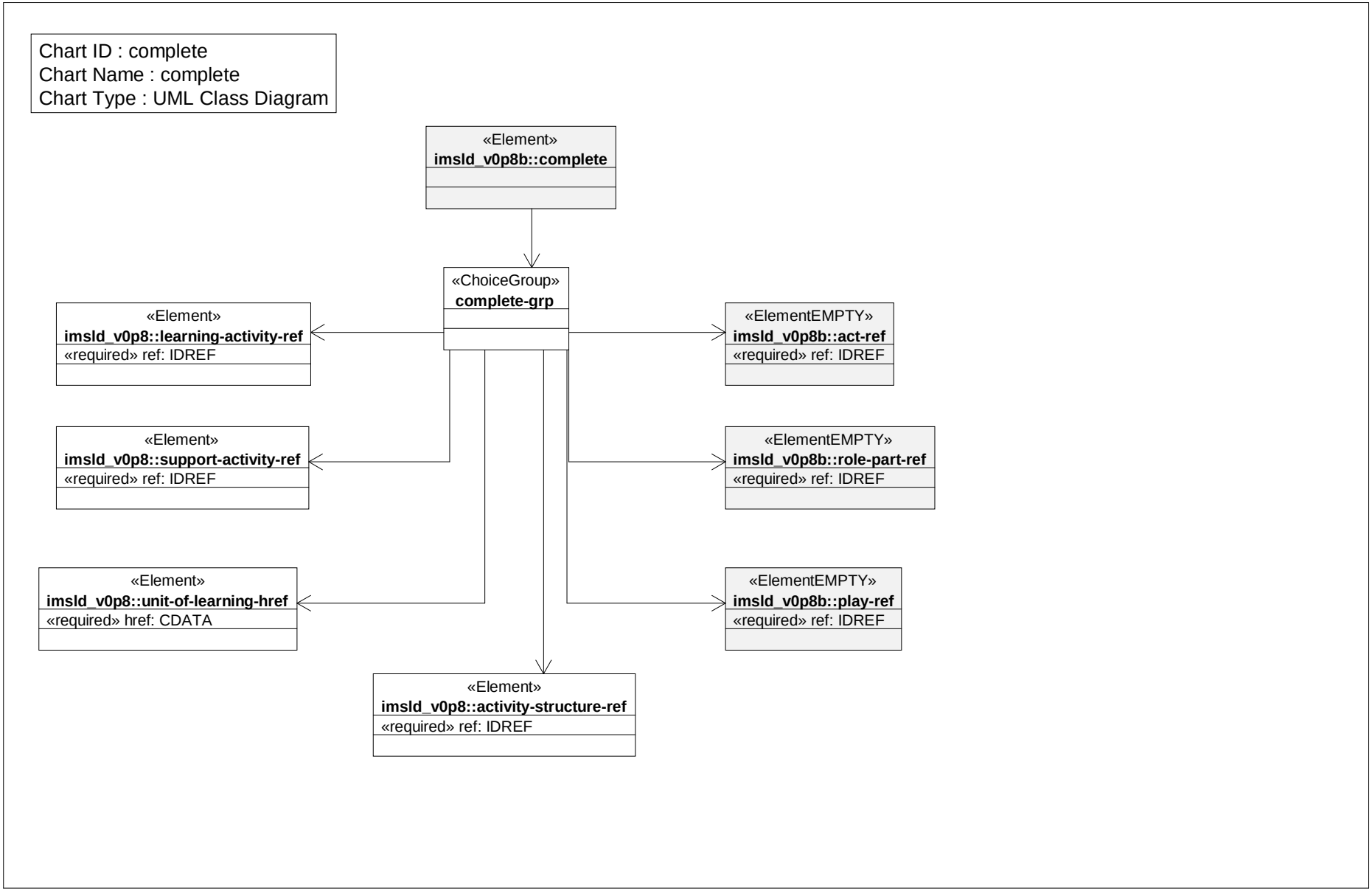


Chart ID : complete-act
Chart Name : complete act
Chart Type : UML Class Diagram

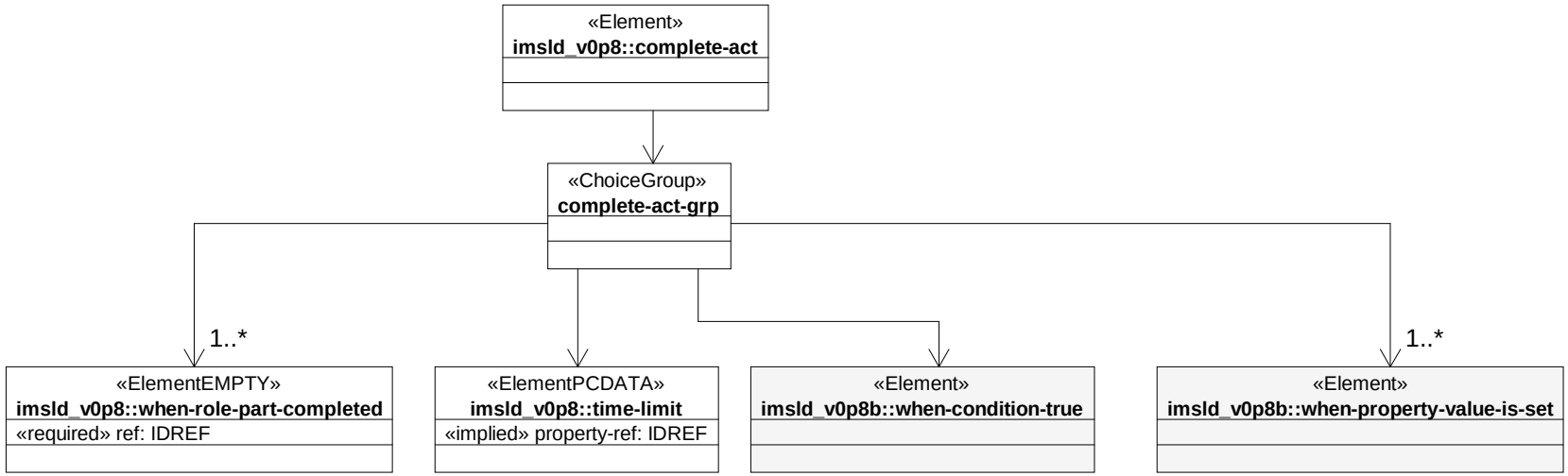


Chart ID : complete-play
Chart Name : complete play
Chart Type : UML Class Diagram

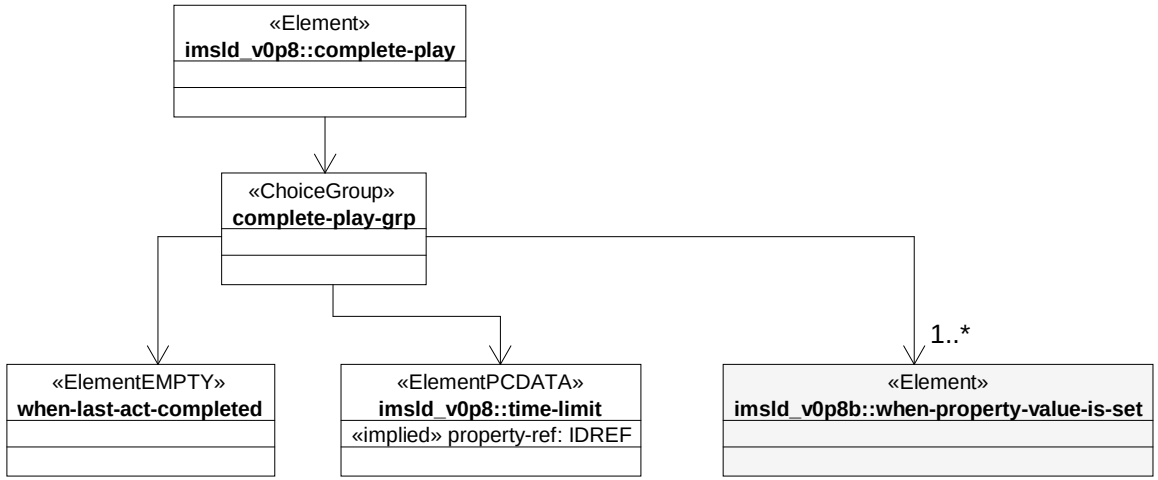
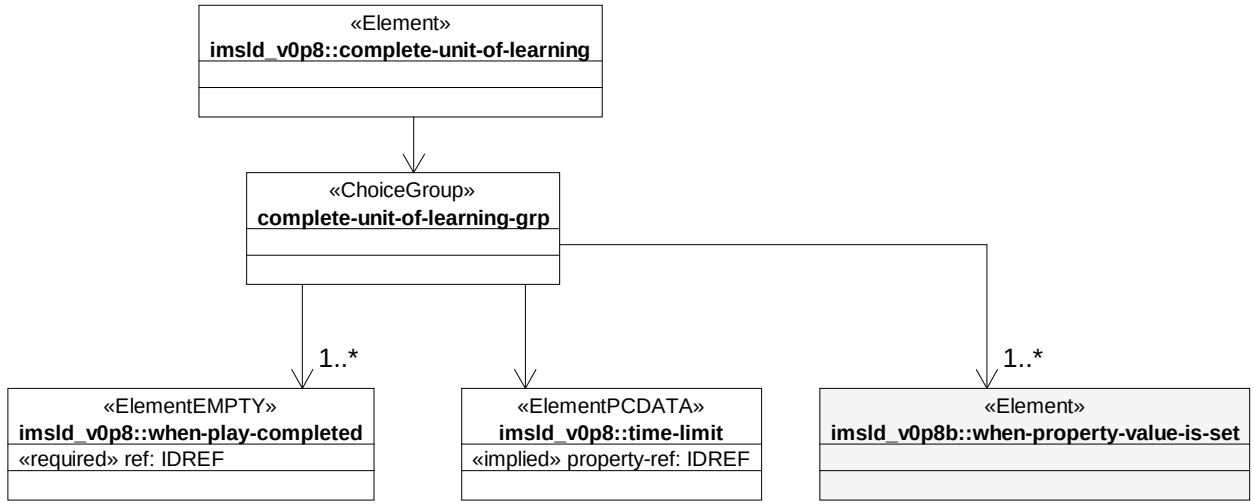
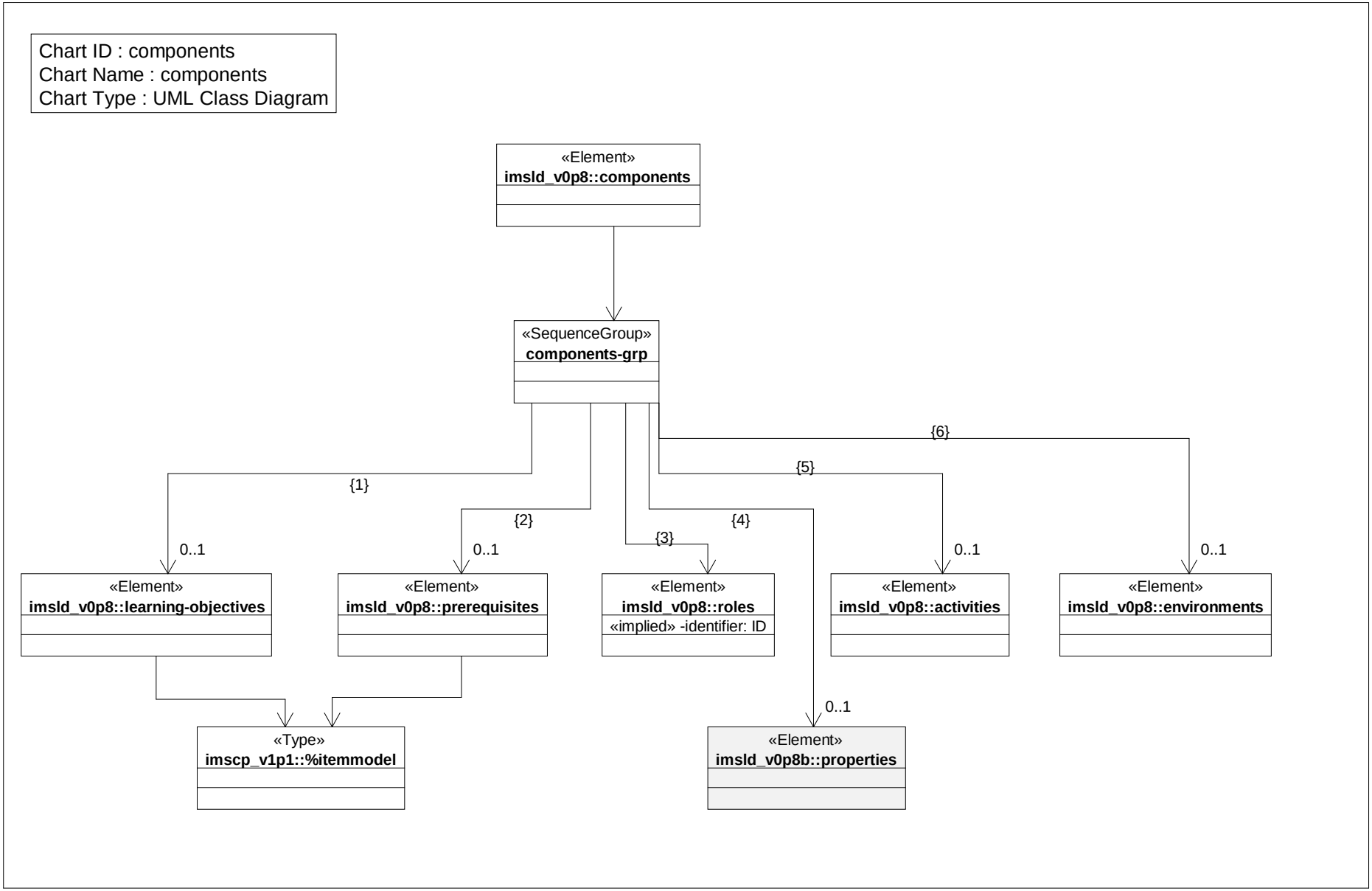


Chart ID : complete-unit-of-learning
Chart Name : complete unit of learning
Chart Type : UML Class Diagram



Class Diagram: components



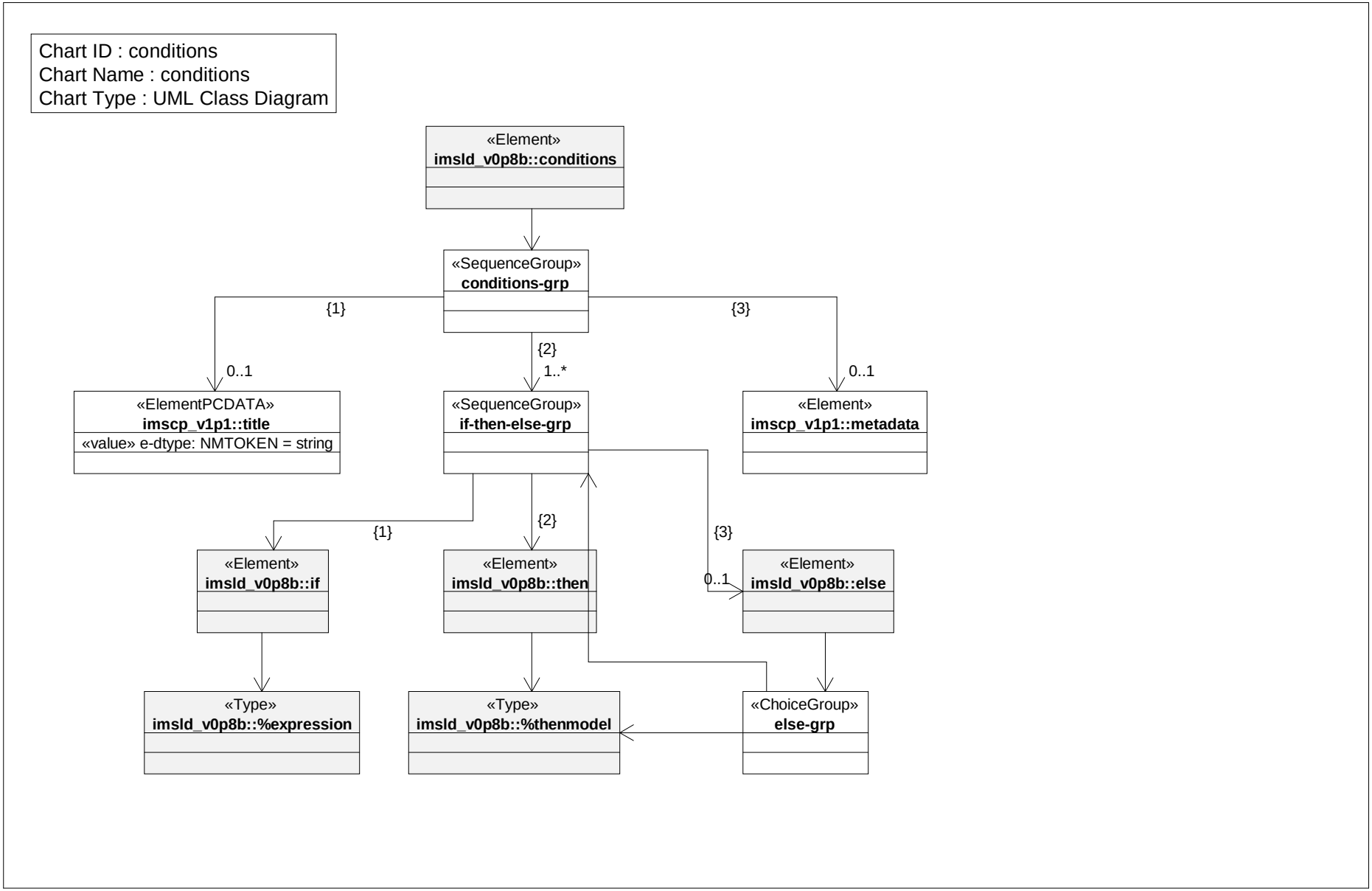
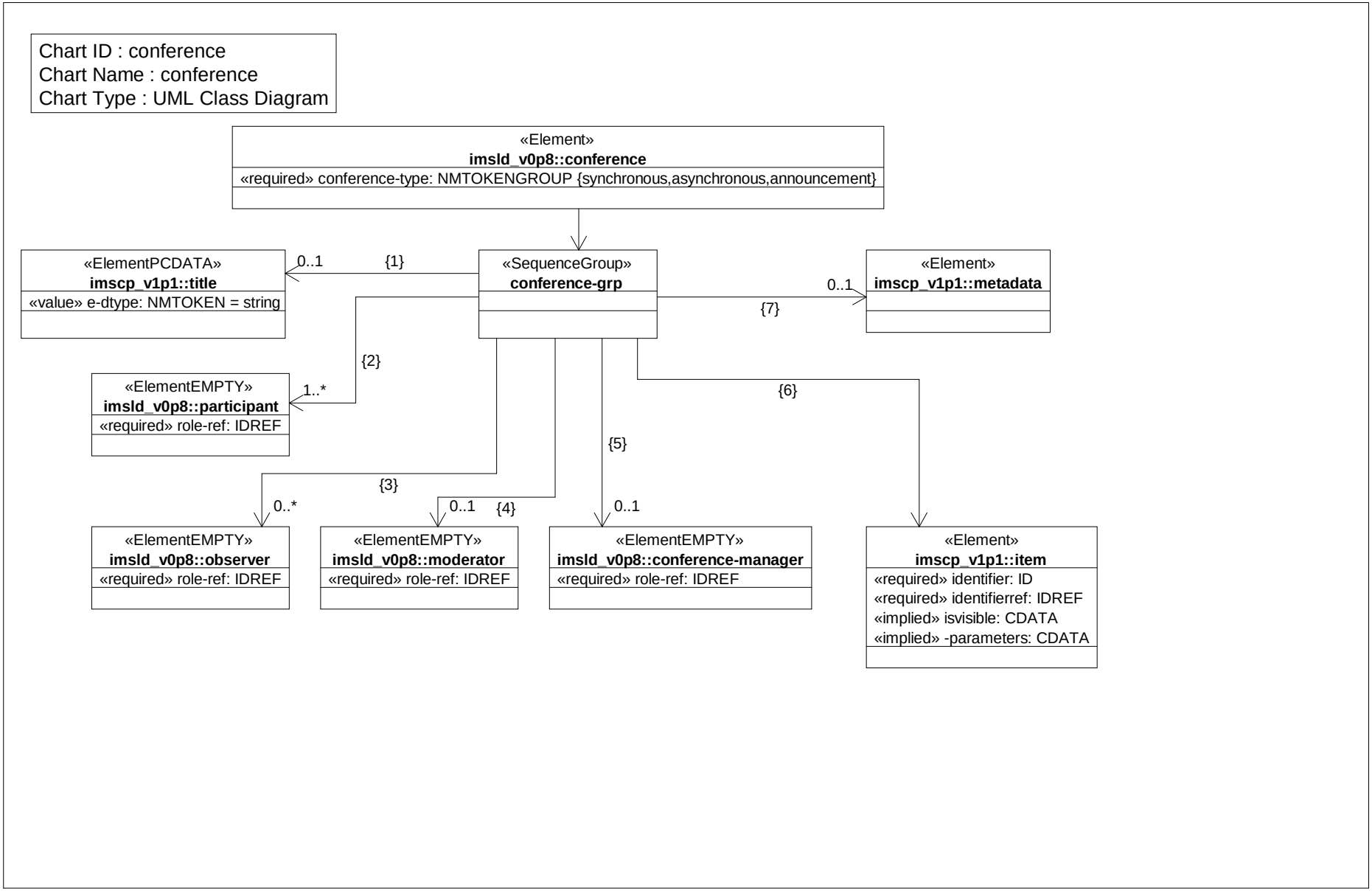
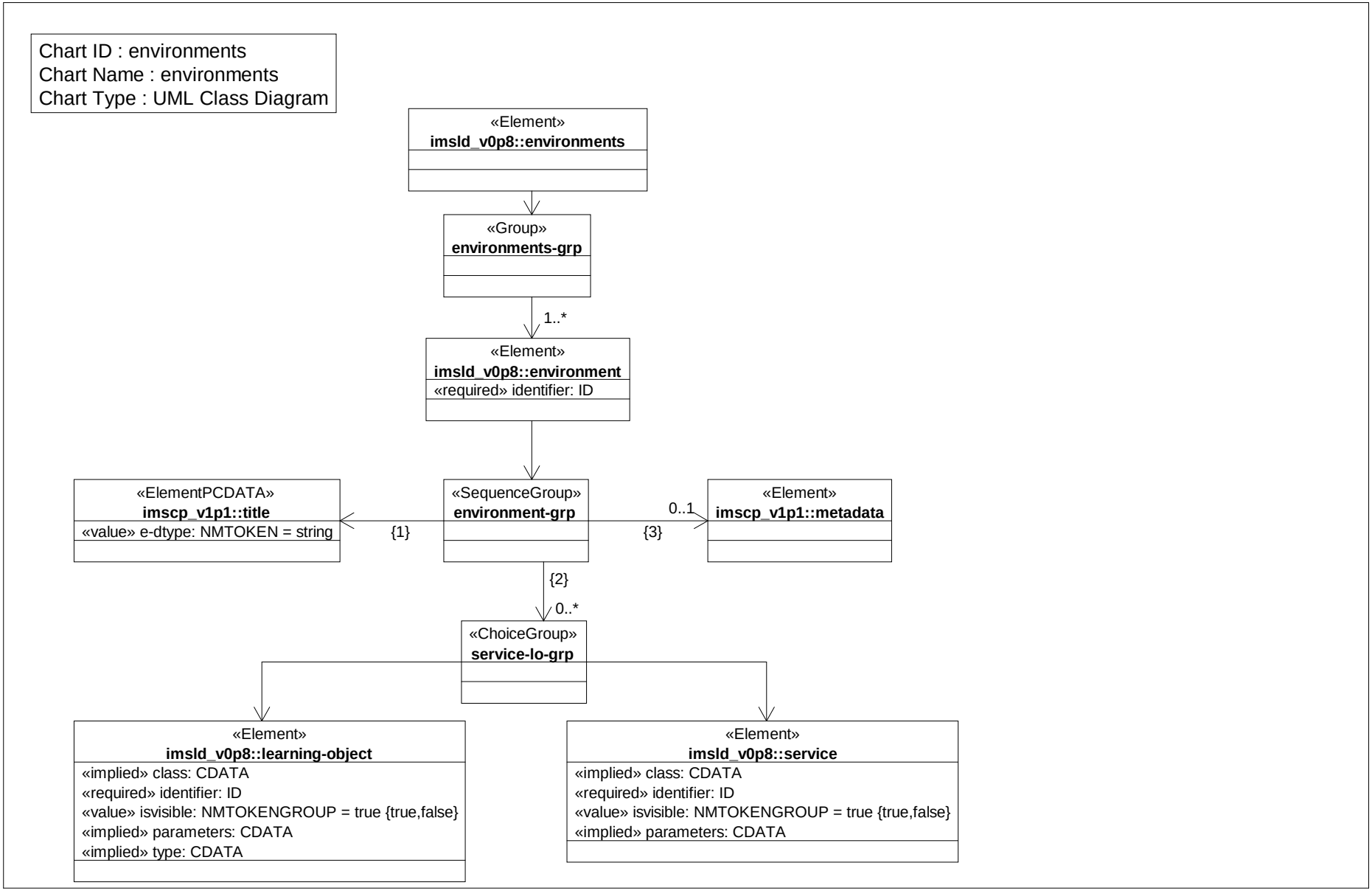


Chart ID : conditions
Chart Name : conditions
Chart Type : UML Class Diagram
«Element»
imsld_v0p8b::conditions
«SequenceGroup»
conditions-grp
{1}
{2}
{3}
0..1
1..*
0..1
«ElementPCDATA»
imscp_v1p1::title
«value» e-dtype: NMTOKEN = string
«SequenceGroup»
if-then-else-grp
«Element»
imscp_v1p1::metadata
{1}
{2}
{3}
0..1
«Element»
imsld_v0p8b::if
«Element»
imsld_v0p8b::then
«Element»
imsld_v0p8b::else
«Type»
imsld_v0p8b::%expression
«Type»
imsld_v0p8b::%thenmodel
«ChoiceGroup»
else-grp

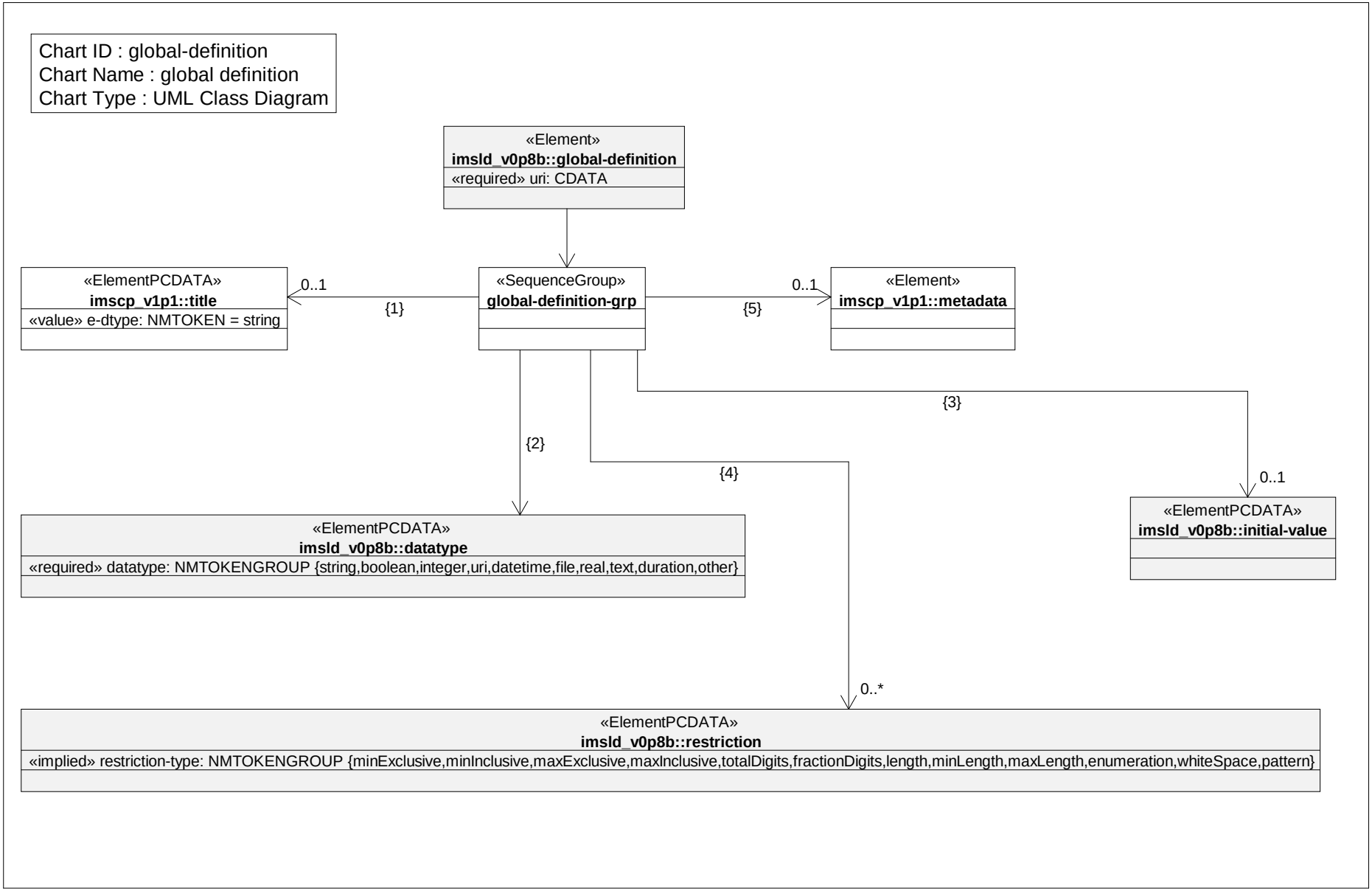
Class Diagram: conference



Class Diagram: environments



Class Diagram: global definition



Class Diagram: global elements

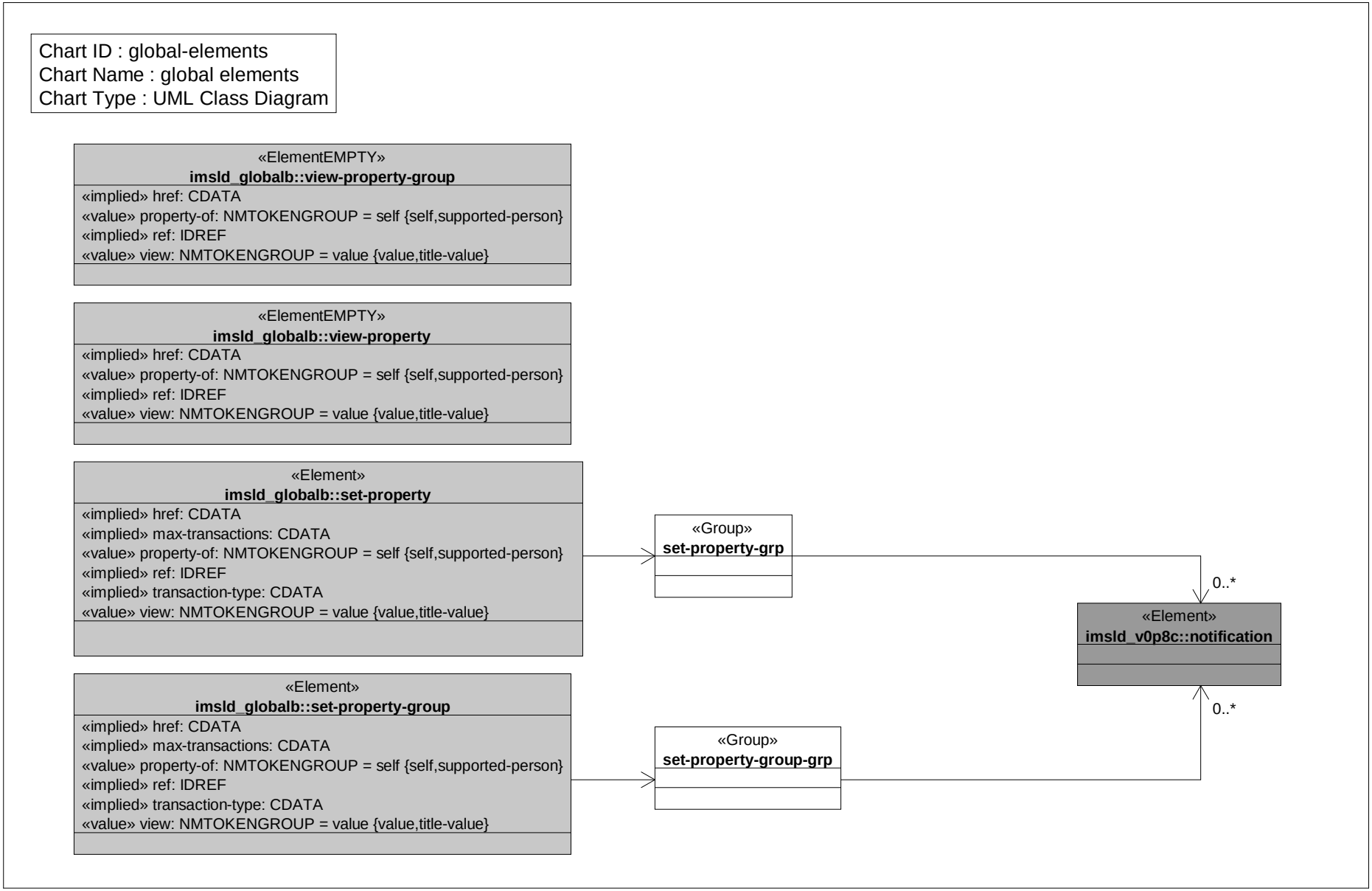


Chart ID : globpers-property
Chart Name : global personal property
Chart Type : UML Class Diagram

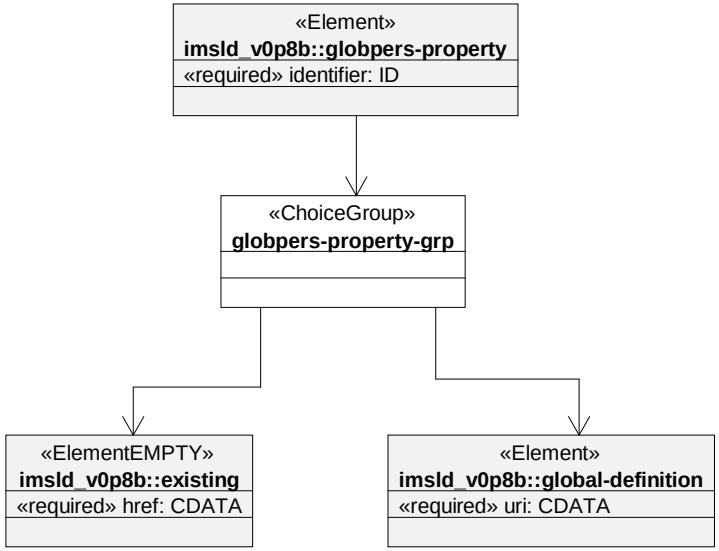
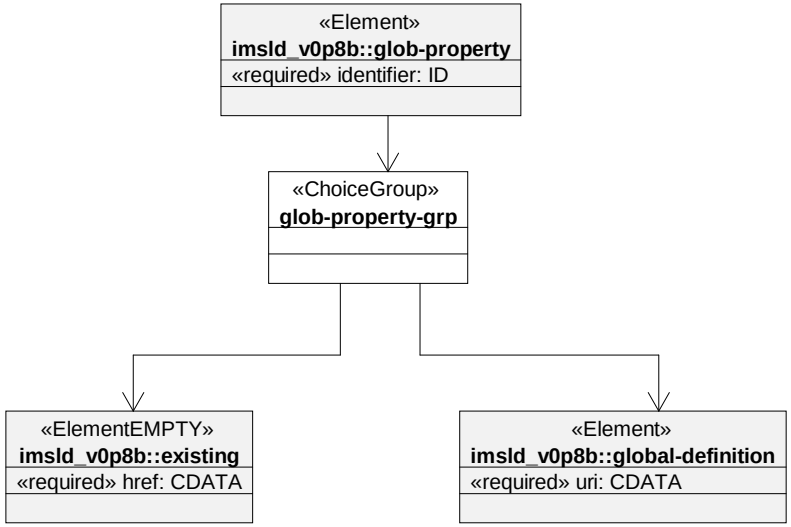
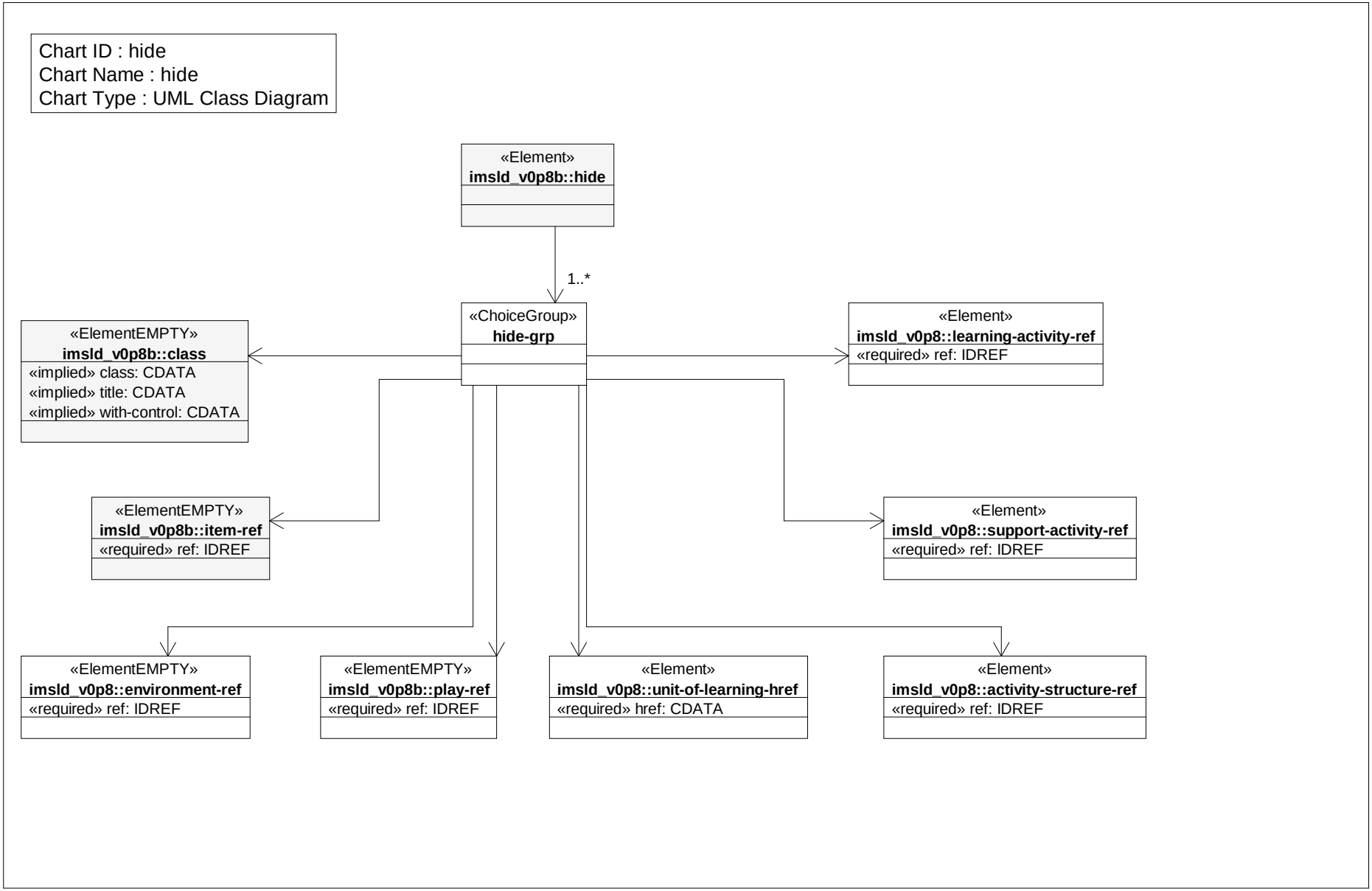


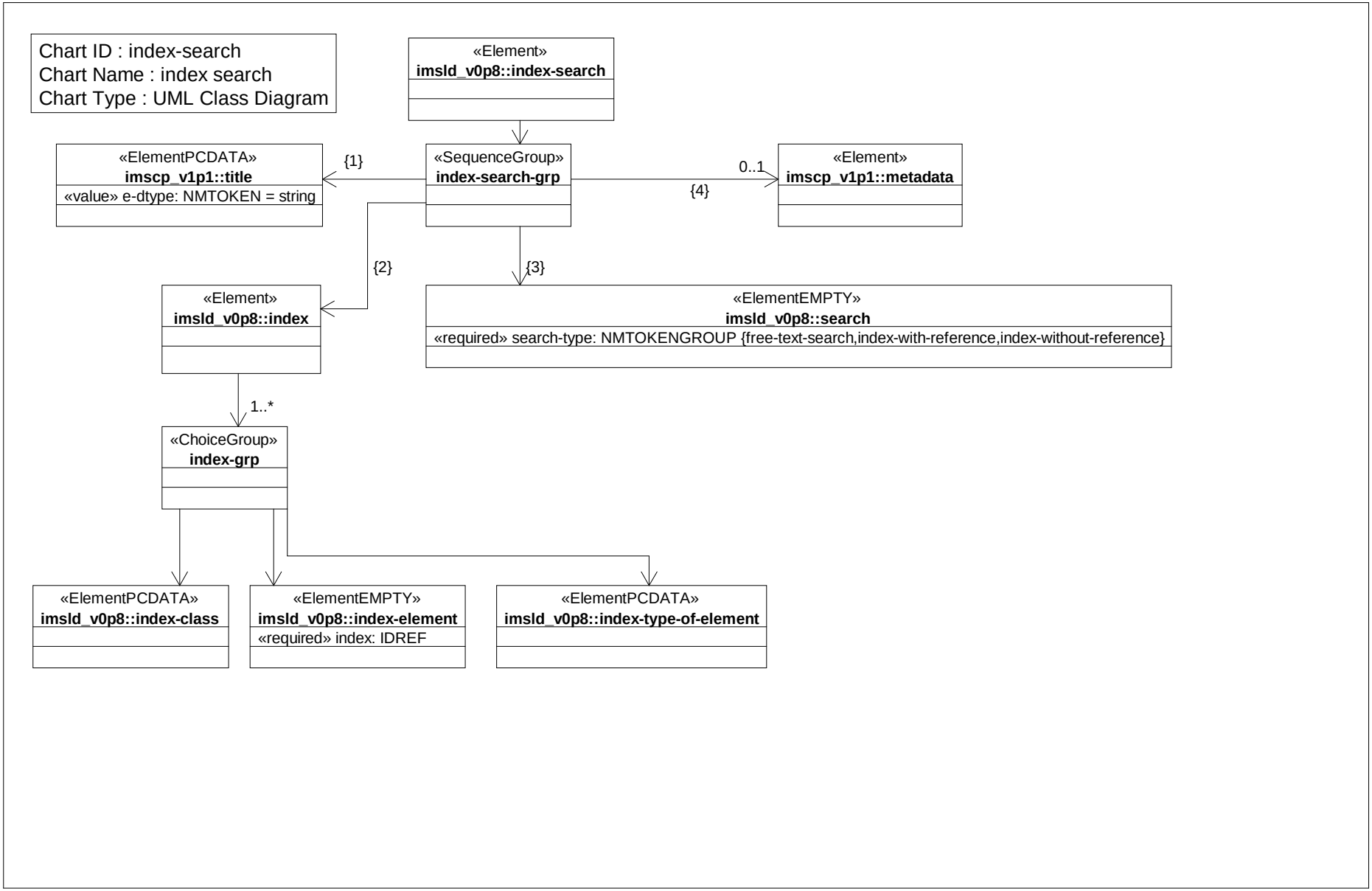
Chart ID : glob-property
Chart Name : global property
Chart Type : UML Class Diagram



Class Diagram: hide



Class Diagram: index search



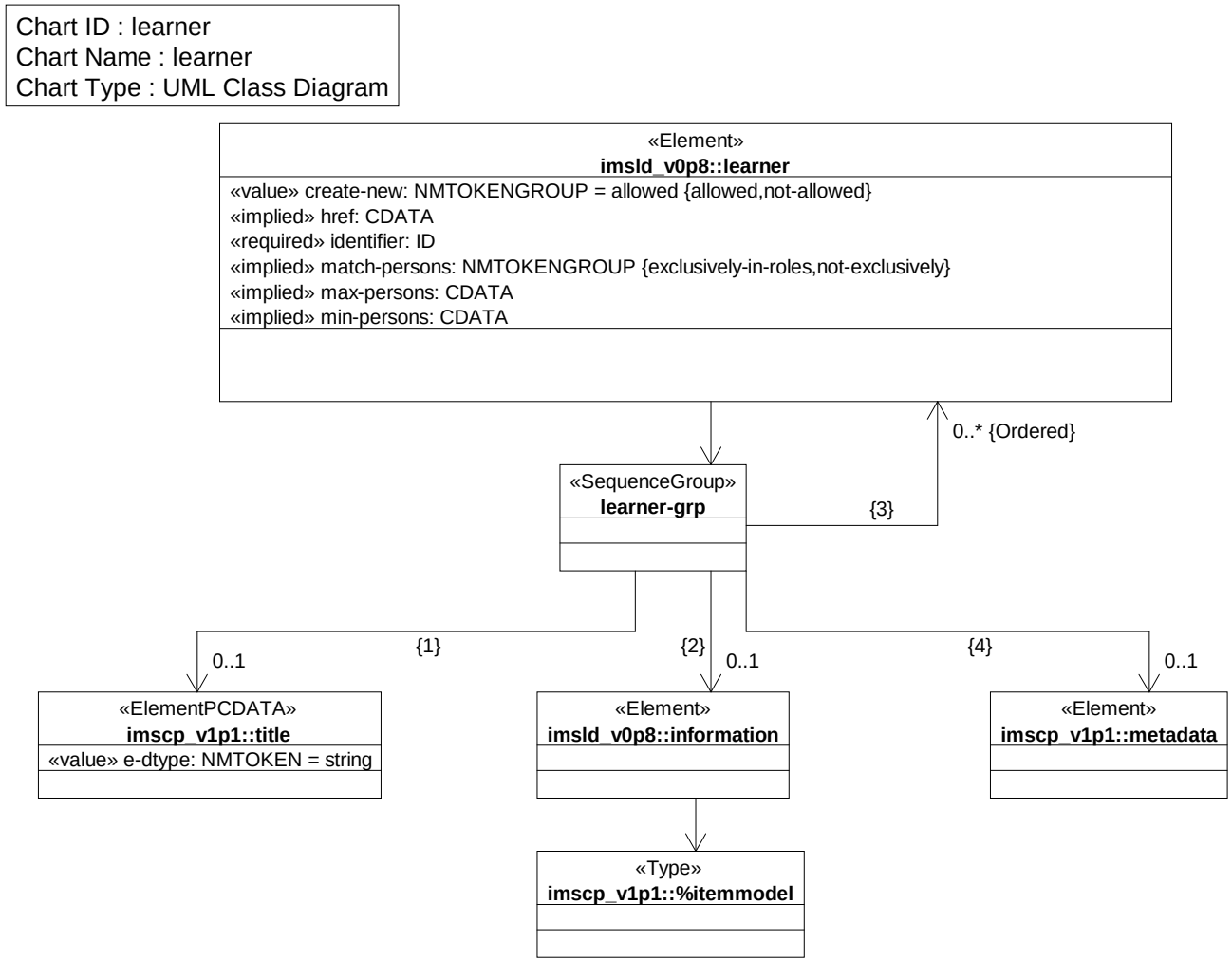


Chart ID : learner
Chart Name : learner
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::learner
«value» create-new: NMTOKENGROUP = allowed {allowed,not-allowed}
«implied» href: CDATA
«required» identifier: ID
«implied» match-persons: NMTOKENGROUP {exclusively-in-roles,not-exclusively}
«implied» max-persons: CDATA
«implied» min-persons: CDATA
0..* {Ordered}
«SequenceGroup»
learner-grp
{3}
{1}
0..1
{2}
0..1
{4}
0..1
«ElementPCDATA»
imscp_v1p1::title
«value» e-dtype: NMTOKEN = string
«Element»
imsld_v0p8::information
«Element»
imscp_v1p1::metadata
«Type»
imscp_v1p1::%itemmodel

Class Diagram: learning activity

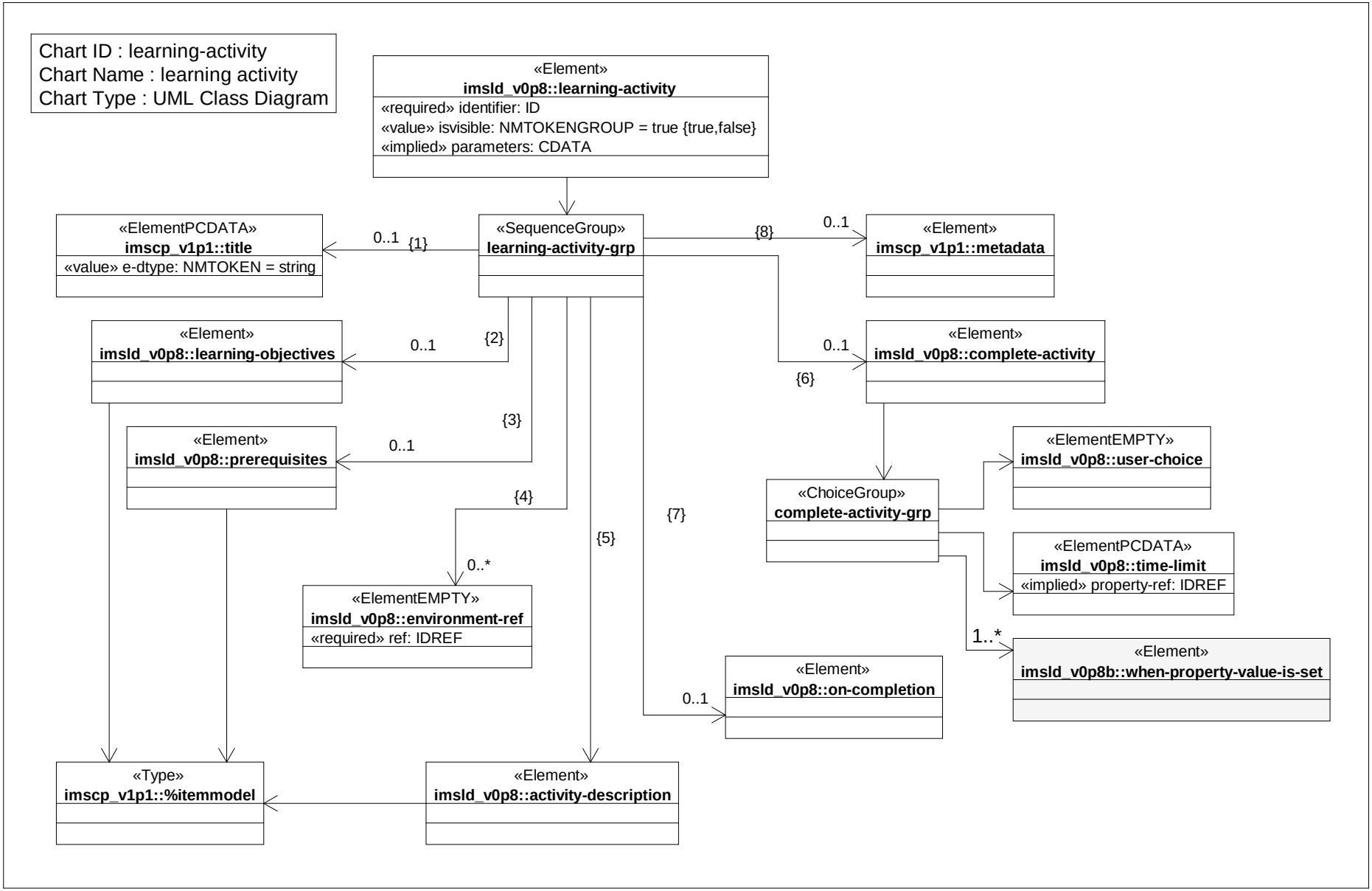


Chart ID : learning-activity-ref
Chart Name : learning activity ref
Chart Type : UML Class Diagram

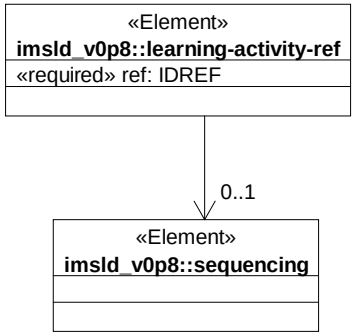


Chart ID : learning-design
Chart Name : learning design
Chart Type : UML Class Diagram

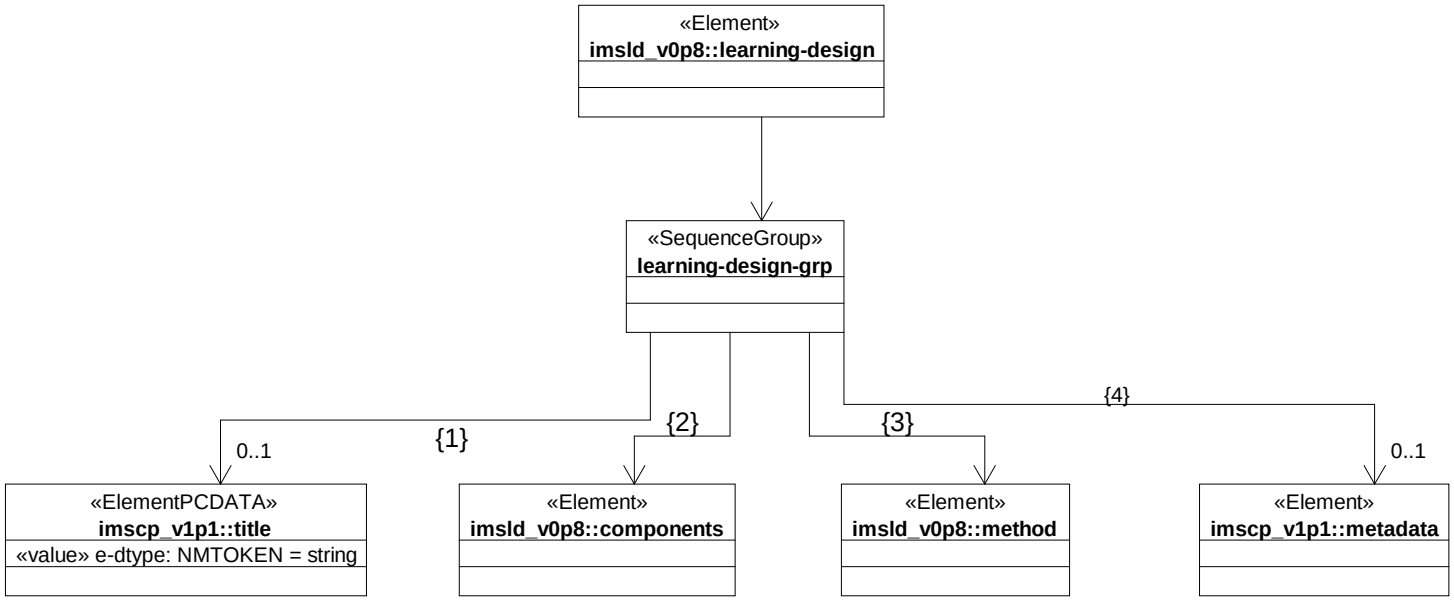
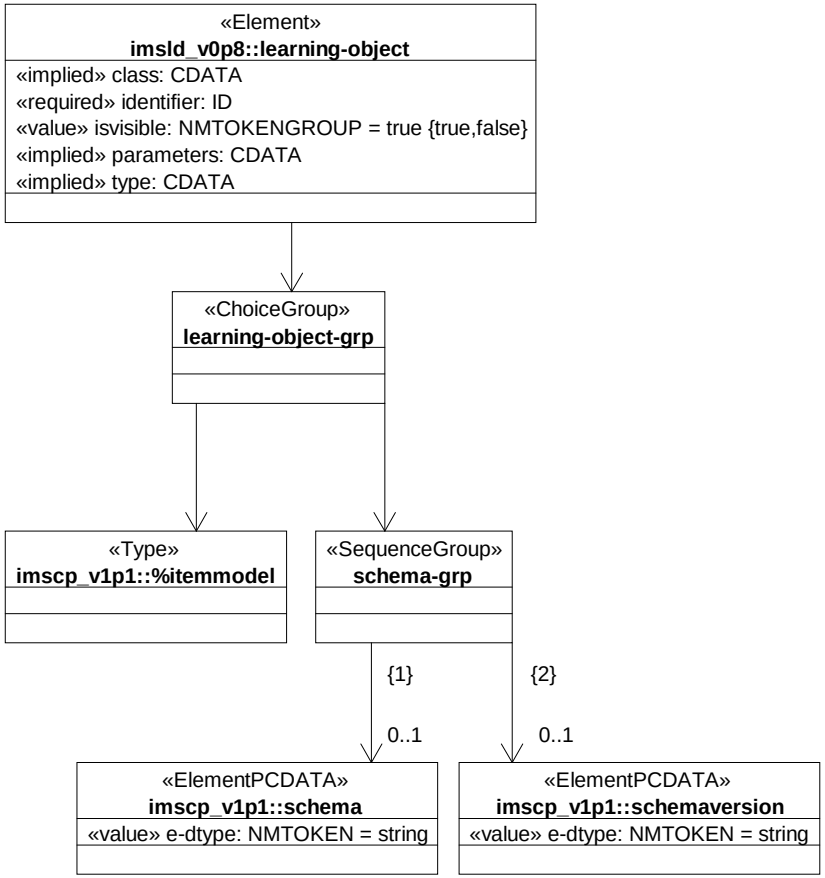
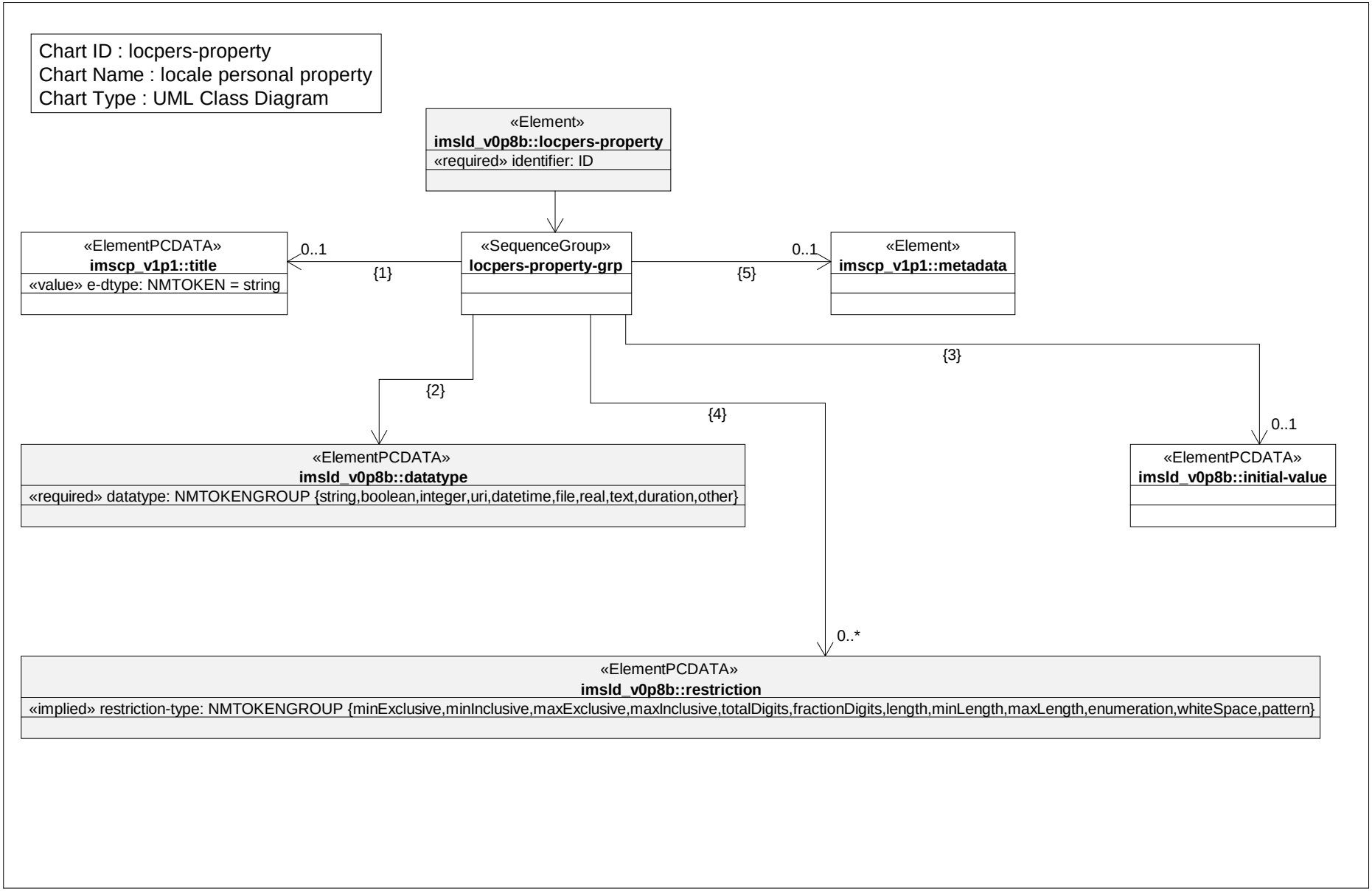


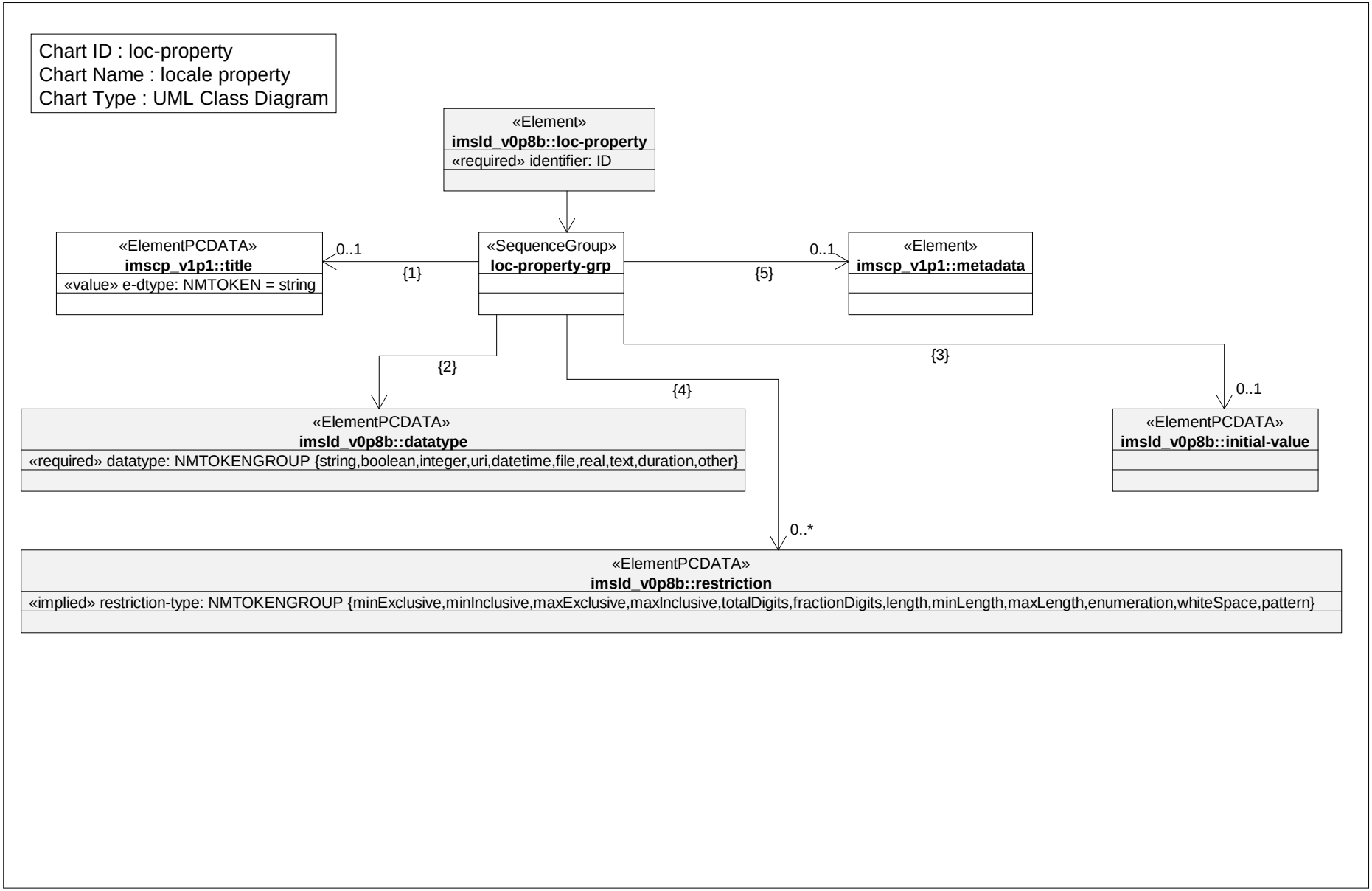
Chart ID : learning-object
Chart Name : learning object
Chart Type : UML Class Diagram



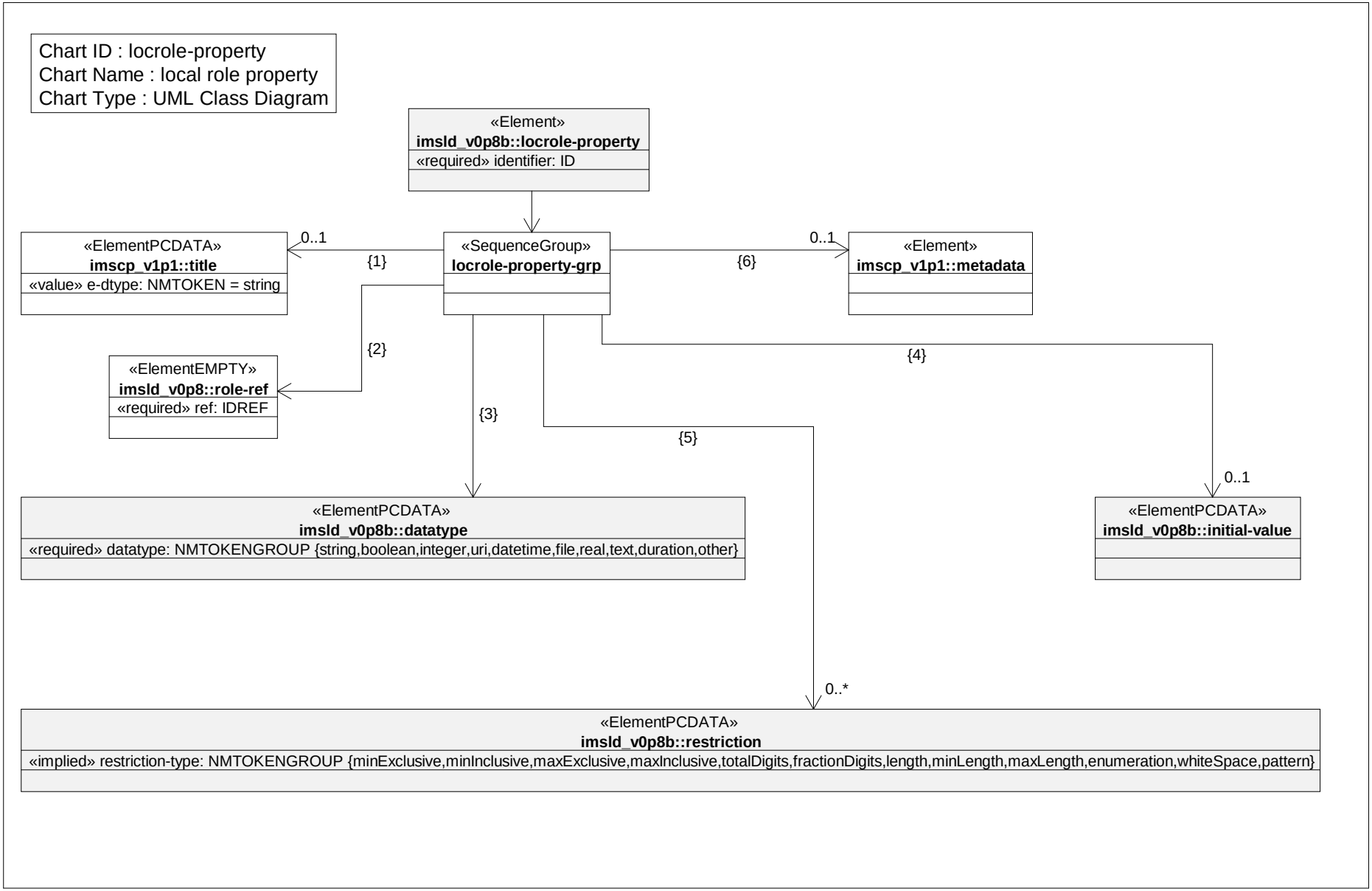
Class Diagram: locale personal property



Class Diagram: locale property



Class Diagram: local role property



Class Diagram: manifest

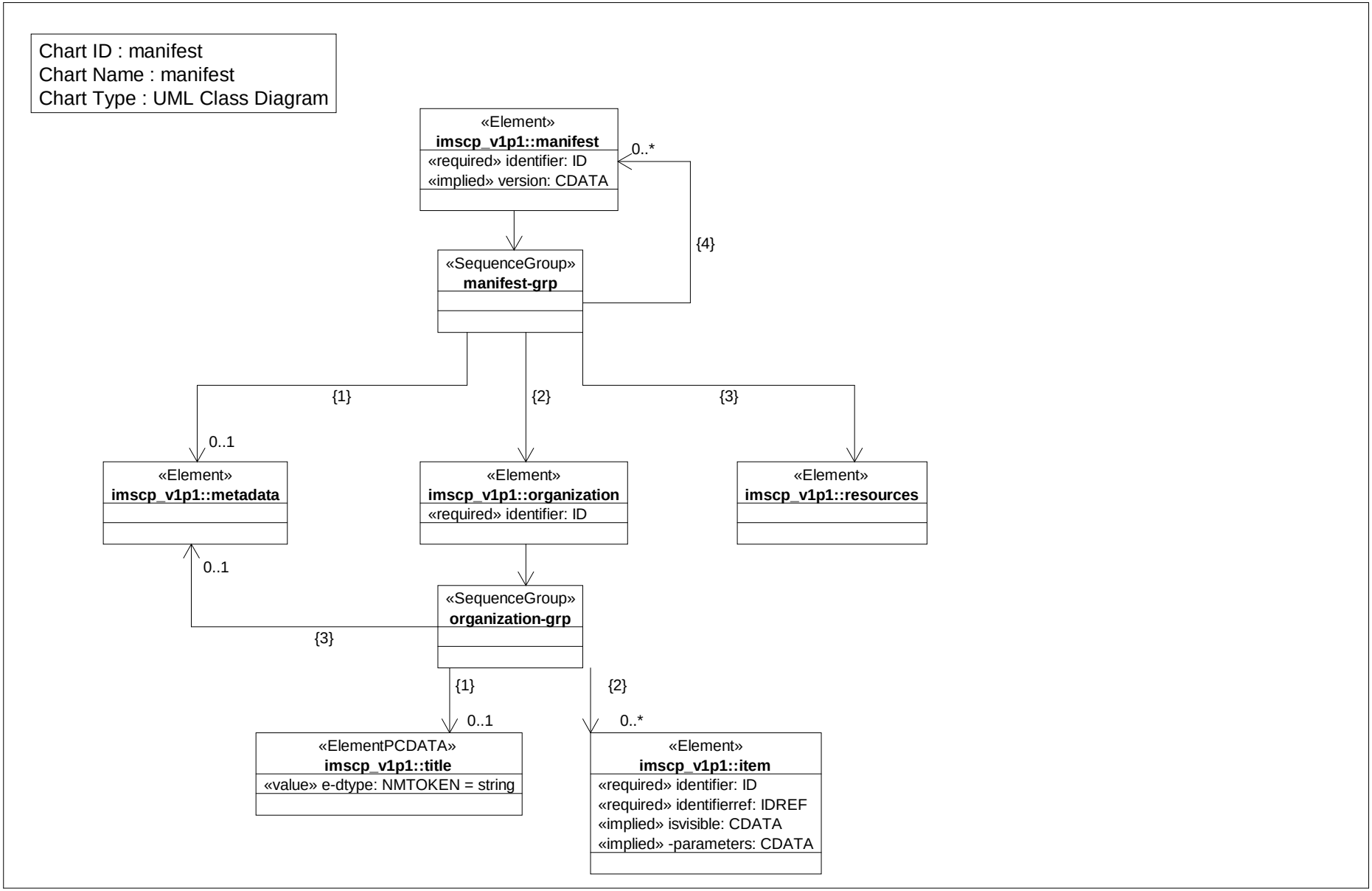
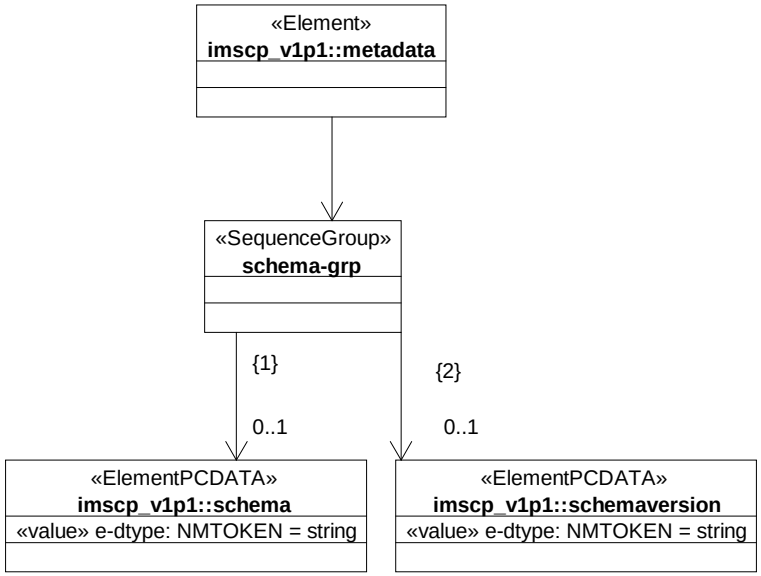


Chart ID : metadata
Chart Name : metadata
Chart Type : UML Class Diagram



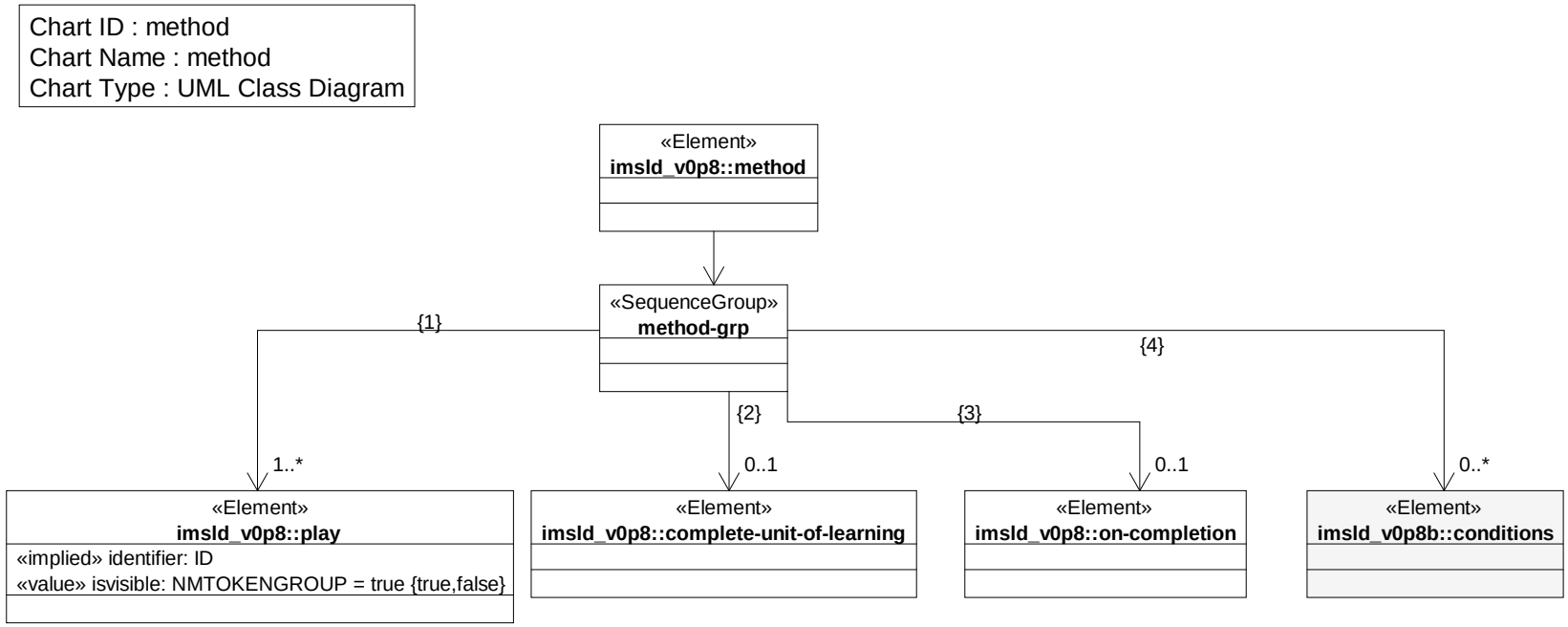
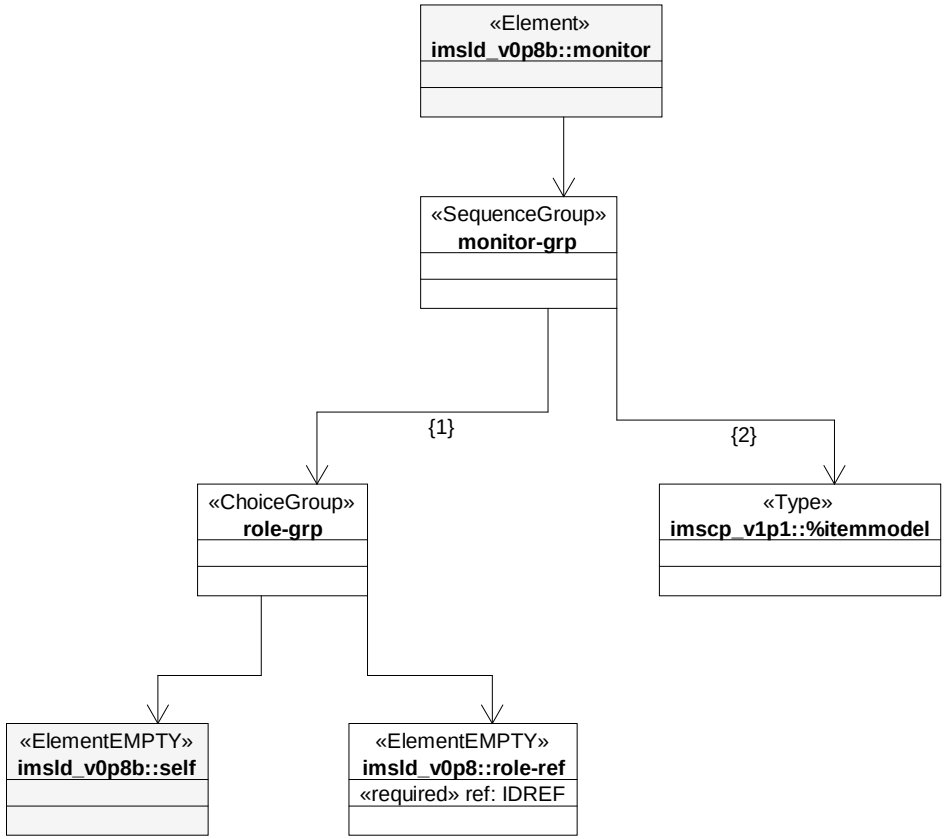


Chart ID : method
Chart Name : method
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::method
«SequenceGroup»
method-grp
{1}
{2}
{3}
{4}
1..*
0..1
0..1
0..*
«Element»
imsld_v0p8::play
«implied» identifier: ID
«value» isvisible: NMTOKENGROUP = true {true,false}
«Element»
imsld_v0p8::complete-unit-of-learning
«Element»
imsld_v0p8::on-completion
«Element»
imsld_v0p8b::conditions

Chart ID : monitor
Chart Name : monitor
Chart Type : UML Class Diagram



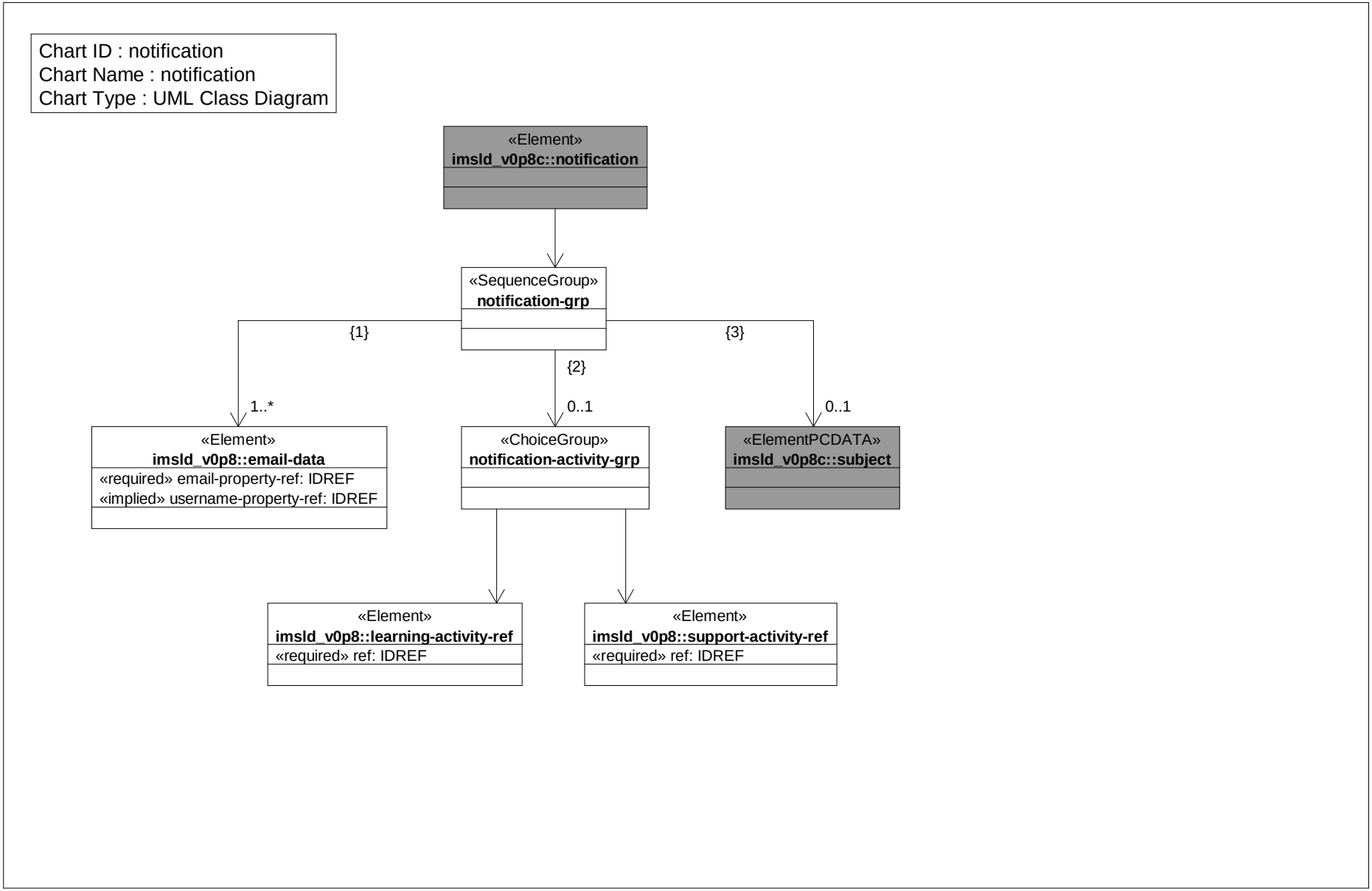
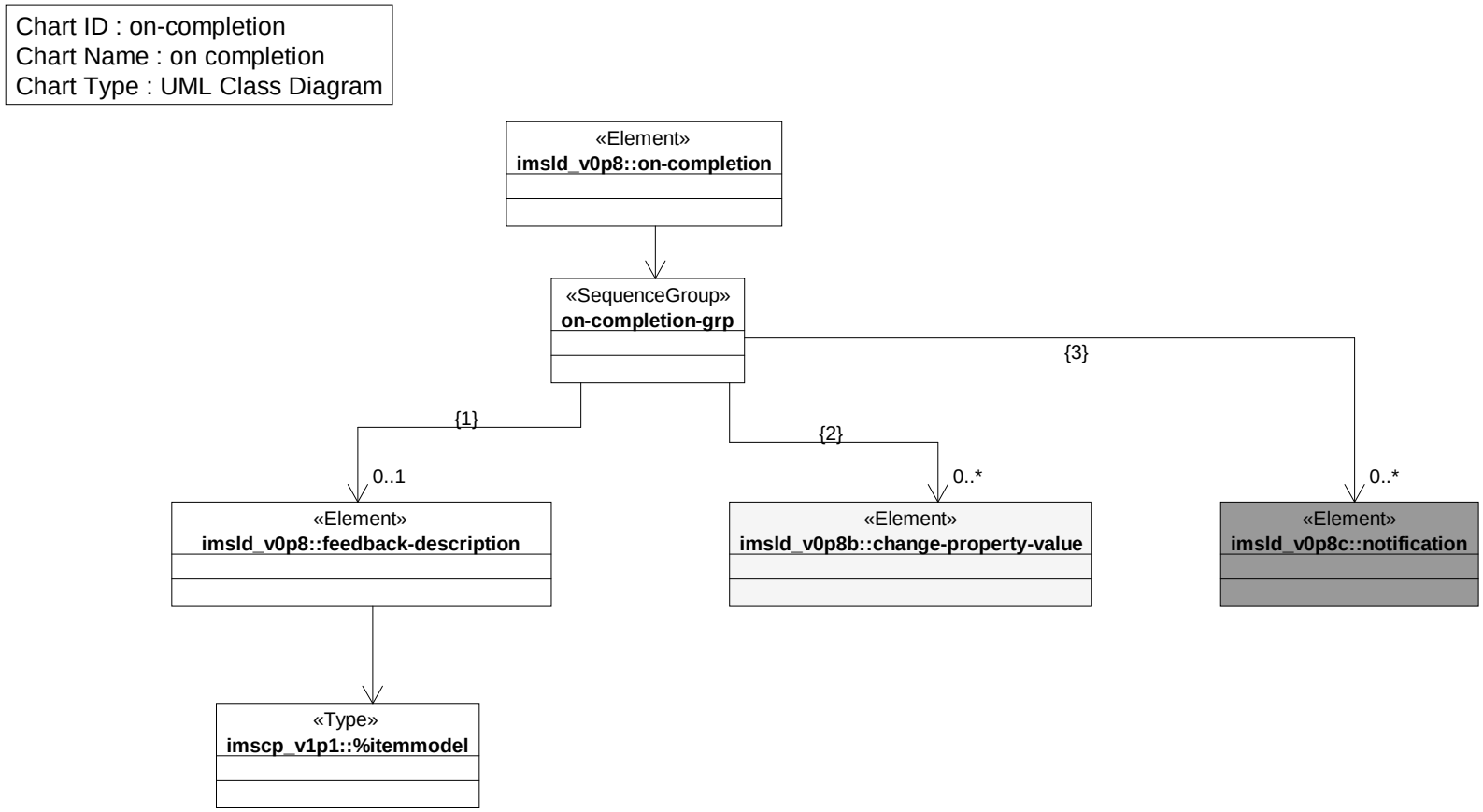
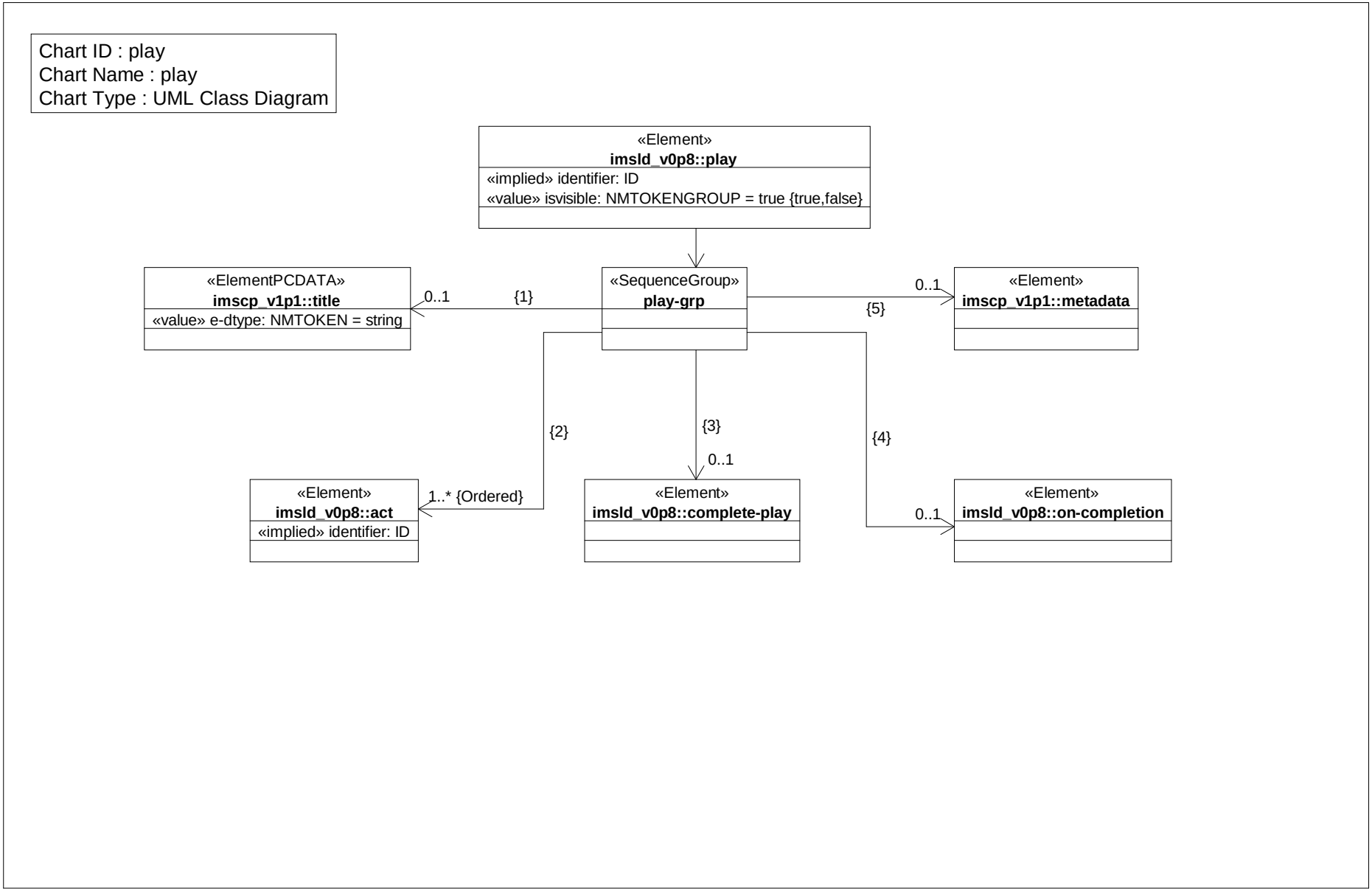


Chart ID : notification
Chart Name : notification
Chart Type : UML Class Diagram
«Element»
imsld_v0p8c::notification
«SequenceGroup»
notification-grp
{1}
{2}
{3}
1..*
0..1
0..1
«Element»
imsld_v0p8::email-data
«required» email-property-ref: IDREF
«implied» username-property-ref: IDREF
«ChoiceGroup»
notification-activity-grp
«ElementPCDATA»
imsld_v0p8c::subject
«Element»
imsld_v0p8::learning-activity-ref
«required» ref: IDREF
«Element»
imsld_v0p8::support-activity-ref
«required» ref: IDREF

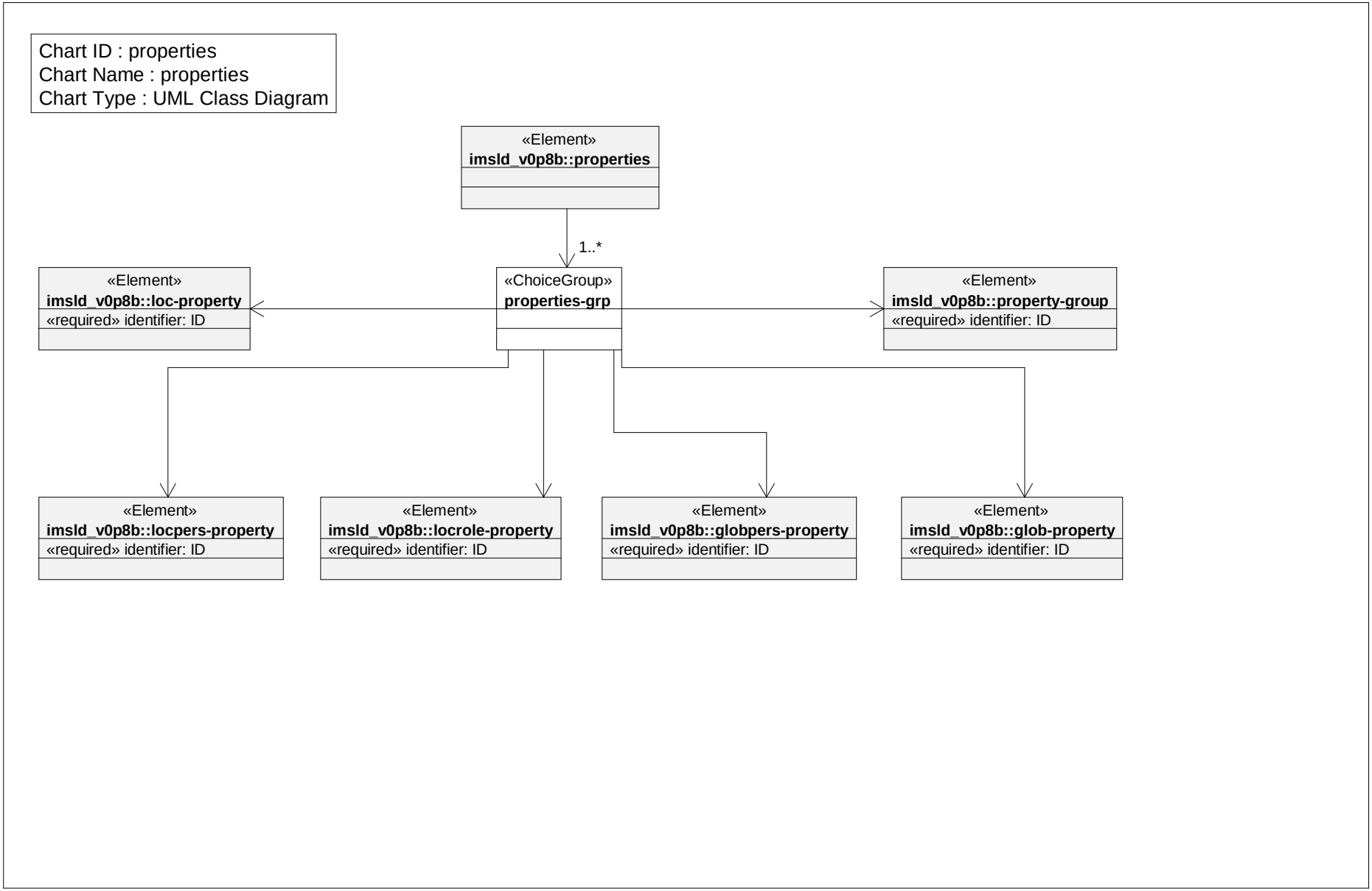
Class Diagram: on completion



Class Diagram: play



Class Diagram: properties



Class Diagram: property group

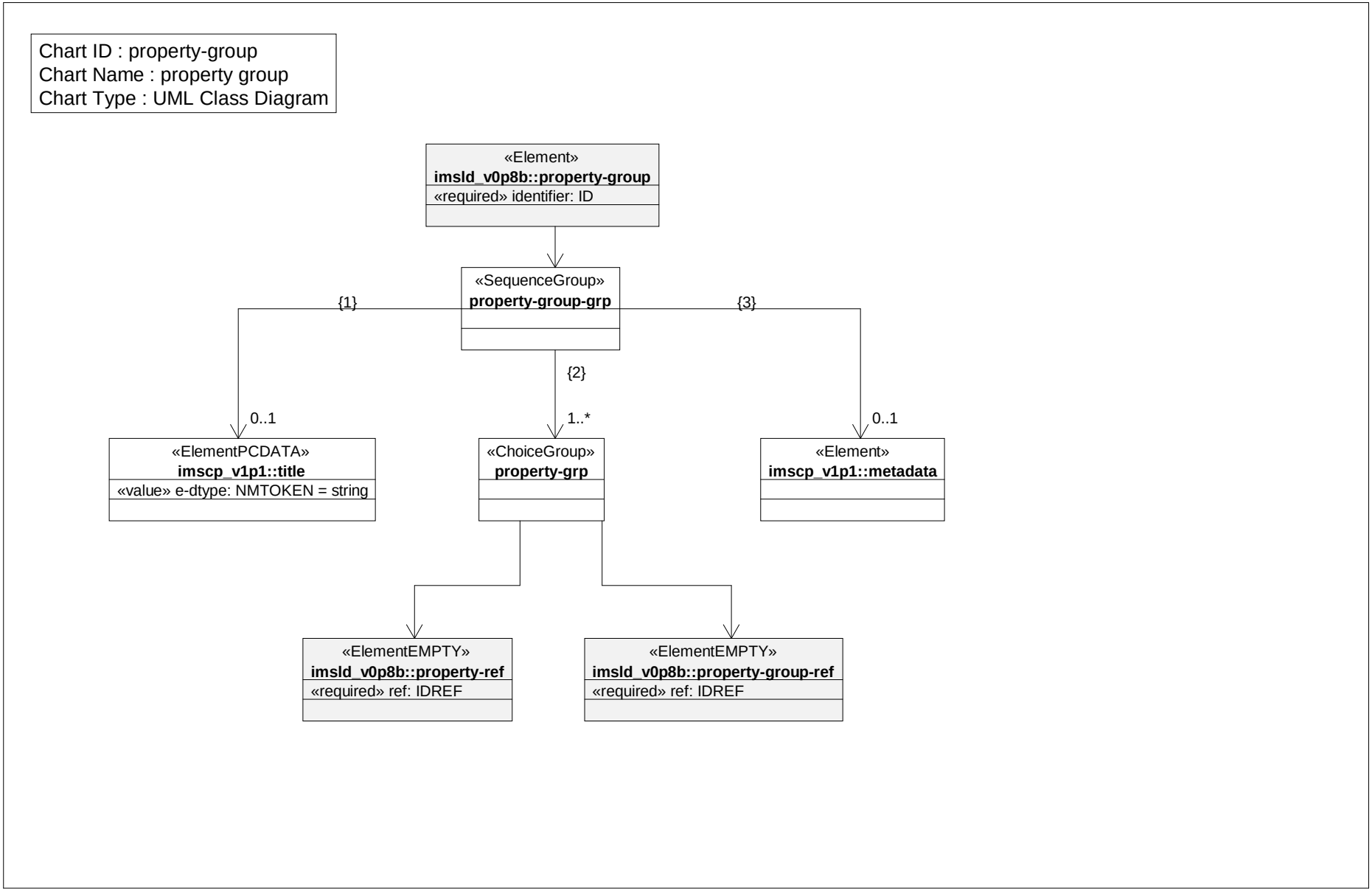
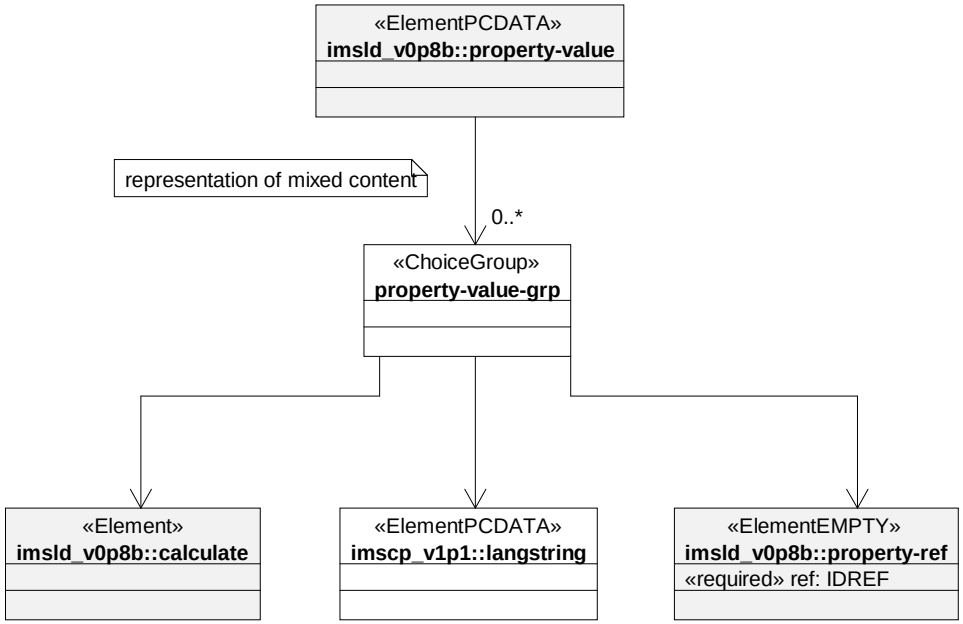
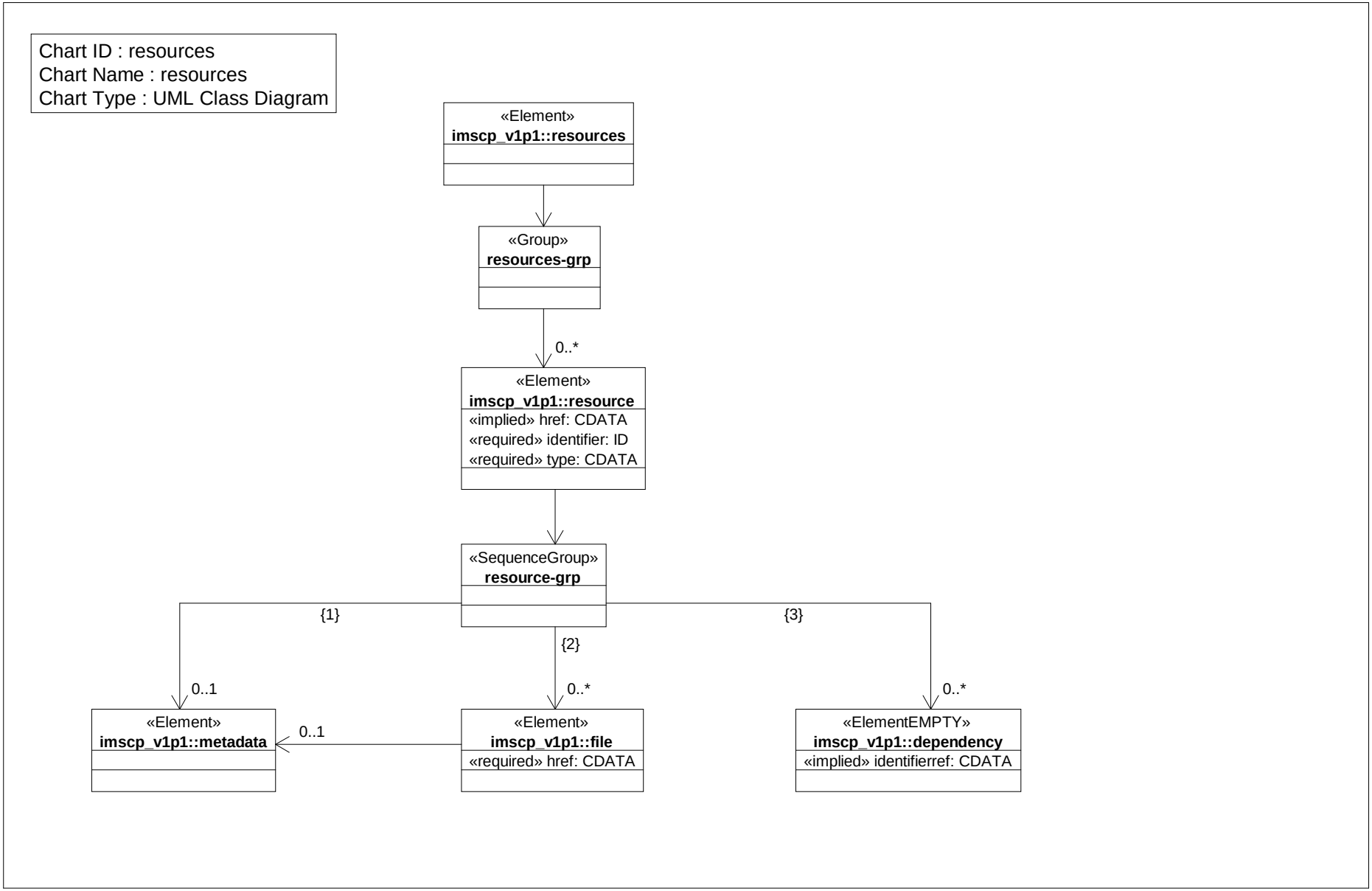


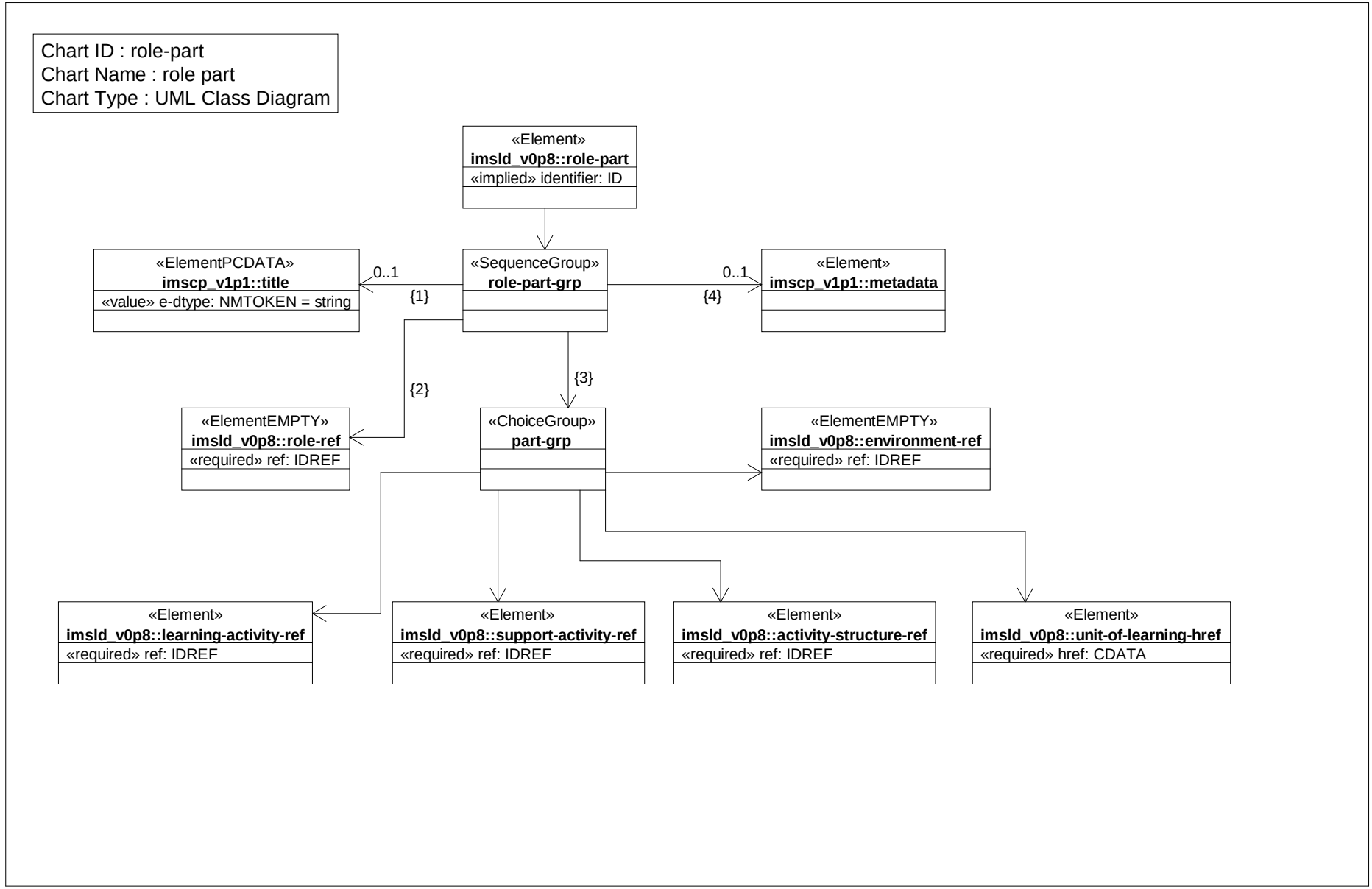
Chart ID : property-value
Chart Name : property value
Chart Type : UML Class Diagram



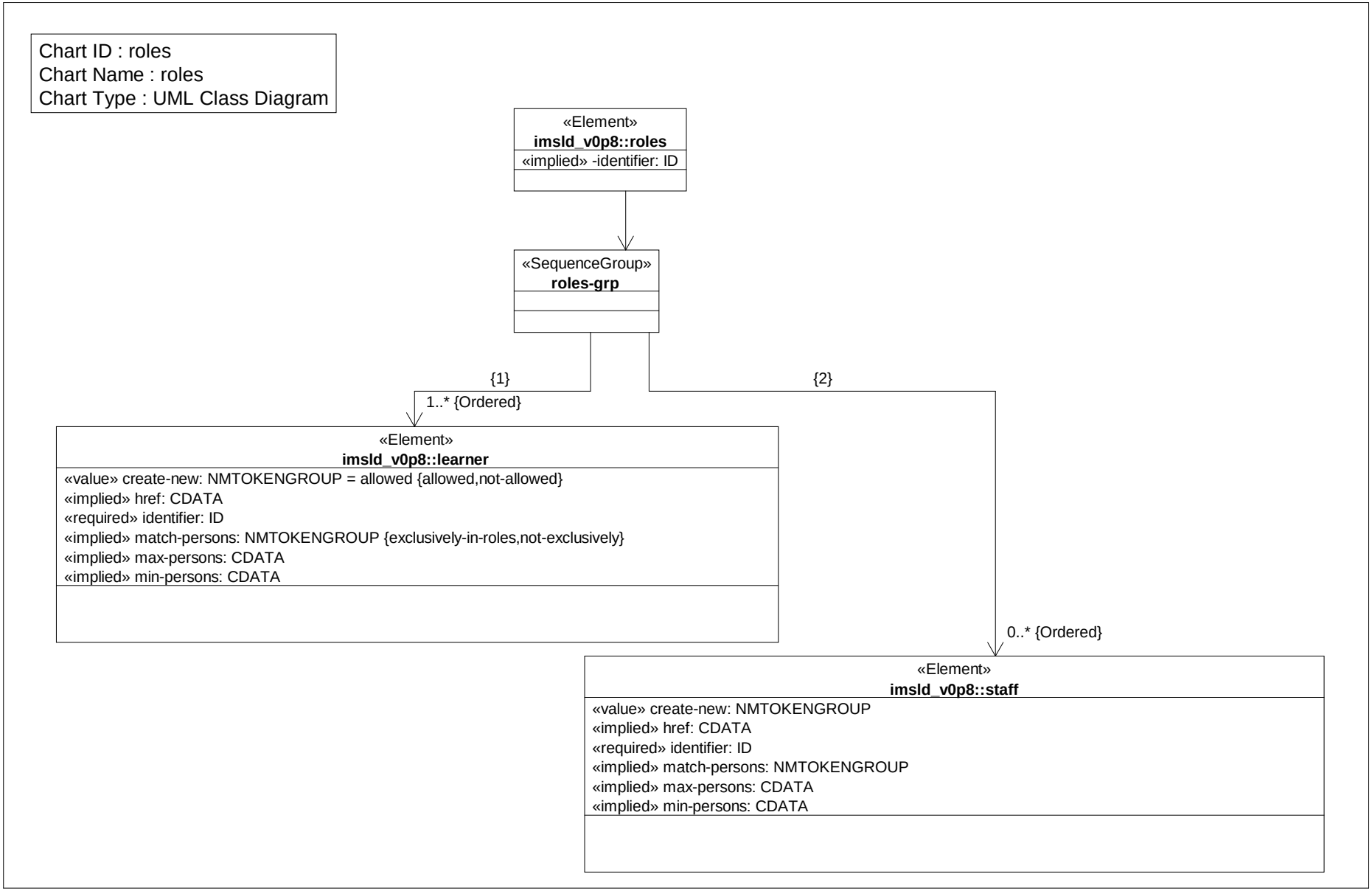
Class Diagram: resources



Class Diagram: role part



Class Diagram: roles



Class Diagram: send mail

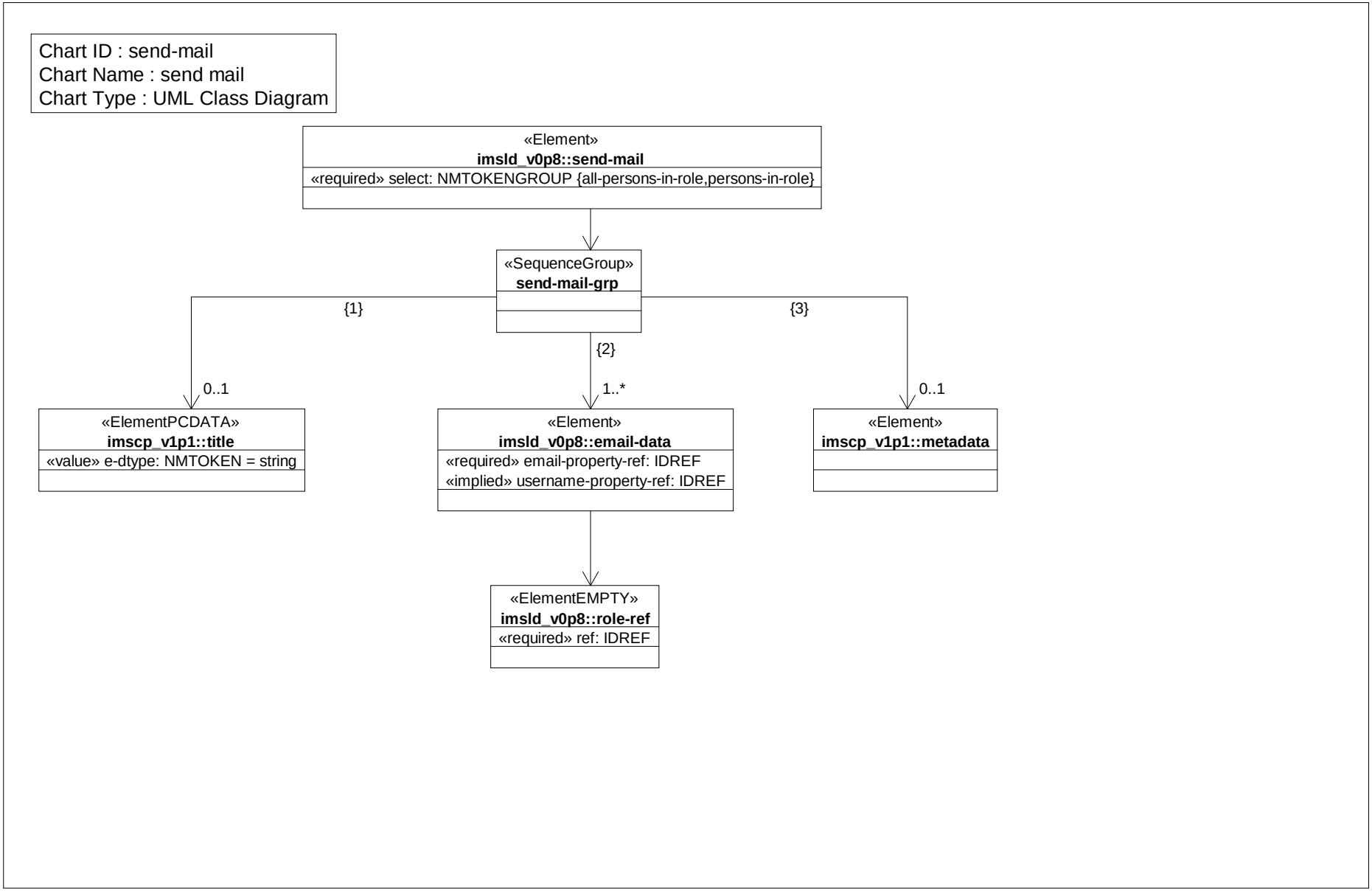
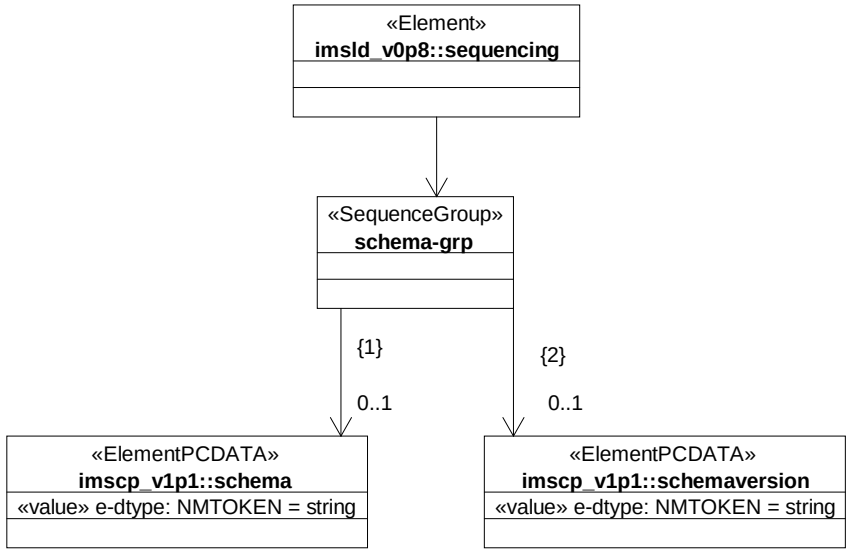
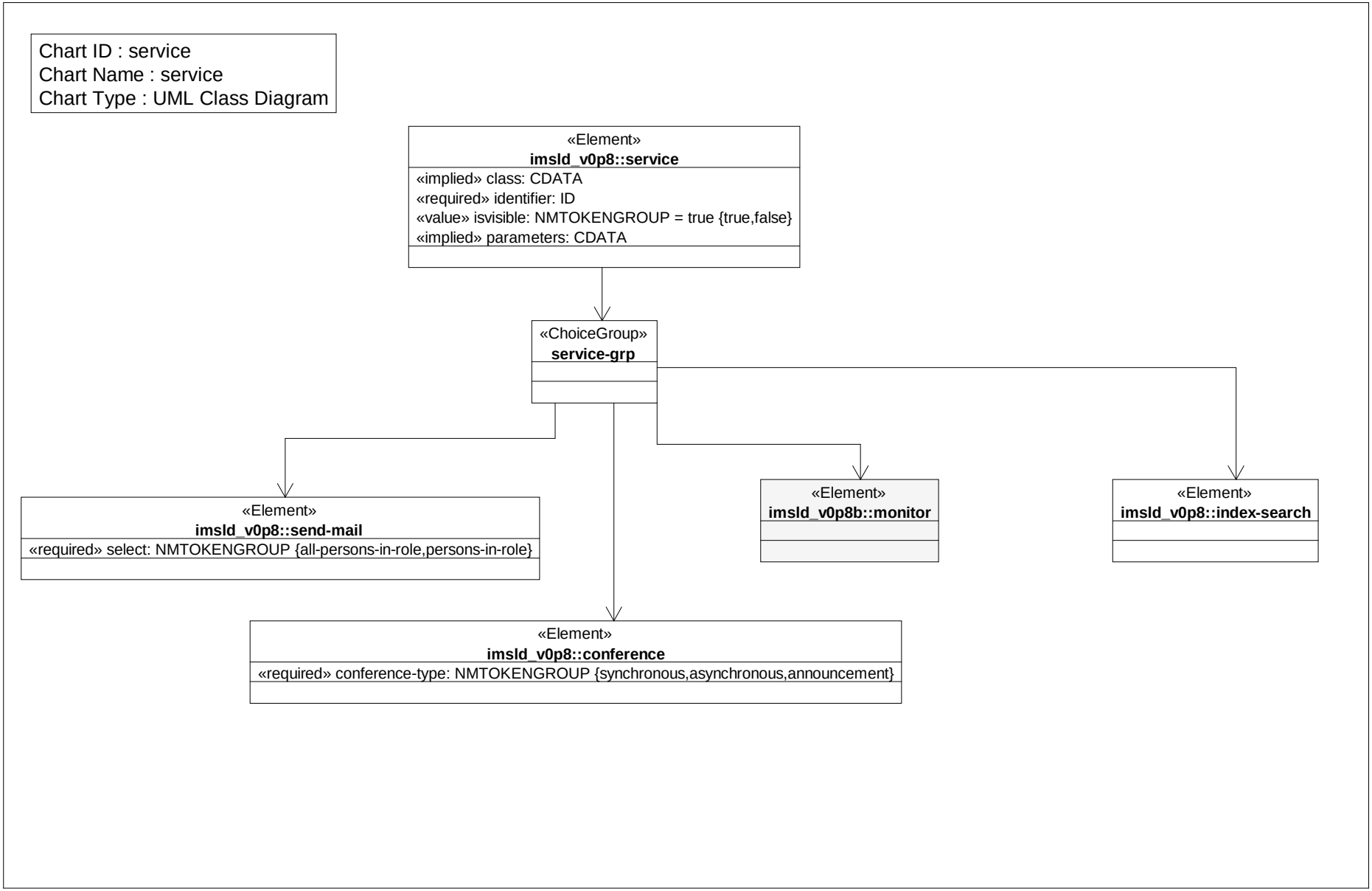


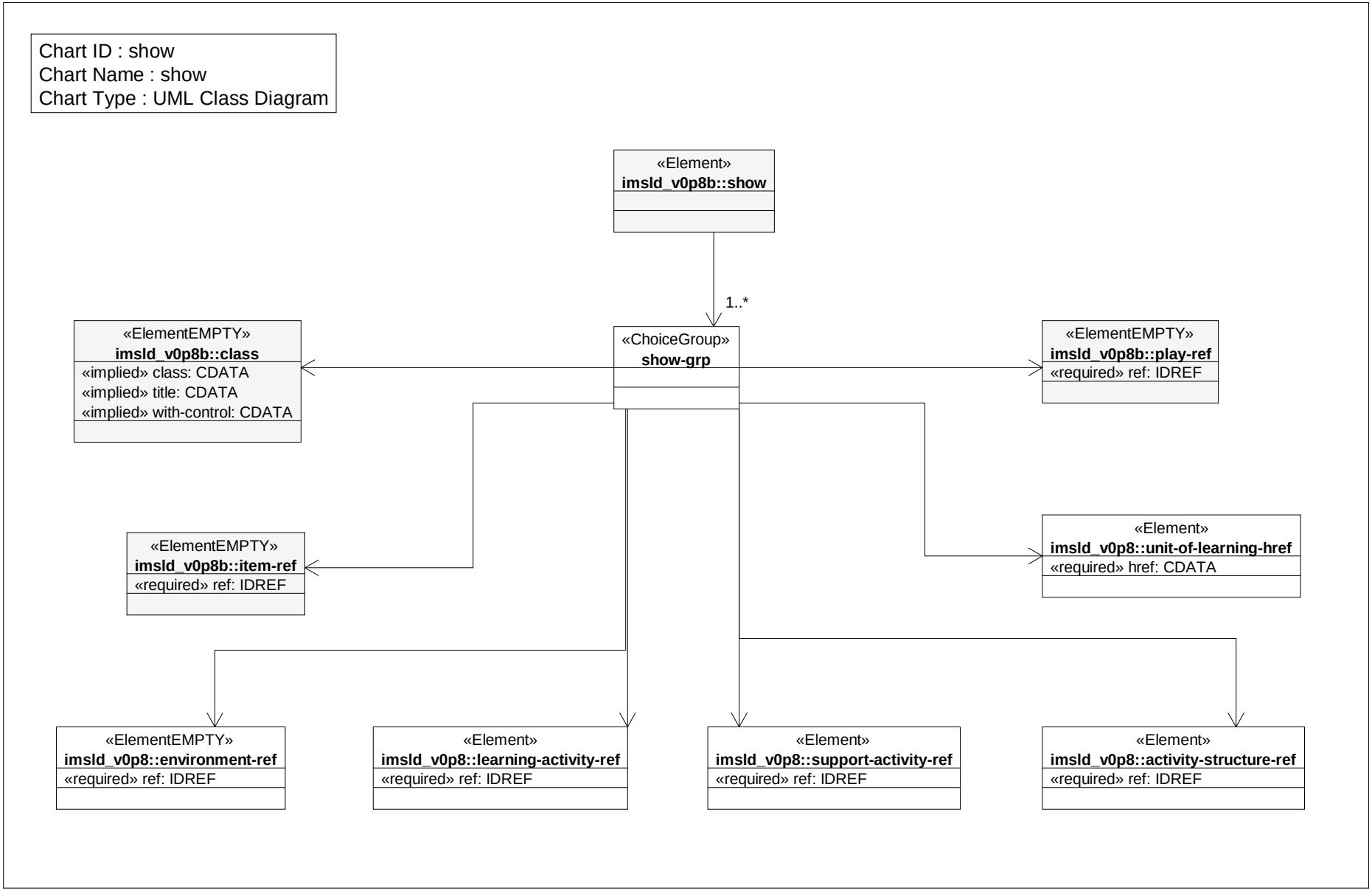
Chart ID : sequencing
Chart Name : sequencing
Chart Type : UML Class Diagram



Class Diagram: service



Class Diagram: show



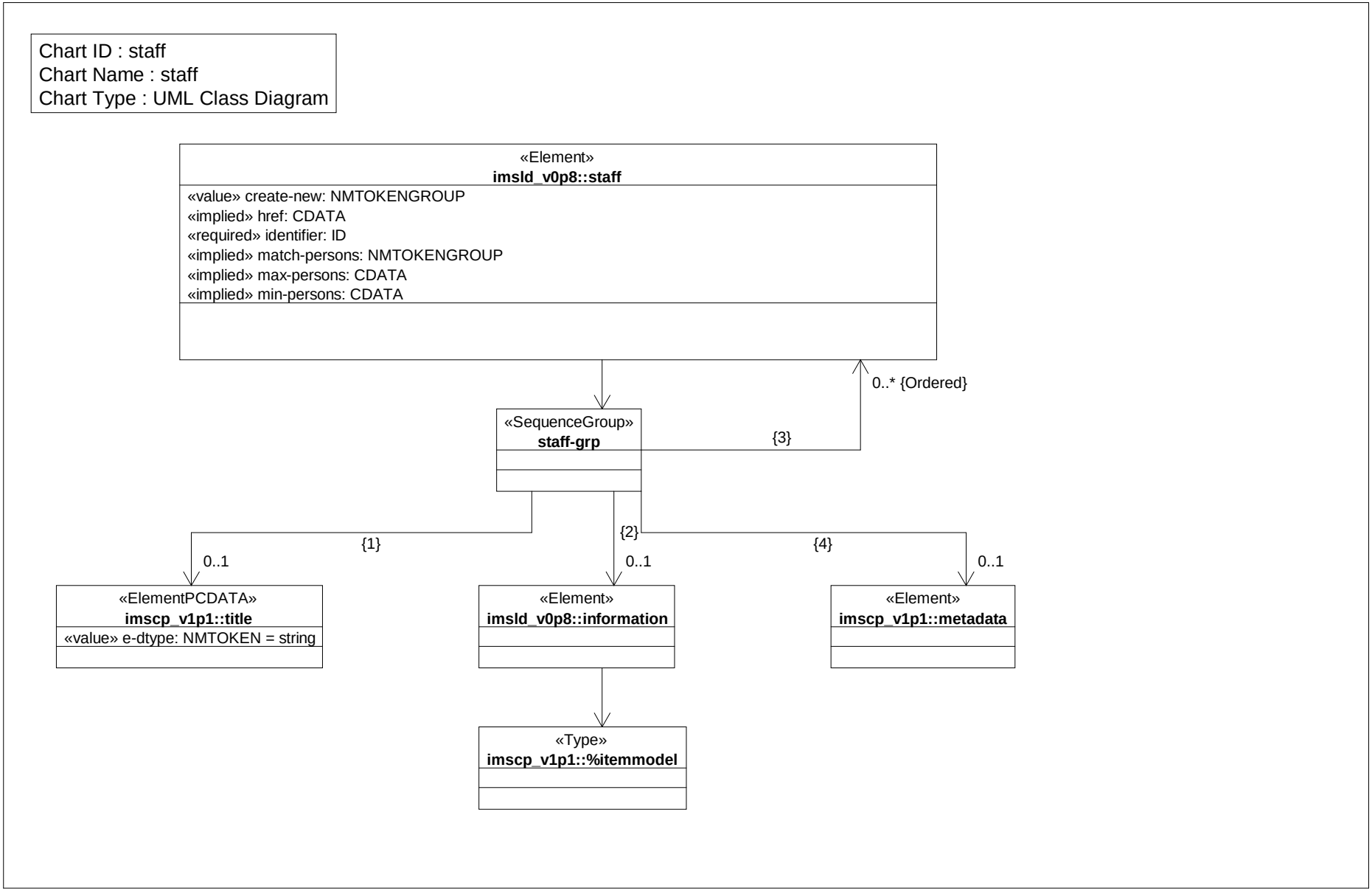


Chart ID : staff
Chart Name : staff
Chart Type : UML Class Diagram
«Element»
imsld_v0p8::staff
«value» create-new: NMTOKENGROUP
«implied» href: CDATA
«required» identifier: ID
«implied» match-persons: NMTOKENGROUP
«implied» max-persons: CDATA
«implied» min-persons: CDATA
0..* {Ordered}
«SequenceGroup»
staff-grp
{3}
{1}
{2}
{4}
0..1
0..1
0..1
«ElementPCDATA»
imscp_v1p1::title
«value» e-dtype: NMTOKEN = string
«Element»
imsld_v0p8::information
«Element»
imscp_v1p1::metadata
«Type»
imscp_v1p1::%itemmodel

Class Diagram: support activity

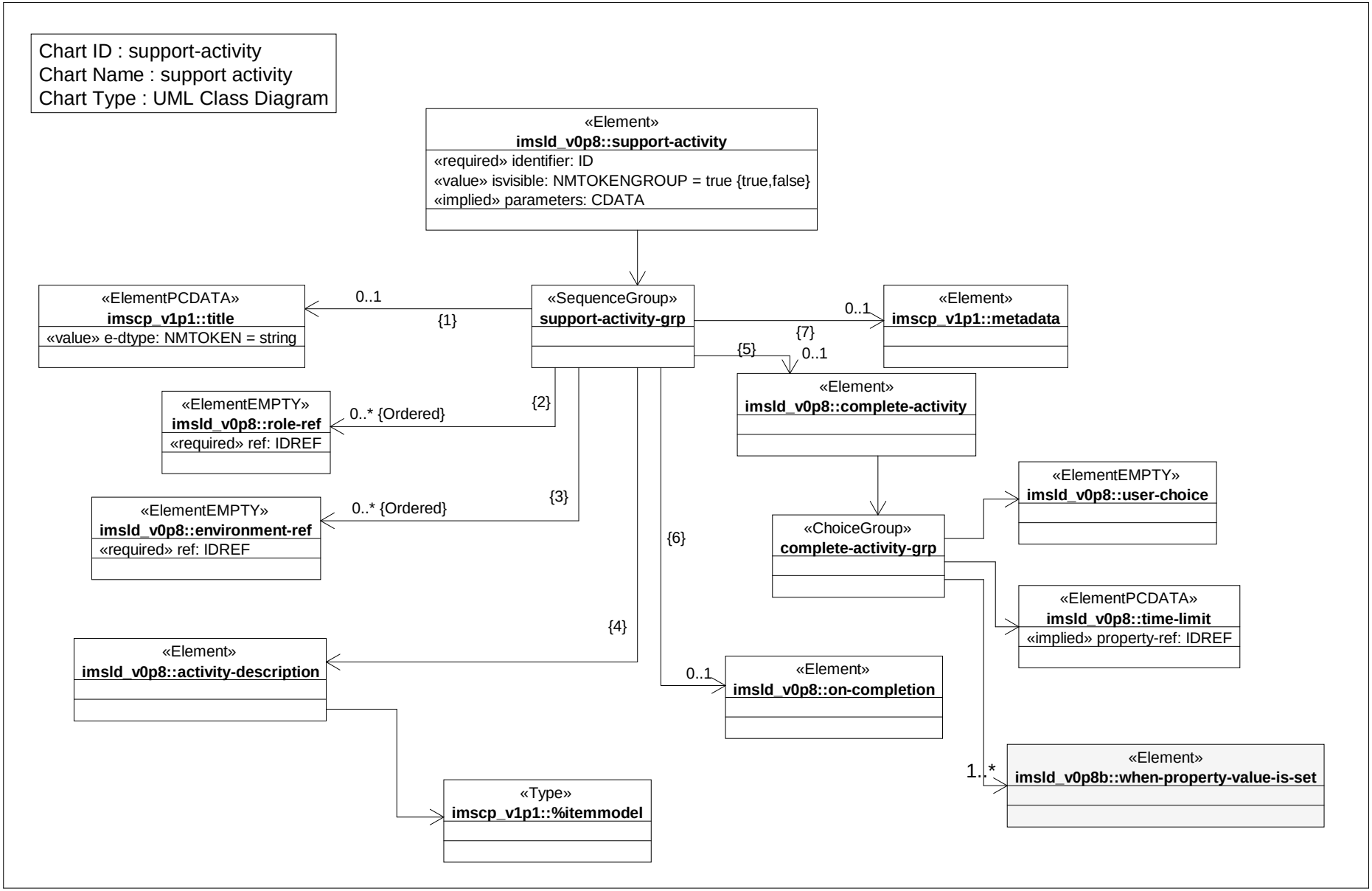
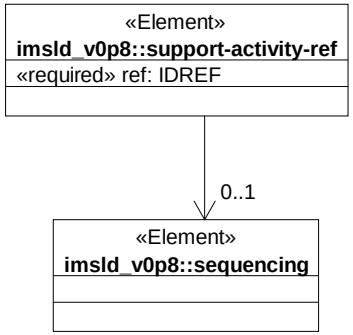


Chart ID : support-activity-ref
Chart Name : support activity ref
Chart Type : UML Class Diagram



Class Diagram: unit of learning

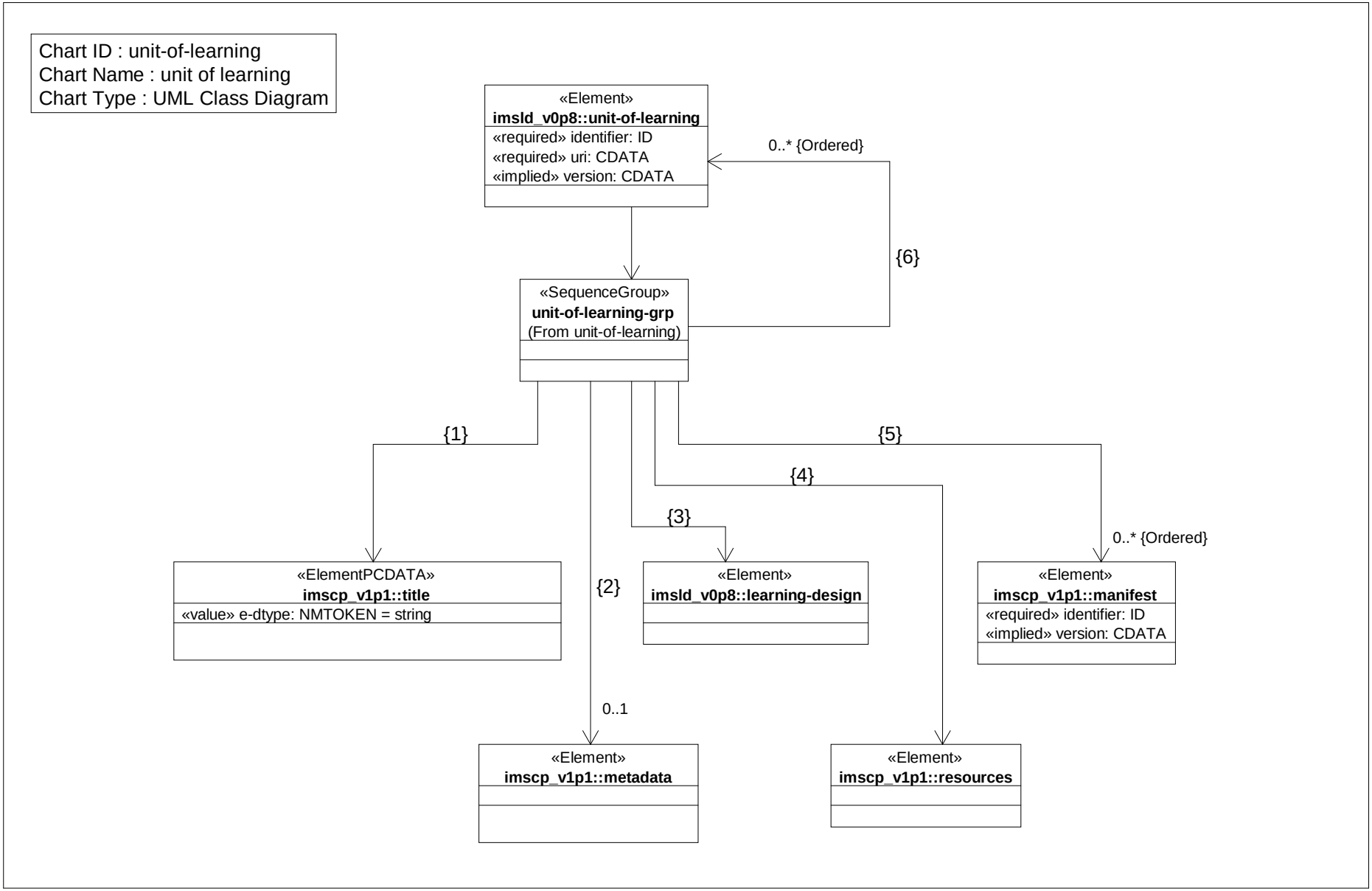


Chart ID : unit-of-learning-href
Chart Name : unit of learning href
Chart Type : UML Class Diagram

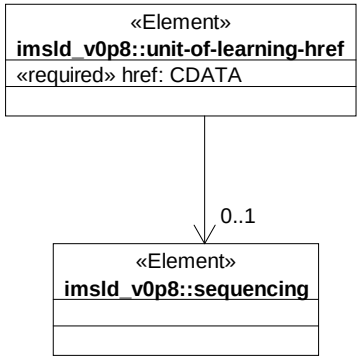


Chart ID : users-in-role
Chart Name : users in role
Chart Type : UML Class Diagram

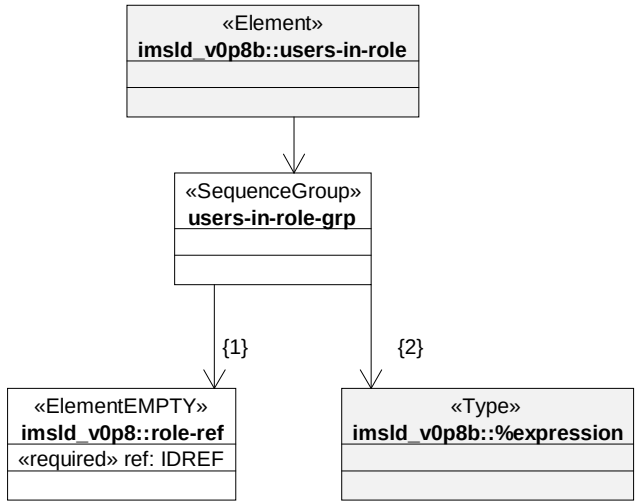
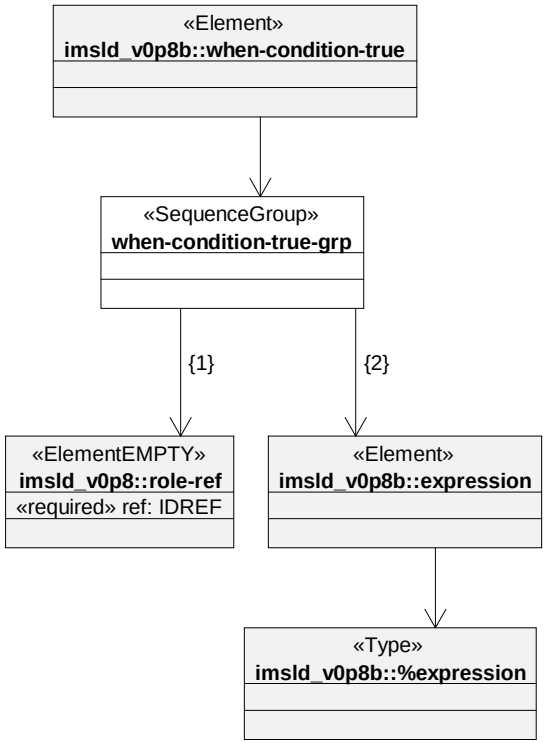
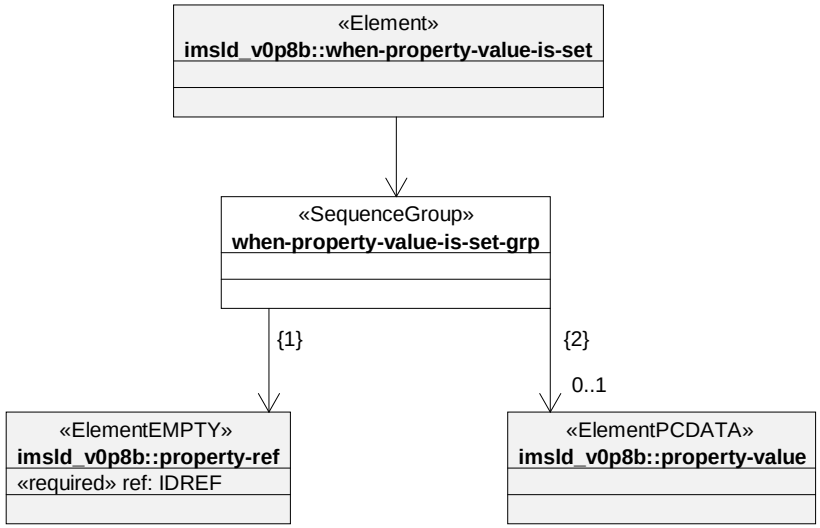


Chart ID : when-condition-true
Chart Name : when condition true
Chart Type : UML Class Diagram



Class Diagram: when propety value is set

Chart ID : when-property-value-is-set
Chart Name : when propety value is set
Chart Type : UML Class Diagram



Data dictionary

Element	Comment
expression	no comments
calculate	no comments
unit-of-learning	GENERAL INTRODUCTION The unit-of-learning (abbr. uol) is the root element of this dtd. A unit-of-learning is part of the unit-of-learning-package. A unit-of-learning-package consists of: 1) the unit-of-learning, which is an XML file named 'ims-unit-of-learning.xml'. The unit-of-learning consists of three subparts: metadata which describe the package; a learning-design, which describes the learning process and the resources which are refered to in the learning design. 2) the physical, digital files which are distributed with the package. Actors in the learning process, dealing with the interpreted units of learning are: . Learner (the people who are supported with the unit of learning in their learning process, the key role). . Staff (the people who support learners during the learning process, e.g. tutors, assessors). Beside these primary actors which are using the units of learning, there are also other actors who are dealing with the creation and management of the units of learning itself. These roles are: . Designer (designs an IMSLD learning-design, including imsld content or additions to webcontent to produce imsld content) . Author (authors webcontent and other non-imsld resources the learning-design is referring to) . Content manager (manages versions and learning objects, produces unit-of-learning-packages) . Legal roles (manages copy-rights) . Development manager (provides the authorization to develop or publish a unit of learning and the global URI's) All roles can be spit down further to an unlimited number of sub-roles and can have different names in different organizations. All actors expect from the e-learning system to get more effectiveness, more efficiency, more attractiveness and higher accessibility. All actors fill these aspects in from their own perspective. A learner wants more effective, efficient, attractive and accessible learning; a tutor wants to tutor in a more effective, efficient, attractive and accessible way, and so forth. The translation in general categories of requirements are as follows: Requirements for the model of units of learning (educational models): R1. It must describe units of learning in a formal way, so that automatic processing is possible (formalisation). R2. It must be able to describe learning designs in the unit of learning, that are based on different theories and models of learning and instruction (pedagogical flexibility). R3. It must explicitly express the semantic meaning of the different learning objects within the context of a learning design within the unit of learning (semantic typed learning objects). R4. It must be able to fully describe a unit of learning, including the learning design, the resources which it is dependent on and the metadata for all objects. The learning design should define all

Element	Comment
	the learning objects, the relationship between the learning objects and the workflow of all learners and staff members with the learning objects, regardless of whether these aspects are represented digital or non-digital (completeness). R5. It must describe a learning design of a unit of learning independent of its concrete content, so that the same learning design can be used in different units of learning (content independence). R6. It must describe the units of learning so that repeated execution is possible, with the same content and the same learning design (reproducibility). R7. It must be able to describe personalization aspects within units of learning, so that the content and activities within units of learning can be adapted based on the preferences, prior knowledge, educational needs and situational circumstances of users. In addition, control over the personalization must be able to be given, as desired, to the student, a staff member, the computer or the designer (personalization). R8. The notation of the (external) content components, where possible, must be medium neutral, so that it can be used in different publication formats, like the web, paper, e-books, mobile, etc. (medium neutrality). R9. It must provide an open and technology independent interface between educational content development and the different technical user agents which interpret the units of learning. Through this, investments in educational development will become resistant to technical changes and conversion problems (interoperability and sustainability). R10. It must fit in with current open standards and specifications (compatibility). R11. It must be possible to identify, isolate, decontextualize and exchange learning objects, and to re-use these in other contexts (reusability). R12. It must make it possible to produce, mutate, preserve, distribute and archive units of learning and all of its containing learning objects (life cycle). The identifier attribute is used to refer locally (i.e. inside this unit-of-learning-package) to the unit-of-learning. The identifier must be unique within this unit-of-learning-package. The URI attribute identifies the unit-of-learning globally and be absolute.The URI is only an identifier, e.g. http://ou.nl/coursecatalog/psychology/P345216. This URI doesn't necessarily point to a concrete resource.
title	*IMS CP* Inherited from IMS Content Packaging (including the attributes). So for a description and function of this element the IMS Content Packaging Specs (version 1.3). Definition: A short name given to the resource, suitable for rendering in user-agents.
metadata	*IMS CP* Inherited from IMS Content Packaging (including the attributes). So for a description and function of this element the IMS Content Packaging Specs

Element	Comment
	(version 1.3).
schema	no comments
schemaversion	no comments
learning-design	This element specifies the learning design (alias: instructional design) of the unit of learning. Any unit-of-learning can have zero or one learning-design. A unit-of-learning with zero learning designs are to support content-updates for an existing run of a unit-of-learning. For new runs a learning design is always expected to be present. Learning designs can be re-used in different units-of-learning, with different content. In an existing run of an unit-of-learning, the learning design may never be adapted for consistency reasons, however the href's of the resources may be adapted during the run as well as the content of the resources where the hrefs point to. For local resources which are available in the unit-of-learning package, this means that a new package may be loaded during the existing run, to update the content of the run (but not the logic, which is defined in the learning-design). Note on authoring templates: In order to implement design templates in the authoring environment, it is expected that a meta dtd will describe how a specific learning design may or may not be adapted (e.g. an author may include extra learning activities at certain points, but not on other points). RULE: In order to distinguish between references (IDREF) within the learning-design model and references to resources, the following rule applies: Attribute name 'ref' (IDREF) refers to an element with an identifier within the learning-design. Example: <act-ref ref=""/> refers to an act element within learning design. Elements with the 'identifierref' attribute, refer to resources. Example: <item identifierref=".."/> refers to a resource. The attribute name 'uri' is used for URI's which are ID's and the attribute 'href' is used for URI's which refer to uri's as ID's.
components	no comments
learning-objectives	Learning-objectives & prerequisites are specified as a series of zero or more learning-objective or prerequisite elements. It contains a standard organization of items, referring to resources or submanifests. There are at the moment two possible resource types: webcontent for descriptions of the learning objectives & prerequisites (e.g. an html page describing the learning objective or prerequisite for human readers). It can also point to a resource with an IMS Reusable Competency Definition Schema. There are two locations where learning-objectives & prerequisites are specified: - At the level of the unit-of-learning (in the root of learning-design) - At the level of learning-activities (within learning-activities). The first ones are a more general description; the second ones are more concrete. There are two types of learning-objectives: -1- human readable descriptions (the items point to text resources) -2- machine-readable specifications. These are addressed through the

Element	Comment
	href attribute of the resources pointed at. The learning-objectives schema's could be user-defined or fixed by an organization. In the latter case, the texts of the learning objectives are referred to (through href).
item	*IMS CP* Inherited from IMS Content Packaging (including the attributes). So for a description and function of this element the IMS Content Packaging Specs (version 1.3).
prerequisites	Learning-objectives & prerequisites are specified as a series of zero or more learning-objective or prerequisite elements. It contains a standard organization of items, referring to resources or submanifests. There are at the moment two possible resource types: webcontent for descriptions of the learning objectives & prerequisites (e.g. an html page describing the learning objective or prerequisite for human readers). It can also point to a resource with an IMS Reusable Competency Definition Schema. There are two locations where learning-objectives & prerequisites are specified: - At the level of the unit-of-learning (in the root of learning-design) - At the level of learning-activities (within learning-activities). The first ones are a more general description; the second ones are more concrete. There are two types of prerequisites: -1- human readable descriptions (the items point to text resources) -2- machine-readable specifications. These are addressed through the href attribute of the resources pointed at. The prerequisite schema's could be user-defined or fixed by an organization. In the latter case, the texts of the prerequisites are referred to (through href).
roles	Roles is a container for the two general roles: learner & staff. A href can be provided when referring to a global role (e.g. a role defined by an institute). This is obligatory when specifying a global role and connected globrole-properties. Global roles are specified with the href attribute. The rest of the declaration, like information, is local. It is not possible to declare global roles in a unit of learning. This is just an organizational issue and is nothing more or less than providing absolute URI's for roles. The attribute 'identifier' on roles can be used to refer to the whole group of all roles within the learning-design (learners and staff). The attributes: - min-persons specifies the minimum number of persons which must be bound to the role in order to start a run. - max-persons specifies the maximum number of persons which can be bound to the role in order to start the run. In both cases: when the attributes are empty, there are no restrictions. - match-persons. This attribute is used when there are several sub roles (e.g. chair, secretary, member).

Element	Comment
	Persons can be matched exclusively to the sub roles, meaning that a person who has the role of chair, may not be bound to one of the other roles at the same time. When it is not exclusive, persons may be bound to more than one sub role (this is the default situation). -create-new. This attribute indicates wether multiple occurences of this role are may be created during run-time. When the attribute has the value "not-allowed" than there is always one and only one instance of the role. If the value is "allowed" a mechanisme in the run-time is provided to create new instances of this role. If a new instance of a role is created, new istances for all sub-roles of that role are created as well. In every learning-design at least one learner role is specified. In institutional installations the role names are fixed. For instance in the OUNL role-identifiers for student is: 'student'. To address this, use the following declaration: This element specifies the roles distinguished in this unit of learning.
learner	In every learning design there is at least one learner-role. Learners can be 'nested', meaning that a role may be divided in sub roles. E.g. in an educational game you can distinguish: <learner ref="student"> <title>student</title> <learner ref="chair"> <title>chair</title> </learner> </learner> .. <resources> <resource identifier="student" type="role" href="http://ou.nl/role/student.role"/> </resources>
information	The information element specifies the resource(s) where the information can be found.
staff	In every learning design there are zero or more staff-members. Staff members can be 'nested', meaning that a role may be divided in sub roles. E.g. in an organization you can distinguish: <staff ref="tutor"><title>tutor</title> <staff ref="assessor"><title>assessor</title> </staff> </staff>
properties	Definition and declaration of new and existing properties. All properties used in the unit of learning are declared in this section. All properties can be addressed for property-operations (property-ref, view-property, view-property-group, etc.). See dossier specifications for more information about properties and operations.
loc-property	Local property, alias: run-property Local property. This property has the same value in a run for every user. The property is owned by the run of the unit-of-learning. The identifier can be used to refer to the property in this unit-of-learning-package. Operations can refer to this identifier to operate on the value.

Element	Comment
datatype	The following types are supported: boolean, integer, real, string, datetime, text, file, uri. These are also predefined in the attribute datatype. For extensions use the 'other' value and specify the content in the element self. boolean: represents binary logic, true or false (aliases: yes/no; 1/0). NB: just as any other data types, booleans can also have <no-value>. integer: is the standard mathematical concept of integer numbers, representing whole positive and negative numbers (including zero), ranging from: -9223372036854775898 to 922372036854775807 (alias: longinteger). real: standard mathematical concept representing arbitrary precision decimal numbers, and must be capable of handling a number to 18 decimal places at least. string: represents any legal character strings. The minimal maximum number of characters is 2000. datetime: This is a datetime specification of the form: CCYY-MM-DDThh:mm:ss CC is the century; YY is the year (year 0000 is prohibited); MM is the month; dd is the day. T is the date/time separator. hh are the hours; mm are the minutes; ss are the seconds. (see ISO 8601). There is also an optional timezone separator. Partial productions of the lexical expression are not allowed. duration: specifies an amount of time: the duration of an event in relative terms (e.g the duration given the start datetime of the run of a unit-of-learning. The format - also used in the W3C XML schema specification - is: PnYnMnDTnHnMnS where: P is the designator that must always be present. n is a variable where an integer is filled in. nY represents the number of years nM represents the number of month nD represents the number of days T is the date/time separator which must always be present when representing time. nH is the number of hours nM is the number of minutes nS is the number of seconds. Example: P2Y0M1DT20H10M55S Meaning that the duration is: 2 years and 0 month and 1 day and 20 hours and 10 minutes and 55 seconds. Limited forms of lexical production are also allowed E.g. a duration of 40 minutes is expressed: PT40M. A duration of 30 days is: P30D text: represents any legal character strings. The minimal maximum number of characters is 64000 (about 10 pages of A4 text). file: represents any binary file as datatype. The property stores this file. uri: represents an URI according to the IETF's RFC 2396 Note: according to the w3c only the word URI should be used in future and not URL or URN. (see: http://www.w3.org/TR/2001/NOTE-uri-clarification-20
initial-value	The initial value of the property is set to the value of this element when specified. When this value isn't specified the initial value is '<no value>'
restriction	Zero or more restrictions of different type may be set on the property-values, meaning that the property-value is valid when it is of the specified data type and its value is within the specified restriction rules.

Element	Comment
	Zero or more restrictions can be specified (these have the same format as specified in the W3C XML schema 1.0 specification) in the attribute: 'restriction-type'. However properties may not contain arrays (lists) of data, but can only contain a single value. So restrictions only apply to this single value. (Also the 'whitespace' restriction is not supported in imsld). The restriction types are: length: constraints the length of the property value of a textual datatype (string, text or uri) in terms of the of characters that it can have. minLength: contraints the minimum number of characters that a property of textual datatype can have for its value. maxLength: contraints the maximum number of units of length that a property of textual datatype can have for its value. enumeration: constraints the value of a property to a specific value (use for value alternative lists). maxInclusive:constraints the value of an ordered (integer, real, datetime) property to a specific inclusive upper bound. minInclusive: constraints the value of a ordered property to a specific inclusive lower bound. maxExclusive: constraints the value of a ordered property to a specific exclusive upper bound. minExclusive: constraints the value of a ordered property to a specific exclusive lower bound. totalDigits:constraints the value of a decimal property to a specific number of digits it must contain. fractionDigits: constraints the value of a decimal property to the maximum number of digits it may have after the decimal point. pattern:constraints the literals comprising the value of a property to a pattern defined by a regular expression.
locpers-property	Local personal property. This property can have a different value for every user in all the roles for a run of this unit-of-learning. The property is owned by the run of the unit-of-learning, specifying a value per user. The identifier can be used to refer to the property in this unit-of-learning-package. Operations can refer to this identifier to operate on the value.
locrole-property	alias: group-property Local role property. This property has the same value for every user in the specified role during the run of a unit-of-learning. The property is owned by the role in the run of the unit-of-learning. The identifier can be used to refer to the property in this unit-of-learning-package. Operations can refer to this identifier to operate on the value.
role-ref	role refers to the identifier of the resource of the role. The element can be used as an operand in an expression.
globpers-property	global personal property, alias: portfolio-property. This property can have a different value for every user, independent of the different runs of units of learning (its specifies the portfolio of the user).

Element	Comment
	The property is owned by the person. The identifier can be used to refer to the property in this unit-of-learning-package. Operations can refer to this identifier to operate on the value.
existing	Refers to a property already declared (e.g. in another unit-of-learning, or in the global dossier) to the knowledge of the author (see 'global-definition' what happens if the the author defines a new global property which in practice already exists). The property is refered to with href, specifying an absolute URI. NB: when validating this unit-of-learning, the URI doesn't have to be present. The declaration of the URI by an external unit-of-learning can happen at any time. So this is only under the control of the author.
global-definition	NB: This element is only included for implementations which lack a tool to define global properties: it is an optional construction in IMSLD and can be removed in systems which only refer to existing (elsewhere) declared global properties. To force consistency, the next rule applies: RULE: Once a global property has been defined in whatever context, it can never be changed! This is also true for re-publications of the same unit-of-learning. So the definition is only used when the URI (href) doesn't exist yet. Otherwise it is ignored. The URI must be an identifier which identifies the global property global unique. It must be an absolute URI. When the URI is an URL, the URI doesn't need to point to the property location, but can be interpreted as an identifier.
glob-property	A global property is a global unique property, which stores one value, independent of user, units-of-learning and role. The identifier can be used to refer to the property in this unit-of-learning-package. Operations can refer to this identifier to operate on the value.
property-group	A definition of a group of properties which belong together (and are editted in e.g. a form). It can only contain properties of the same type. The identifier can be used to refer to the property-group in this unit-of-learning-package. Operations can refer to this identifier to operate on the value.
property-ref	Refers to a property This can be a property of all kind: - local property - global property - local personal property - local role property - global personal property - local role property The ref refers to the property declaration in the learning-design. The element can be used as an operand in an calculation or expression..
property-group-ref	no comments

Element	Comment
activities	This element is a container for activity definitions,including 'activity-structure'.
learning-activity	Definition: a learning activity is an instruction for a learner to perform an task within an environment. Examples: solve a problem, study a book, discuss a topic with peers, write a dissertation, etcetera.
environment-ref	This element refers to an environment in this package.
activity-description	Alias: task. The activity-description is the actual cue given to the user (rendered in the user-interface) to describe the activity to be performed by the user. In most cases the activity-description is a text (of type webcontent). In other cases it can be an audio-file (webcontent), a video file or any other cue to the user. Activity-descriptions also define the environment for the activity. Every noun mentioned in the description refers to a resource in the environment. It is up to the author to have a strict representation of the nouns in the environment or a more open one (leaving nouns implicit).
complete-activity	This container has elements to specify when an activity is completed. When this element doesn't occur, the activity is set to 'completed'.
user-choice	This element is used in the completed element of activities and specifies that the user may decide himself when the activity is completed. This means that a control must be available in the user-interface to set the activity status to 'completed'. A user can do this once (no undo). When he/she selected the activity to be completed, than this activity stays completed in the run.
time-limit	The time limit specifies that it is completed when a certain amount of time have been passed, relative to the start of the run of the current unit of learning. The data type time is expressed in the 'duration' format which is also used in the W3C XML schema specification. The format is: PnYnMnDTnHnMnS where: P is the designator that must always be present. n is a variable where an integer is filled in. nY represents the number of years nM represents the number of month nD represents the number of days T is the date/time separator nH is the number of hours nM is the number of minutes nS is the number of seconds. Example: P2Y0M1DT20H10M55S Meaning that the duration is: 2 years and 0 month and 1 day and 20 hours and 10 minutes and 55 seconds. Limited forms of lexical production are also allowed E.g. a duration of 40 minutes is expressed: PT40M. A duration of 30 days is: P30D The time is always counted relative to the time when the run of the unit-of-learning has been started (see element: 'time-unit-of-learning-started'. Authors have to take care that the time limits set on role-parts, acts and plays are logical. In runtime the time limit of the play overrules the time limit on act and that one on the role-parts.

Element	Comment
	The time-limit may be specified in a property (property-ref attribute). (of type loc-property, datatype=string, to be declared by the author). In that case an author may set controls (set-property) on this property for users to control the value of the property. When a property-ref is specified, content in the element is ignored: the property overrules.
when-property-value-is-set	Simple expression, containing two child elements: a property and an optional property-value. The condition evaluates to true when: 1) the property is set to the specified property-value; 2) the property is not NULL and the propery-value is omitted;
property-value	The element can be used as an operand in an calculation or expression. This element specifies the value a property is set or compared to. This depends on the context. For instance within an If statement the property is compared to the value. In a change-property-value context, the property is set to this value. Depending on the property-type this value is of type PCDATA or langstring. Property-value's may be calculated from the values of other properties. It is also possible to take over the property value of another property (with property-ref).
langstring	This is identical to the XHTML <p> element. The binding comes from the IMS METADATA. The attribute xml:lang may be added to all elements according to the W3C specifications. It is specifically needed on this element, but deleted here in order to support XML schema notation. When using dtd's the xml:lang attribute can be added to the different elements.
calculate	This is the container for the elements to perform calculations. This container is also used in expressions.
is-member-of-role	no comments
is	no comments
is-not	no comments
and	no comments
or	no comments
sum	no comments
subtract	no comments
multiply	no comments
divide	no comments
greater-than	no comments
less-than	no comments
users-in-role	no comments
no-value	no comments
time-unit-of-learning-started	no comments
datetime-activity-started	no comments

Element	Comment
current-datetime	no comments
complete	no comments
learning-activity-ref	Refers to a learning-activity. The element can be used as an operand in an calculation or expression.
sequencing	This is where IMS Simple Sequencing Schema elements are namespaced into the Learning Design. It occurs only in activity-structures: on activity-lists as well as on single activities.
support-activity-ref	Refers to a support-activity. The element can be used as an operand in an calculation or expression.
unit-of-learning-href	The element can be used as an operand in an calculation or expression. This element refers to the resource of a unit-of-learning (uol). That can be the resource of this current unit-of-learning, the resource of a unit-of-learning present in the package or a resource pointing to a unit-of-learning outside of the package (absolute URI). Formally called: unit-of-study-ref. Because -ref extensions point to elements within learning-design, this name has changed.
activity-structure-ref	Reference to an activity-structure.
role-part-ref	no comments
act-ref	Refers to an act (in method/play/act).
play-ref	no comments
not	no comments
on-completion	When an activity is completed, the optional actions contained in this element are excecuted. In level A it contains only one element. The container is for extensions of level B.
feedback-description	The element points to a resource of type webcontent, where the feedback description can be found. After completion this text becomes visible.
change-property-value	This element is used to change values of properties after an event (e.g. completion of something). E.g. When the activity is completed, a property value may be changed to reflect this fact. In the dossier also an automated record of completed activities is kept, so it isn't necessary to record the completion as such, but to register (or change) other things.
notification	A notifications happens after an event which is known by the runtime environment. Such an event can be e.g.: the completion of an activity, an expression evaluates to true, or a property value is set. The notification makes a new learning or support activity active for a role or sends a message. This notification is of the highest priority, meaning that an otherwise invisible item will be made visible and accessible to the user. Depending on the implementation an email message can be send to the user, notifying that a new activity has arrived (with a link to that activity in the message). For the mail header the subject field can be filled in to a specific value (otherwise a standard message will be send). A notification can be inserted in external vocabularies (after an event like set-

Element	Comment
	property), however, then the content must be provided in the package (because it contains references to identifiers in the package). When the identifier cannot be resolved the notification is ignored (but doesn't stop the xhtml content to be presented).
email-data	This element refers to the property resources where the relevant e-mail data can be found for the connected role. This is used for send-mail purposes (as a service in the environment, or in notifications). This element has two attributes: - email-property-ref: this attribute contains a reference to the property containing the email address of the users being notified - username-property-ref: this optional attribute contains a reference to the property containing the user name of the users being notified Both properties (email, username) should be available for all persons assigned to the role and the sending party.
subject	It specifies the subject of a notification, to be presented to the notified actor when the notification is activated. E.g. in the mail-header (subject field).
support-activity	Definition: a support activity is an instruction for a staff or learner role, to support a role (learner/staff) or users in a role in performing the activities. The support role uses an environment in order to perform the support activity. Most of the times, support activities are performed by staff members (e.g. tutors) to support learners. In some pedagogical models however learners can support learners (peer to peer teaching). It is also possible that staff members support staff members. When the optional role-ref element is set, it is expected that the support actvity will act for every single user in the specified role(s). That is: the same support activity is repeated for every user in the role(s). When the role-ref is not available, the support activity is a single activity (like the learning-activity).
activity-structure	An activity-structure groups activities. When the attribute 'number-to-select' is set, the activity-structure is completed when the total number of activities selected are completed. The number-to-select must be similar or smaller than the number of activities (including unit-of-learning's) which are at the immediate child level. When the number-to-select isn't set, the activity-structure is completed when all the activities in the structure are completed. The attribute 'sort' determines the sort-order in relation to the visibility. Default the order in which activities are made visible is in the order specified in the activity-structure. When the value is set to 'visibility-order', activities are presented in the order they where made visible (this imitates a kind of inbox: new activities come available over time). The tree is handled depth first (and not breadth first). If there is a sequencing

Element	Comment
	element it overrules the attributes: structure-type and number-to-select.
environments	no comments
environment	no comments
learning-object	Learning objects can be of the following types: - knowledge-object - tool-object - QTI
service	no comments
send-mail	This service is used to send mail to users in roles (with mailaddress in property for level b/c).
conference	The elements participant, observer, conference-manager, moderator facilitate the setting of the user rights in the conferences. It depends on the implementation how this is managed: 1. when the conference system is an integral part of the runtime it is expected to be set automatically; 2. when the conference is external the user-rights can be set manually by the conference manager. The conference managers, must be able to get a list from the runtime agent about which conferences of what type, for what users with what rights must be set. 3. the latter can also be implemented by a developing a legacy interface to the rights management system of the conferencing system. In all instances the runtime system must be able to provide this information in a structured way. The item element refers to the resource where the conferencing system is to be found or identified. External conferencing systems can be of any kind accessible through the internet (resource type is webcontent). Examples: netmeeting, placeware (synchronous), first-class, lotus notes, news groups (asynchronous). An announcement object sets the rights: creator of announcement = participant. Reader of announcements = observer.
participant	Specifies who the participants are in the conference. Participants can read (listen/see) the information, and can contribute to the conference. This element has an effect on setting the user rights in the conference. At least one role must be specified to identify the participants in the conference.
observer	Specifies who the observers are in the conference. Observers have only reading rights; they may not contribute. This element has an effect on setting the user rights in the conference.
conference-manager	The conference manager is allowed to create new sub conferences and delete conferences he/she created. The new conferences are children of the existing base conference. The conference manager may not delete the base conference. It is deleted by system management when deleting the information of the (completed) run of the unit of learning. The conference manager has all the rights of observer, participant.

Element	Comment
moderator	Specifies who the moderators are in the conference. Moderators are persons who have the right to control and change the contributions of participants before they are made visible to other participants or observers. When a moderator is specified it means that participants may not contribute directly to the conference, but via the moderator. The moderator can reject, adapt or accept a proposed contribution of a participant. In all cases the contributor is notified of the judgement of the moderator. When there are more users in the role connected to the moderator, all have the same rights, but always the first one who did the job decides. This element has an effect on the setting of the user rights in the conference.
monitor	no comments
self	no comments
index-search	index-search
index	This element is a wrapper for indexing aspects, used to set up a search service. The index is made in the background (not visible to users). The visibility is determined with the search element. The functionality of the index is dependent on the search element: - when search is free-text-search, then the index is made on the resource pointed at in the index (i.e. the underlying html texts). - when search is index-with/without-reference, than only an index is made of the elements which share the same class, including underlying items. This has the form of a table of content.
index-class	This element selects the class to make the index on. Only one class item per element may be provided. Example: <index-class index-class="problemdescription"/> makes an index on all objects in the design which have one of the strings in the class attribute assigned to "problemdescription".
index-element	This element selects the element to make the index on. The index attribute specifies the element to index-on (only one reference per index-element). This indexing only makes sense when there is a structure to index on, or underlying text to index for free-text-search.
index-type-of-element	In this element the type of element to index on is entered. Only one element name per index-type-element occurance. The element names much match the element names uses in the IMSLD schema. E.g.: <index-type-of-element>learning-activity</index-type-of-element>
search	This element specifies how a user can access the indexed entities. There are three possibilities: 1. the user gets a free text search dialog, where he can search the index in a free text format (this also means that the index has to be build for free text retrieval). The syntax for free text retrieval is implementation dependent, e.g. the format found in search engines like Google or Altavista. 2. the user is presented a text index (table of content) with (hyper-)linked (or on other media e.g. page numbers) references to the source. 3. the user is presented a text index (table of content) without (hyper-)linked references. This provides e.g. information about the structure of the unit of learning.
method	The method is a container for the definition of the dynamics of the learning process.

Element	Comment
	It consists of a play (which could be interpreted as the runscript for the unit of learning) and a statement for the completion of the unit of learning.
play	The play is the root element when interpreting the learning design. It represents the flow of activities during the learning process (the 'workflow' or better: the 'learningflow'). The play is modelled according to the metaphor of an theatrical performance. A play consists of a series of acts and an act consists of a series of role-part's. There is always at least one play in every learning design (and every unit-of-learning). In runtime the play is interpreted to show and hide activities, (other)units-of-learning, environments and resources to the users. When there is more than one play, these are interpreted concurrently and independent of each other. The same user can see the results of more than one play in the user-interface. Practical experience have shown that a lot of designs use multiple plays, to represent the flows of activities per role, e.g. a play for the learners and a play for staff. However this can only be done when the activities are independent of each other.
act	A play is modelled according to the metaphor of an theatrical performance. A play consists of a series of acts and an act consists of a series of role-part's. An act represents a series of concurrent role-part's. There is at least one act in a play. When there is more then one act in a play these are interpreted in a sequenced manner: from first act to last act. Only one act in a play is the active act at any moment in time, starting with the first. When the first act is completed, the second act is made the active act. The first is still visible and accessible, but in the interface it is made clear that it this is only looking in history. When the second act is completed, the third act is made active, etc. Acts which are sequenced in row after the current active act are never visible. Conditions cannot overrule this, meaning that the act is of higher priority than conditions.
role-part	A play is modelled according to the metaphor of an theatrical performance. A play consists of a series of acts and an act consists of a series of role-part's. A role-part relates exactly one role to exactly one type of activity (including the performance of another unit-of-learning and activity-structures). Role-parts within one act, are performed concurrently. The definition of a role-part differs here somewhat from real role-part's in a theatrical play (here the metaphor is not completely correct, like all metaphors). When an activity or item attribute isvisible is set to 'false', the link

Element	Comment
	in the activity-tree may be made visible when the role-part sets the activity for a role (implementation dependent) , but the content isn't accessible. .
complete-act	This container has elements to specify when an act is completed. When this element doesn't occur, the completed is set to 'unlimited'.
when-role-part-completed	This element states that an act is completed when the referenced role-part(s) is(are) completed. More than one role-part can be selected, meaning that all the referenced role-parts must be completed before the act is completed. NB: all role-parts referrences must be in specified in the current act!
when-condition-true	Simple expression for a condition. This condition applies to all the individual users mentioned in the containing role-ref! When the contained expression is true for all users in the specified roles, this condition is true.
expression	no comments
complete-play	This container has elements to specify when a play is completed. When this element doesn't occur, the completed is set to 'unlimited'.
when-last-act-completed	This element states that a play is completed when the last act is completed.
complete-unit-of-learning	This container has elements to specify when a unit-of-learning is completed. When this element doesn't occur, the completed is set to 'unlimited'.
when-play-completed	This element states that an unit-of-learning is completed when the referenced play('s) is (are) completed. More than one play can be selected, meaning that all the referenced play's must be completed before the unit-of-learning is completed. When a unit-of-learning is completed this should be made aware in the runtime environment to the managers of the system.
conditions	Conditions are used to personalize the presentation of the unit-of-learning. All conditions are pre-conditions and must be evaluated: - when entering the unit of learning (new session); - every time when the value of a property has been changed. This applies only to the following properties: a) properties where the person has access to in the context of the unit of learning, and b) the property has to be evaluated in one of the expressions in the unit of learning. These properties include properties, which are available in the expression, but are set automatically (e.g. time-unit-of-learning-started). An action is performed (fired) according to the success (true) or failure (false) of the condition. The action is to show, hide, change-property-value or notify a role. The show and hide actions set the visibility attribute (isvisible) of different objects: activities, environments, items, play's, activity-structures, units-of-learning and different classes of objects (set with the 'class' attribute).
if	If refers to an expression-schema which evaluation results in the value: true or false. The expression-schema is kept as a seperate schema with a different namespace (e.g. http://imsglobal.org/xsd/imsld_expr_v1_p0. In theory different expression schema's may be used.

Element	Comment
	However it is preferred to use the expression (and calculation) schema provided with IMSLD). In authoring environments it is expected to be integrated at this place of the schema. When the expression resolves to 'true', the 'then' rule fires. When it resolves to 'false' the 'else' rule fires when it is present (otherwise nothing happens in this rule).
then	no comments
show	This container contains what has to be shown when the condition (if) is true. This effects the 'isvisible' status of the entity (set to true).
class	Inherited from HTML (related to CSS). It is used to identify classes of common objects in order to manipulate them once. A class attribute contains a CDATA string. Just as in HTML more than one class may be specified in one CDATA string, each separated with a blank space. The priority order for classes is the same as specified in the CSS specification (see www.w3.org/style/css). Note that the classes can be used for style sheet like functions (e.g. set visibility), but they can also have a semantic classification purpose (just as in HTML) not connected to style sheets or automated processing at all.
item-ref	Refers to the identifier of an item in the design context.
hide	The general rule is: activities that are shown to a user can never be hidden again. In the show-model however they are present! This container contains what must be made hidden when the condition (if) is true. This affects the 'isvisible' status of the entity (set to false).
else	no comments
resources	IMSCP
resource	no comments
file	no comments
dependency	no comments
manifest	IMS Content Packaging. Items can refer to resources or submanifests.
organization	no comments
set-property	With this element a specified property-value may be set by the user. It works outside of the context of a textline (e.g. outside <p>). The view attribute sets whether the value or the title+value should be delivered. The user gets a control in the user-interface to set the value of the property. The type of control is dependent on the property datatype and the restrictions. In the control the current value is shown and the datatype and restrictions are made explicit so that the user knows exactly what values are valid and which are not. This allows for client-side checking of the input (dependent on implementation this may also be dealt with at

Element	Comment
	the serverside). The element refers to the property URI or identifier with a ref or href. In order to avoid confusions it is good practice to include the imsldcontent with property-opertations in the unit-of-learning-package. In case of personal properties, the property value of the user himself is set. Only in the context of a monitor object (with the role-ref specified) the values of all the users in the specified role can be set. The attribute max-transactions represents the number of times a property may set by a user. Technical (upload) errors do not count as a trial, but only successful transactions. When the attribute sn't specified, the number of attempts is set to unlimited. The attribute transaction-type: future extension, e.g. for secure transactions
set-property-group	With this element the values of the properties contained in a specified property-group may be set by the user. It works outside of the context of a textline (e.g. outside <p>). The view attribute determines whether the titles of the containing values should be shown. The group-title is always shown. The user gets a control in the user-interface to set the value of the property-group. The type of control per property is dependent on the property datatype and the restrictions set on the property. In the control the current value of the properties are shown and the datatype and restrictions are made explicit so that the user knows exactly what values are valid and which are not. This allows for client-side checking of the input (dependent on implementation this may also be dealt with at the serverside). All values of all properties in the group are set at by the user before updating. The transaction is always counted for the group of properties, not for single properties. The element refers to the property-group identifier or URI with a ref or href. In order to avoid confusions it is good practice to include the imsldcontent with property-opertations in the unit-of-learning-package. In case of personal properties, the property values of the user himself is set. Only in the context of a monitor object (with the role-ref specified) the values of all the users in the specified role can be set. The attribute max-transactions represents the number of times a property may set by a user. Technical (upload) errors do not count as a trial, but only successful transactions. When the attribute sn't specified, the number of attempts is set to unlimited.
view-property	With this element a specified property-value may be viewed. It works outside of the context of a textline (e.g. outside the context of a <p> element. The view attribute sets whether the value or the title+value should be delivered. It refers to the property with an ref or href. In order to avoid confusions it is good practice to include the imsldcontent

Element	Comment
	with property-opertations in the unit-of-learning-package. In case of personal properties, the property value of the user himself is returned. Only in the context of a monitor object (with the role-ref specified) the values of all the users in the specified role(s) are returned.
view-property-group	With this element the values of the properties in a specified property-group may be viewed. It works outside of the context of a textline (e.g. outside <p>). The view attribute determines whether the titles of the containing values should be shown. The group-title is always shown. It refers to the property-group identifier or URI with a ref or href. In order to avoid confusions it is good practice to include the imsldcontent with property-opertations in the unit-of-learning-package. In case of personal properties, the property value of the user himself is returned. Only in the context of a monitor object (with the role-ref specified) the values of all the users in the specified role(s) are returned.

DTD generation XSLT script

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                              xmlns:msxsl="urn:schemas-microsoft-com:xslt"
                              xmlns:user="http://mycompany.com/mynamespace">

   <xsl:output method="text" version="1.0" encoding="UTF-8" indent="yes"/>


  <!-- scripting extension is needed to maintain the stack of processed entities. This
       isn't possible when using standard XSLT 1.0. Alternatively all entities need to
       be sorted by hand after generating.
  -->
  <msxsl:script language="JScript" implements-prefix="user">

     var entities="";

     function add_entity (entity) {
       entities = entities + entity;
       return entities;
     }

     function get_entities() {
        return entities;
     }
  </msxsl:script>



 <!-- starting point: start generating the entitities and the elements -->
 <xsl:template  match="/">
  <xsl:text disable-output-escaping="yes">&lt;!-- Generated by UML DTD converter</xsl:text>
  <xsl:text disable-output-escaping="yes">&#13;&#10;     (C) 2002 Open Univeriteit Nederland --
>&#13;&#10;</xsl:text>
```

```
  <xsl:call-template name="entities"/>
  <xsl:call-template name="elements"/>
 </xsl:template>


 <!-- the is one of the three main templates. It loops over all entities and create the DTD declarations for it -->

 <xsl:template name="entities">
  <xsl:text disable-output-escaping="yes">&#13;&#10;&#13;&#10;&lt;!-- entity declarations -->&#13;&#10;</xsl:text>

  <!-- call recursive funtion that will deal with dependencies of entities for one level -->
  <xsl:for-each select="//classes/class/stereotype[child::text()='Type']">
    <xsl:call-template name="process-entity">
      <xsl:with-param name="entity" select="parent::class/@name"/>
    </xsl:call-template>
  </xsl:for-each>
 </xsl:template>

 <!-- this template will be called recursively when an entity is encountered as
      part on an entity -->
 <xsl:template name="process-entity">
   <xsl:param name="entity"/>
   <xsl:variable name="processed-entities" select="user:get_entities()"/>

   <!-- check if we have processed this entity already -->
   <xsl:if test="not (contains($processed-entities,concat( '(',$entity,')' )))">
     <!-- appearently not -->
     <xsl:variable name="grpname" select="//association[child::startsin/@type='Class' and
child::startsin/@name=$entity]/endsin/@name"/>

     <xsl:for-each select="//association[child::startsin/@name=$grpname]">
       <xsl:variable name="classname" select="endsin/@name"/>
       <xsl:variable name="stereotype" select="//class[@name=$classname]/stereotype"/>

       <!-- check if this linked element is an entity -->
```

```
      <xsl:if test="$stereotype='Type'">
        <!-- class this function recursively -->
        <xsl:call-template name="process-entity">
          <xsl:with-param name="entity" select="$classname"/>
        </xsl:call-template>
      </xsl:if>

    </xsl:for-each>

    <!-- we have to create an dummy variable to cal the script -->
    <xsl:variable name="dummy" select="user:add_entity(concat ('(',$entity,')'))"/>

    <!-- write the entity declaration -->
    <xsl:text>&#13;&#10;&#9;&lt;!ENTITY % </xsl:text>
    <xsl:value-of select="substring-after($entity,'%')"/>
    <xsl:text> "</xsl:text>
    <xsl:call-template name="modelgrp">
      <xsl:with-param name="class" select="$entity"/>
      <xsl:with-param name="processed_groups" select="string('')"/>
    </xsl:call-template>
    <xsl:text>" </xsl:text>
    <xsl:text disable-output-escaping="yes">></xsl:text>
  </xsl:if>

</xsl:template>


<!-- the is one of the three main templates. It loops over all elements and creates the DTD declarations for it -->

<xsl:template name="elements">
 <!-- write complex elements (branches) -->
 <xsl:text disable-output-escaping="yes">&#13;&#10;&#13;&#10;&lt;!-- element delcarations -->&#13;&#10;</xsl:text>
 <xsl:for-each select="//classes/class/stereotype[child::text()='Element']">
   <xsl:text disable-output-escaping="yes">&#13;&#10;&#9;&lt;!ELEMENT </xsl:text>
   <xsl:value-of select="parent::class/@name"/>
   <xsl:text> </xsl:text>
```

```
    <xsl:call-template name="modelgrp">
      <xsl:with-param name="class" select="parent::class/@name"/>
      <xsl:with-param name="processed_groups" select="string('')"/>
    </xsl:call-template>
    <xsl:text> </xsl:text>
    <xsl:text disable-output-escaping="yes">></xsl:text>
    <xsl:call-template name="attributes"><xsl:with-param name="classname" select="parent::class/@name"/></xsl:call-
template>
    <xsl:text>&#13;&#10;</xsl:text>
  </xsl:for-each>

  <!-- write simple elements (leaves) #PCDATA-->
  <xsl:for-each select="//classes/class/stereotype[child::text()='ElementPCDATA']">
    <xsl:variable name="classname" select="parent::class/@name"/>
    <xsl:variable name="association" select="//association[child::startsin/@type='Class' and
child::startsin/@name=$classname]"/>
    <xsl:variable name="linked-classname" select="$association/endsin/@name"/>

    <xsl:text disable-output-escaping="yes">&#13;&#10;&#9;&lt;!ELEMENT </xsl:text>
    <xsl:value-of select="parent::class/@name"/>
    <xsl:text> </xsl:text>

    <xsl:choose>
      <!-- check if we are dealing with a mixed content element -->
      <xsl:when test="not($linked-classname)">
        <!-- this is a simpel #PCDATA element -->
        <xsl:text>(#PCDATA)</xsl:text>
      </xsl:when>

      <xsl:otherwise>
        <!-- this is a mixed content element, so not so simple after all -->
        <xsl:call-template name="group">
          <xsl:with-param name="class" select="$linked-classname"/>
          <xsl:with-param name="multiplicity" select="$association/endrole/multiplicity"/>
          <xsl:with-param name="grouptype_char" select="string(' | ')"/>
          <xsl:with-param name="mixed-content" select="string('true')"/>
```

```
          <xsl:with-param name="processed_groups" select="string('')"/>
        </xsl:call-template>
       </xsl:otherwise>
     </xsl:choose>

     <xsl:text> ></xsl:text>
     <xsl:call-template name="attributes"><xsl:with-param name="classname" select="parent::class/@name"/></xsl:call-
template>
     <xsl:text>&#13;&#10;</xsl:text>


   </xsl:for-each>

   <!-- write simple elements (leaves) #EMPTY-->
   <xsl:for-each select="//classes/class/stereotype[child::text()='ElementEMPTY']">
     <xsl:text disable-output-escaping="yes">&#13;&#10;&#9;&lt;!ELEMENT </xsl:text>
     <xsl:value-of select="parent::class/@name"/>
     <xsl:text disable-output-escaping="yes"> EMPTY ></xsl:text>
     <xsl:call-template name="attributes"><xsl:with-param name="classname" select="parent::class/@name"/></xsl:call-
template>
     <xsl:text>&#13;&#10;</xsl:text>
   </xsl:for-each>

   <!-- write simple elements (leaves) #ANY-->
   <xsl:for-each select="//classes/class/stereotype[child::text()='ElementANY']">
     <xsl:text disable-output-escaping="yes">&#13;&#10;&#9;&lt;!ELEMENT </xsl:text>
     <xsl:value-of select="parent::class/@name"/>
     <xsl:text disable-output-escaping="yes"> ANY ></xsl:text>
     <xsl:call-template name="attributes"><xsl:with-param name="classname" select="parent::class/@name"/></xsl:call-
template>
     <xsl:text>&#13;&#10;</xsl:text>
   </xsl:for-each>
  </xsl:template>
```

```
<!-- the is one of the three main templates. It loops over all attribute and creates the DTD declarations for it --
>
<xsl:template name="attributes">
  <xsl:param name="classname"/>

  <xsl:variable name="class" select="//classes/class[@name=$classname]/attributes"/>

  <xsl:if test="$class">
    <xsl:text disable-output-escaping="yes">&#13;&#10;&#9;&lt;!ATTLIST </xsl:text>
    <xsl:value-of select="$classname"/>

    <xsl:for-each select="$class/attribute">
      <xsl:text>&#13;&#10;&#9;&#9;&#9;</xsl:text><xsl:value-of select="@name"/><xsl:text>  </xsl:text>

      <xsl:choose>
        <xsl:when test="type='CDATA' or type='ID' or type='IDREF'">
          <xsl:value-of select="type"/>
          <xsl:text>  #</xsl:text>
          <xsl:value-of select="translate (stereotype,'abcdefghijklmnopqrstuvwxyz','ABCDEFGHIJKLMNOPQRSTUVWXYZ')"/>
          <xsl:if test="initialvalue">
            <xsl:text>  "</xsl:text>
            <xsl:value-of select="initialvalue"/>
            <xsl:text>"</xsl:text>
          </xsl:if>
        </xsl:when>

        <xsl:when test="type='NMTOKEN'">
          <xsl:value-of select="type"/>

          <xsl:if test="initialvalue">
            <xsl:text>  "</xsl:text>
            <xsl:value-of select="initialvalue"/>
            <xsl:text>"</xsl:text>
          </xsl:if>
        </xsl:when>
```

```
        <xsl:when test="type='NMTOKENGROUP'">
          <xsl:text>  (</xsl:text>
          <xsl:value-of select="translate(constraint,',','|')"/>
          <xsl:text>) </xsl:text>

          <xsl:choose>
            <xsl:when test="initialvalue">
              <xsl:text>  "</xsl:text>
              <xsl:value-of select="initialvalue"/>
              <xsl:text>"</xsl:text>
            </xsl:when>

            <xsl:otherwise>
              <xsl:text>  #</xsl:text>
              <xsl:value-of select="translate
(stereotype,'abcdefghijklmnopqrstuvwxyz','ABCDEFGHIJKLMNOPQRSTUVWXYZ')"/>
            </xsl:otherwise>

          </xsl:choose>

        </xsl:when>
      </xsl:choose>
    </xsl:for-each>

    <xsl:text disable-output-escaping="yes"> ></xsl:text>

  </xsl:if>
</xsl:template>
```

```
 <!-- we are dealing with a branch, so we have to determine which type of branch -->
 <xsl:template name="modelgrp">
   <xsl:param name="class"/>
   <xsl:param name="processed_groups"/>

   <xsl:variable name="grpclass" select="//association[child::startsin/@type='Class' and
child::startsin/@name=$class]"/>
   <xsl:variable name="stereotype" select="//classes/class[@name=$grpclass/endsin/@name]/stereotype"/>

   <xsl:choose>
     <xsl:when test="$stereotype='SequenceGroup'">
       <xsl:call-template name="group">
         <!-- class name of the associated class -->
         <xsl:with-param name="class" select="$grpclass/endsin/@name"/>
         <!-- multiplicity of the associated class -->
         <xsl:with-param name="multiplicity" select="$grpclass/endrole/multiplicity"/>
         <!-- pass the processed groups to avoid recursion -->
         <xsl:with-param name="processed_groups" select="$processed_groups"/>
         <!-- grouptype_char defines the separator between elements of a group and therefore the type of the group --
>
         <xsl:with-param name="grouptype_char" select="string(',')"/>
        </xsl:call-template>
      </xsl:when>

     <xsl:when test="$stereotype='ChoiceGroup'">
       <xsl:call-template name="group">
         <!-- class name of the associated class -->
         <xsl:with-param name="class" select="$grpclass/endsin/@name"/>
         <!-- multiplicity of the associated class -->
         <xsl:with-param name="multiplicity" select="$grpclass/endrole/multiplicity"/>
         <!-- pass the processed groups to avoid recursion -->
         <xsl:with-param name="processed_groups" select="$processed_groups"/>
         <!-- grouptype_char defines the separator between elements of a group and therefore the type of the group --
>
         <xsl:with-param name="grouptype_char" select="string(' | ')"/>
        </xsl:call-template>
```

```
    </xsl:when>

    <xsl:when test="$stereotype='Group'">
     <xsl:call-template name="group">
       <!-- class name of the associated class -->
       <xsl:with-param name="class" select="$grpclass/endsin/@name"/>
       <!-- multiplicity of the associated class -->
       <xsl:with-param name="multiplicity" select="$grpclass/endrole/multiplicity"/>
       <!-- pass the processed groups to avoid recursion -->
       <xsl:with-param name="processed_groups" select="$processed_groups"/>
       <!-- grouptype_char defines the separator between elements of a group and therefore the type of the group --
>
       <xsl:with-param name="grouptype_char" select="string(',')"/>
     </xsl:call-template>
    </xsl:when>


    <!-- we are dealing with a one-to-one association -->
    <xsl:otherwise>
      <xsl:text>(</xsl:text>
      <xsl:value-of select="$grpclass/endsin/@name"/>
      <xsl:if test="$stereotype='Type'"><xsl:text>;</xsl:text></xsl:if>
      <xsl:text>)</xsl:text>
      <xsl:call-template name="write-multiplicity">
        <!-- multiplicity of the associated class -->
        <xsl:with-param name="multiplicity" select="$grpclass/endrole/multiplicity"/>
      </xsl:call-template>

    </xsl:otherwise>

  </xsl:choose>
 </xsl:template>
```

```
<!-- write the content model for a group -->
<xsl:template name="group">
  <xsl:param name="class"/>
  <xsl:param name="multiplicity"/>
  <xsl:param name="processed_groups"/>
  <xsl:param name="grouptype_char"/>
  <xsl:param name="mixed-content" select="string('false')"/>

  <xsl:variable name="associations" select="//association[child::startsin/@type='Class' and
child::startsin/@name=$class]"/>

  <xsl:text>(</xsl:text>
  <!-- check if we are dealing with a mixed content model. If this is the case we add the #PCDATA -->
  <xsl:if test="$mixed-content = 'true'"><xsl:text>#PCDATA | </xsl:text></xsl:if>

  <xsl:for-each select="$associations">
    <xsl:sort select="constraints" data-type="number"/>

    <!-- check if the current is a leaf node or a branch -->
    <xsl:call-template name="check-element">
      <xsl:with-param name="classname" select="endsin/@name"/>
      <xsl:with-param name="multiplicity" select="endrole/multiplicity"/>
      <xsl:with-param name="processed_groups" select="$processed_groups"/>
    </xsl:call-template>

    <xsl:if test="position()&lt; count($associations)">
      <xsl:value-of select="$grouptype_char"/>
    </xsl:if>
  </xsl:for-each>

  <xsl:text>)</xsl:text>
  <xsl:call-template name="write-multiplicity">
    <xsl:with-param name="multiplicity" select="$multiplicity"/>
  </xsl:call-template>

</xsl:template>
```

```
 <xsl:template name="check-element">
   <xsl:param name="classname"/>
   <xsl:param name="multiplicity"/>
   <xsl:param name="processed_groups"/>

   <xsl:variable name="stereotype" select="//classes/class[@name=$classname]/stereotype"/>
   <xsl:choose>
     <xsl:when test="$stereotype='Type' or $stereotype='Element' or $stereotype='ElementANY' or
$stereotype='ElementEMPTY' or $stereotype='ElementPCDATA' or contains($processed_groups,concat( '(',$classname,')'
))">

       <xsl:if test="$multiplicity and $stereotype='Type'"><xsl:text>(</xsl:text></xsl:if>
       <xsl:value-of select="$classname"/>
       <xsl:if test="$stereotype='Type'"><xsl:text>;</xsl:text></xsl:if>
       <xsl:if test="$multiplicity and $stereotype='Type'"><xsl:text>)</xsl:text></xsl:if>

       <xsl:call-template name="write-multiplicity">
         <xsl:with-param name="multiplicity" select="$multiplicity"/>
       </xsl:call-template>
     </xsl:when>
     <xsl:otherwise>
       <xsl:choose>
         <xsl:when test="$stereotype='ChoiceGroup'">
           <xsl:call-template name="group">
             <xsl:with-param name="class" select="$classname"/>
             <xsl:with-param name="multiplicity" select="$multiplicity"/>
             <xsl:with-param name="grouptype_char" select="string(' | ')"/>
             <xsl:with-param name="processed_groups" select="concat ($processed_groups,' (',$classname,')')"/>

           </xsl:call-template>
         </xsl:when>
         <xsl:when test="$stereotype='SequenceGroup'">
                 <xsl:call-template name="group">
```

```
            <xsl:with_param name="class" select="$classname"/>
            <xsl:with_param name="multiplicity" select="$multiplicity"/>
            <xsl:with_param name="grouptype_char" select="string(',')"/>
            <xsl:with_param name="processed_groups" select="concat ($processed_groups,' (',$classname,')')"/>

           </xsl:call-template>
          </xsl:when>
        </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
 </xsl:template>



 <xsl:template name="write-multiplicity">
   <xsl:param name="multiplicity"/>
   <xsl:choose>
     <xsl:when test="$multiplicity='0..1'"><xsl:text>?</xsl:text></xsl:when>
     <xsl:when test="$multiplicity='0..*'"><xsl:text>*</xsl:text></xsl:when>
     <xsl:when test="$multiplicity='1..*'"><xsl:text>+</xsl:text></xsl:when>
  </xsl:choose>
 </xsl:template>
</xsl:stylesheet>
```

Literature

[1] Migrating from XML DTD to XML-Schema using UML. Rational Software White Paper. Available: http://www.rational.com/media/whitepapers/TP189draft.pdf.