International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

M.3020

(07/2017)

SERIES M: TELECOMMUNICATION MANAGEMENT, INCLUDING TMN AND NETWORK MAINTENANCE

Telecommunications management network

Management interface specification methodology

Recommendation ITU-T M.3020

ITU-T M-SERIES RECOMMENDATIONS

TELECOMMUNICATION MANAGEMENT, INCLUDING TMN AND NETWORK MAINTENANCE

Introduction and general principles of maintenance and maintenance organization	M.10–M.299
International transmission systems	M.300–M.559
International telephone circuits	M.560–M.759
Common channel signalling systems	M.760–M.799
International telegraph systems and phototelegraph transmission	M.800–M.899
International leased group and supergroup links	M.900–M.999
International leased circuits	M.1000–M.1099
Mobile telecommunication systems and services	M.1100–M.1199
International public telephone network	M.1200–M.1299
International data transmission systems	M.1300–M.1399
Designations and information exchange	M.1400–M.1999
International transport network	M.2000–M.2999
Telecommunications management network	**M.3000–M.3599**
Integrated services digital networks	M.3600–M.3999
Common channel signalling systems	M.4000–M.4999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T M.3020

Management interface specification methodology

Summary

Recommendation ITU-T M.3020 describes the management interface specification methodology (MISM). It describes the process to derive interface specifications based on user requirements, analysis and design (RAD). Guidelines are given on RAD using unified modelling language (UML) notation; however, other interface specification techniques are not precluded. The guidelines for using UML are described at a high level in this ITU-T Recommendation.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T M.3020	1992-10-05		11.1002/1000/1516
2.0	ITU-T M.3020	1995-07-27	4	11.1002/1000/1517
3.0	ITU-T M.3020	2000-02-04	4	11.1002/1000/4871
4.0	ITU-T M.3020	2007-07-22	4	11.1002/1000/9097
5.0	ITU-T M.3020	2008-07-29	4	11.1002/1000/9550
6.0	ITU-T M.3020	2009-05-14	2	11.1002/1000/9736
7.0	ITU-T M.3020	2010-09-06	2	11.1002/1000/10863
8.0	ITU-T M.3020	2011-07-14	2	11.1002/1000/11368
8.1	ITU-T M.3020 (2011) Amd. 1	2014-07-14	2	11.1002/1000/12202
9.0	ITU-T M.3020	2017-07-22	2	11.1002/1000/13268

Keywords

Management interface, specification methodology.

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Recommendation ITU-T M.3020

Management interface specification methodology

1 Scope

This Recommendation describes the management interface specification methodology (MISM). It describes the process to derive machine-machine interface specifications based on user requirements, analysis and design (RAD). Guidelines are given on RAD using unified modelling language (UML) notation; however, other interface specification techniques are not precluded. The guidelines for using UML are described in this Recommendation. An interface specification addresses management service(s) defined in [ITU-T M.3200] and/or supporting the management processes defined in [ITU-T M.3050.x] series. Such a specification may support part of or one or more management services. The management services comprise of management functions. These functions may reference those defined in [ITU-T M.3400] or the processes defined in [ITU-T M.3050.x] series, specialized to suit a specific managed area, or new functions may be identified as appropriate.

The methodology is applicable to both the traditional manager/agent style of management interfaces [ITU-T M.3010] and the service oriented architecture (SOA) principles adopted for the management architecture of next generation networks [ITU-T M.3060].

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T M.3010] Recommendation ITU-T M.3010 (2000), *Principles for a telecommunications management network.*

[ITU-T M.3050.0] Recommendation ITU-T M.3050.0 (2007), *Enhanced Telecom Operations Map (eTOM) – Introduction*.

[ITU-T M.3050.x] ITU-T M.3050.x (2007) series of Recommendations, *Enhanced Telecom Operations Map (eTOM)*.

[ITU-T M.3060] Recommendation ITU-T M.3060/Y.2401 (2006), *Principles for the Management of Next Generation Networks*.

[ITU-T M.3200] Recommendation ITU-T M.3200 (1997), *TMN management services and telecommunications managed areas: overview.*

[ITU-T M.3400] Recommendation ITU-T M.3400 (2000), *TMN management functions.*

[ITU-T Q.812] Recommendation ITU-T Q.812 (2004), *Upper layer protocol profiles for the Q and X interfaces.*

[ITU-T X.680] Recommendation ITU-T X.680 (2015) | ISO/IEC 8824-1:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

[ITU-T X.681] Recommendation ITU-T X.681 (2015) | ISO/IEC 8824-2:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*

[ITU-T X.501] Recommendation ITU-T X.501 (2016) | ISO/IEC 9594-2:2016, *Information technology – Open Systems Interconnection – The Directory: Models.*

[ITU-T X.722] Recommendation ITU-T X.722 (1992) | ISO/IEC 10165-4:1992, *Information technology – Open Systems Interconnection – Structure of management information: Guidelines for the definition of managed objects*.

[ITU-T Z.100] Recommendation ITU-T Z.100 (2016), *Specification and Description Language (SDL) – Overview of SDL-2010*.

[OMG UML1] Object Management Group (2011), *Unified Modeling Language (OMG UML), Infrastructure, Version 2.4.1.*

[OMG UML2] Object Management Group (2011), *Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1.*

A list of non-normative references can be found in the Bibliography.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 user [ITU-T M.3010]

3.1.2 management service [ITU-T M.3010]

3.1.3 management function set [ITU-T M.3010]

3.1.4 activity diagram [OMG UML1]

3.1.5 actor [OMG UML1]

3.1.6 association [OMG UML1]

3.1.7 class [OMG UML1]

3.1.8 class diagram [OMG UML1]

3.1.9 classifier [OMG UML1]

3.1.10 collaboration diagram [OMG UML1]

3.1.11 composition [OMG UML1]

3.1.12 modelElement [OMG UML1]

3.1.13 sequence diagram [OMG UML1]

3.1.14 state diagram [OMG UML1]

3.1.15 stereotype [OMG UML1]

3.1.16 use case [OMG UML1]

3.1.17 reference point [ITU-T M.3060]

3.1.18 distinguished name [ITU-T X.501]

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 agent: Encapsulates a well-defined subset of management functionality. It interacts with managers using a management interface. From the manager's perspective, the agent behaviour is only visible via the management interface.

NOTE – Considered equivalent to IRPAgent [b-3GPP TS 32.150].

3.2.2 information object class: Describes the information that can be passed/used in management interfaces and is modelled using the stereotype "Class" in the UML meta-model. For a formal definition of information object class and its structure of specification, see Annex B.

3.2.3 information service: Describes the information related to the entities (either network resources or support objects) to be managed and the way that the information may be managed for a certain functional area. Information services are defined for all IRPs.

NOTE – Considered identical to the definition of information service found in [b-3GPP TS 32.150].

3.2.4 information type: Specification of the type of input parameters of operations.

3.2.5 integration reference point: An architectural concept that is described by a set of specifications for the definition of a certain aspect of the management interface, comprising a requirements specification, an information service specification, and one or more solution set specifications.

NOTE – Considered identical to the definition of IRP found in [b-3GPP TS 32.150].

3.2.6 lower camel case: It is the practice of writing compound words in which the words are joined without spaces. Initial letter of all except the first word shall be capitalized. Examples: 'managedNodeIdentity' and 'minorDetails' are the lower camel case (LCC) for "managed node identity" and "minor details" respectively.

3.2.7 management goals: High-level objectives of a user in performing management activities.

3.2.8 management interface: The realization of management capabilities between a manager and an agent, allowing a single manager to use multiple agents and a single agent to support multiple managers.

NOTE – Q, C2B/B2B and Itf-N (3GPP) are examples of management interfaces.

3.2.9 management role: Defines the activities that are expected of the operational staff or systems that perform telecommunications management. Management roles are defined independent of other components, i.e., telecommunications resources and management functions.

3.2.10 management scenario: A management scenario is an example of management interactions from a management service.

3.2.11 manager: Models a user of agent(s) and it interacts directly with the agent(s) using management interfaces.

Since the manager represents an agent user, it gives a clear picture of what the agent is supposed to do. From the agent perspective, the manager behaviour is only visible via the management interface.

NOTE – Considered equivalent to IRPManager [b-3GPP TS 32.150].

3.2.12 matching information: Specification of the type of a parameter (possibly reference to IOC or attribute of IOC).

3.2.13 naming attribute: It is a class attribute that holds the class instance identifier.

NOTE – See examples of naming attribute in [b-3GPP TS 32.300].

3.2.14 protocol-neutral specification: Defines the management interfaces in support of management capabilities without concern for the protocol and information representation implied or required by, e.g., CORBA and XML.

3.2.15 protocol-specific specification: Defines the management interfaces in support of management capabilities for one specific choice of management technology (e.g., CORBA).

NOTE – Considered equivalent to solution set [b-3GPP TS 32.150].

3.2.16 telecommunications resources: Telecommunications resources are physical or logical entities requiring management, using management services.

3.2.17 upper camel case: It is the lower camel case except that the first letter is capitalized. Examples: 'ManagedNodeIdentity' and 'MinorDetails' are the upper camel case (UCC) for "managed node identity" and "minor details" respectively.

3.2.18 well known abbreviation: An abbreviation can be used as the modelled element name or as a component of a modelled element name. The abbreviation, when used in such manner, must be documented in the same document where the modelled element is defined.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

ADM Administrative (usage: requirements category)

ASN.1 Abstract Syntax Notation One

CM Conditional-Mandatory

CO Conditional-Optional

CON Conceptual (usage: requirements category)

CORBA Common Object Request Broker Architecture

DN Distinguished Name

FUN Functional (usage: requirements category)

GDMO Guidelines for the Definition of Managed Objects

IDL Interface Definition Language

IOC Information Object Class

IRP Integration Reference Point

IS Information Service

LCC Lower Camel Case

MCC Mobile Country Code

MISM Management Interface Specification Methodology

MNC Mobile Network Code

NE Network Element

NON Non-functional (usage: requirements category)

OO Object Oriented

OSI Open Systems Interconnection

RDN Relative Distinguished Name

SDL Specification and Description Language

SOA	Service Oriented Architecture
SS	Solution Set
SNC	Sub Network Connection
TP	Termination Point
TS	Technical Specification
UCC	Upper Camel Case
UML	Unified Modelling Language
WKA	Well Known Abbreviation
XML	Extensible Markup Language

5 Conventions

Clause A.1 contains conventions applicable to the requirements phase.

Clause B.1 contains conventions applicable to the analysis phase.

6 Requirements for methodology and notational support

In developing the methodology and choosing a notation, the following requirements apply:

1) The methodology, including the choice of notation, shall support the capture of all the relevant requirements of the problem space, namely telecommunications management.
2) The methodology facilitates the production of requirements, its corresponding Analysis|Information Services and their corresponding Design Specifications|Solution Sets.
3) The notation shall facilitate unambiguous generation of the specification in the target management protocol profile. The methodology does not address possible choices of protocol services (e.g., CORBA Security Service).

   NOTE – Management protocols applicable for ITU-T use are specified in [ITU-T Q.812].
4) The methodology shall allow specification of mandatory and optional items in all three phases. It also specifies the relation of mandatory|optional items between the three phases.
5) It should be possible to generate, from the protocol-neutral specification (Analysis|IS), interoperable language specific definitions, i.e., Design|SS (for example UML to interface definition language (IDL), UML to GDMO/ASN.1).

7 Methodology

7.1 General considerations

The purpose of this methodology is to provide a description of the processes leading towards the definition of machine-machine management interfaces.

7.2 Application and structure of the methodology

The management interface specification methodology (MISM) specifies a three-phase process with features that allow traceability across the three phases. The three phases apply industry-accepted techniques using object oriented analysis and design principles. The three phases are requirements, analysis and design. The techniques should allow the use or development of commercially available support tools. Different techniques may be used for the phases depending on the nature of the problem.

7.3 Detailed methodology

7.3.1 General

The requirements and analysis phases produce UML specifications. The design phase uses network management paradigm specific notation. The outputs of the 3 phases are:

– Requirements phase – Requirements.

– Analysis phase – Implementation independent specification.

– Design phase – Technology specific specification.

Initially, the design phase will be developed using a manual or customized approach. When interoperable protocol specific definition can be generated by tools, then UML notation can be applied to the design phase.

The clauses below describe the three phases.

7.3.2 Requirements

The requirements for the problem being solved fall into two classes. The first class of requirements is referenced here as business requirements. A subject matter expert on the topic shall be able to determine that the requirements adequately represent the needs of the management problem being solved. The second class is referred to as specification requirements. These requirements shall provide sufficient details so that the interface definition in the analysis and design phases can be developed. As final interface definitions must be traceable to the requirements, it may be necessary to have interaction between the three phases. Any ambiguity in the requirements will have to be resolved by this interaction to assure that an implementable specification can be developed.

Human-computer interface data may be specified in the second class of requirements. These requirements may have great impact on concepts and data designed in the subsequent phases. For more details, see Appendix I.

Different techniques may be used to specify the two classes of requirement. Irrespective of the technique, the readability of the requirements is critical. The requirements themselves are not required to be in a machine-readable notation as long as readability and traceability are possible. Enumerating requirements is the recommended solution to delineate the different requirements for traceability.

The requirements phase includes identifying aspects such as security policy, scope of the problem domain in terms of the applications, resources, and roles assumed by the resources. The requirements specify roles, responsibilities, and the relationships between the constituent entities for the problem space. Different techniques, including textual representation, may be used to specify the business level requirements. In order to facilitate traceability of these requirements to the design and implementation phases, enumerating requirements is recommended.

The problem must be bounded with a specific scope. One way to determine the scope is by using the management services identified in [ITU-T M.3200] and function sets identified in [ITU-T M.3400]. Requirements are specified using the resources being managed and management functions. An alternative to the management services approach is described in [ITU-T M.3050.x] "enhanced Telecom Operations Map (eTOM)" which provides a business process based approach.

The relationship between the [ITU-T M.3200] and [ITU-T M.3050.x] approaches is described in [ITU-T M.3050.0].

Management functions must be grouped and supported within applications that address specific business needs, so the linkage between the eTOM processes, the [ITU-T M.3200] management services, the [ITU-T M.3400] management function sets and management functions is important to assist in making this grouping clear and effective. Augmenting [ITU-T M.3400] may be required in order to meet the business requirements of the problem.

UML use cases and scenarios should be used to interact with subject matter experts in capturing the business level requirements. The requirements should also identify the failure conditions visible to the business process.

NOTE – It is not required that every requirement be expressed as a use case.

The requirements produced must be complete and detailed. The recursive nature of the methodology is used to achieve this completeness. The completeness of the requirements (clear and well-documented) drives the analysis and design phases.

Guidelines and template for requirement structure and identification are described in clause A.1.2.

Use cases are goals that are fulfilled through a sequence of steps. Each step can be considered as a sub-goal of the use case. As such each step represents either another use case (subordinate use case) or an autonomous action that is at the lowest level of the case decomposition.

Guidelines and template for use cases are described in clause A.1.2.

7.3.3 Analysis

In the analysis phase, the requirements are used to identify the interacting entities, their properties and the relationships among them. This allows the interfaces offered by the entities to be defined. In the UML notation, these entities become classes. The class descriptions along with the interfaces exposed should be traceable to the requirements. The relationship among the classes, defined in the analysis specification, and the classes in the design specification is not necessarily one to one.

This phase should take into account the needs of human-computer interface data (i.e., the information model must contain sufficient information so that designs can be developed based on the analysis results).

This Recommendation gives high-level guidance on the use of UML notation to support management interface specification; however, Specification and Description Language (SDL) [ITU-T Z.100] might be used to augment the UML definitions.

The analysis phase should be independent of design constraints. For example, the analysis may be documented using object oriented (OO) principles even though the design may use a non-object-oriented technology. The information specified in the analysis phase includes class descriptions, data definitions, class relationships, interaction diagrams (sequence diagrams and/or collaboration diagrams), state transition diagrams and activity diagrams. The class definitions include specification of operations, notifications, attributes and behaviour captured as notes or textual description.

Protocol-neutral common management services (if available) – or other existing services – should be reused during the analysis phase in order to support management interface harmonization.

Guidelines and template for use cases are described in Annex A.

The analysis template uses information type as one characteristic to describe information object class (IOC) attributes and operation/notification parameters. The valid information type(s) that can be used and their semantics are defined in Annex E.

7.3.4 Design

7.3.4.1 General

In the design phase, an implementable interoperable interface specification is produced. This will involve the selection of a target specification language. The design phase specifications are dependent on the specific management paradigm (e.g., IDL for CORBA interfaces).

This phase distinguishes three kinds of specifications of data: management paradigm (e.g., extensible markup language (XML)) dependent design of data to be communicated across multiple interfaces (e.g., fault and performance), messages (e.g., alarm report) to be communicated over each individual interface, and encoding method of the data (e.g., compressed XML) consistent with a particular paradigm.

The selection of a specific management paradigm is addressed in other ITU-T Recommendations. An overview is provided in the following clauses.

In the design phase, it is recommended that the UML descriptions from the requirements and analysis phases be referenced to augment behavioural specification. For example, the behaviour definition of guidelines for the definition of managed objects (GDMO) can reference state charts, sequence diagrams and class definition in the analysis phase. If required, additional UML diagrams describing interactions between entities, corresponding to specific protocol paradigms, may be included.

As additional paradigms are adopted for use by management, the notations/languages defined by these paradigms will be used.

7.3.4.2 CORBA

In the context of common object request broker architecture (CORBA) based management, the information model is defined using IDL.

7.3.4.3 GDMO

In the context of the paradigm based on open systems interconnection (OSI) systems management [ITU-T X.722], the design specification is the information model specification using GDMO templates for managed object classes, attributes, behaviour, notifications, actions, naming instances of the class, and error/exception specifications. The syntax of the information is specified using Abstract Syntax Notation One (ASN.1) notation [ITU-T X.680].

In GDMO, the object class hierarchy specifies the properties of the object classes that are needed for management. Extensive use of inheritance (super and subclasses) is needed to benefit the most from the reuse of specifications. The object classes are specified using the templates from [ITU-T X.722]. The templates defining the information model should be registered (according to the rules of [ITU-T X.722]) with a value for the ASN.1 object identifier. For those object classes that are already specified in other ITU-T Recommendations and ISO standards, only a reference to the particular Recommendation and object class is needed. Naming is not a part, nor the purpose, of the object class hierarchy.

7.3.4.4 XML

For further study.

8 Management interface specifications

A management interface specification includes the requirements, analysis and design specifications discussed in clause 7. A structure for specifying these specifications is provided in Annexes A, B and C.

These techniques and supporting notations are also applicable when designing a system to the management interface specifications, even though system design is not considered as part of the ITU-T management Recommendations. They assist in describing how the interface specifications are applied in managing the resources within a system such as a network element (NE).

9 Traceability in MISM process

In order to achieve traceability between requirements, analysis and design, it is necessary that appropriate identification be assigned. Traceability is supported through references between entities specified within each phase and between phases. Traceability is from design|solution set to analysis|information services and from analysis|information services to requirements. Traceability is further applicable between artifacts of the requirements specification and between artifacts of the analysis|information service, e.g., between use cases and textual requirements. Requirements should be identified as described in clause 7.3.2. The analysis phase output specifies for the various use cases further detailed information requirements. The design phase should point to the various diagrams and text in the analysis phase output. The pointer may be in terms of a reference to the appropriate clauses.

Traceability from the design phase to subject matter level requirements is usually indirect. This is required because the output of the phases is defined to different level of details.

Guidelines for traceability between the requirements phase and the analysis phase are described in Annex B.

The following mechanism for traceability with requirements, etc., specified in other documents (possibly not following the advocated identification schema) is recommended:

forum/body "::" document ID "::" id

where "id" could be one of:

1) requirement ID;
2) use case ID;
3) requirement title/text;
4) use case title;
5) subclause of the document which uniquely identifies a requirement or use case.

Examples:

3GPP::32.111-1::getAlarmList

ITU-T::M.3016::1.5.1.2

10 Documentation structure

Even though there are three phases, the documentation of the interface may combine their outputs into one or more documents. It is recommended that the requirements and analysis be combined and separate design documents are developed for each specific network management protocol paradigm.

Annex A

Requirements

(This annex forms an integral part of this Recommendation.)

The following are guidelines for specification of requirements.

The normal (or full format) requirements template is found in clause A.2. In addition, a simplified requirements template is defined and found in clause A.3.

A.1 Conventions

A.1.1 Use of UML notation for requirements

Table A.1 identifies the correspondence between management concepts and UML notation. This Recommendation specifies the high-level concepts and notations to be used in the different phases. Stereotypes are used to extend UML notation. The approved stereotypes for use within the management environment are included in this Recommendation (see Annex C).

Table A.1 – Requirements concepts

Management concept	UML notation	Comment
user	Actor	A user is modelled as an actor.
management role	Actor	An actor plays a role. It is normally advisable to only model a single role for each actor.
management function	use case	A management function is modelled by one or more use cases.
management function set	use case	A management function set is a composite use case with each management function (potentially) modelled as a separate use case.
management service	use case	A management service is modelled as a high-level use case.
management scenario	sequence diagram	Sequence diagrams are preferred over collaboration diagrams.
telecommunication resource type	Class	The class diagrams depict the property details of the telecommunications resource type, at the level of detail appropriate to the phase of the methodology.
management goals	–	Management goals are captured as textual descriptions as there is no applicable UML notation.

A.1.2 Use case template

When use cases are provided, the conventions and templates in Table A.2 should be followed.

Table A.2 – Use case template

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal(*)	This is the objective/end result the use case strives to achieve and should be a concise statement of what the use case should achieve in a successful scenario. There may be a statement about priority relative to other use cases and required performance of the use case, e.g.: • Real Time. • Near real time. • Not real time.	
Actors and roles(*)	The names of actors/roles involved in the use case including role characteristic for each actor.	
Telecom resources	The names of the telecommunication resources involved in the use case.	
Assumptions	A description of the environment providing a context for the use case. Assumptions are mutually exclusive to pre-conditions. Assumptions are concerned with static properties.	
Pre-conditions	A list of all system and environment conditions that must be true before the use case can be triggered. Pre-conditions are mutually exclusive to assumptions. Pre-conditions are related to dynamic properties and can result in an exception. This is never the case with assumptions.	

Table A.2 – Use case template

Use case stage	Evolution/Specification	<<Uses>> Related use
Begins when	The name of the single event that triggers the start of the use case. Optional and normally not used to specify triggers such as "when the manager must retrieve information".	
Step 1(*) (M|O)	A use case describes a list of steps (manual and automated) that are necessary to accomplish the goal of the use case. Steps may invoke other use cases. Steps are numbered for traceability. Each step is identified as being mandatory (M) or optional (O). Sub-steps are identified relative to the containing step, e.g.: Step n Step n.1 Step n.2 where n.1 and n.2 are sub-steps of step n.	Reference to a used use case.
Step n (M|O)	Steps added as necessary and in a logical sequence.	
Ends when(*)	The list of event(s) that indicates the use case completion. NOTE – In this context, "event" should be considered in the most general sense and not limited to, e.g., notifications exchanged across a management interface. As an example, the completion of processing can be considered an event that indicates completion of a use case.	
Exceptions	A summary list of exception conditions and faults detected by the use case during its operation.	
Post-conditions	A list of all system and environmental conditions that must be true when the use case has completed. The statement of post-conditions determines if the use case is expected to be fully successful, partially successful or even to have failed in order to be completed.	
Traceability(*)	Requirements or use case exposed by the use case.	
NOTE – Fields marked with "*" are mandatory for all use case specifications. Other fields are only mandatory when relevant for the specific use case.		

A.1.3 Requirements categories

It is useful to classify requirements in different categories. The following categories are considered relevant for MISM:

– Conceptual (CON) – Identifies a concept, data type, relationship, format, or structure.

– Functional (FUN) – Identifies a functional capability, dynamic situation, a sequence, timing parameters, or an interaction.

– Non-functional (NON) – Non-functional requirements, including abnormal conditions, error conditions and bounds of performance.

– Administrative (ADM) – System administration and operational requirements not related to the use cases normal operations.

Requirements should be written based on the following template:

REQ-Label-Category-Number {Category, number} Details {Source Citation}

where "Label" is an abbreviation for the Recommendation (or part thereof). The set of labels is not finite and not subject for standardization.

Guidelines on requirements numbering can be found in Appendix III.

A.2 Requirements template

1 Concepts and background

Define major goals and objectives and the applicable management interfaces (and reference points) for this specification. Use [ITU-T M.3200] categorization as a source for identifying the management service(s) supported by this interface.

This subclause should give a clear description of the users' benefit, i.e., the reason for performing this management service. Background and context should be added as necessary, but the explanatory and descriptive parts should be separated. Supporting background information, where required, should be placed in an appendix.

1.a SubClauseTitle

SubClauseTitle is the name of the subclause.
"a" represents a number, starting at 1 and increasing by 1 with each new subclause.
The use of subclauses is optional.

2 Business level requirements

2.1 Requirements

2.1.a SubSetTitle
SubSetTitle is the name of a sub-set of the business level requirements.
"a" represents a number, starting at 1 and increasing by 1 with each new sub-set.
The use of sub-sets is optional and all business level requirements can be stated in subclause 2.1 (requirements).
List major requirements in text, and identify use cases with actor/role and resources. The high-level use cases (subclause 2.4 below) should bring out the business level requirements and are distinguished from the specification requirements by not refining to lower levels. Clause 2.4 contains many examples of what makes up the high-level use cases. Policy-related information (e.g., security, persistence) are candidates for inclusion at this level. Numbering the requirements is required for traceability.
Requirements should be specified as described in clause A.1.3. Within a requirements specification, it is suggested that requirements be written in the sequence of clause A.1.3 (either for the entire specification or for each sub-set).
Use of requirements categories is optional, and – when used – a subset of the categories can be applied.
As an example, conceptual requirement number 23 in Recommendation tagged 'SM' would be specified as follows:

Identifier	Definition
REQ-SM-CON-23	A Service Order consists of a name, address, phone number, service description and an optional FAX number for contacts {T1M1.5 Document 246 11/96}

One or more tables can be used with supportive text between tables as necessary.

2.2 Actor roles

A textual description of the actor (see clause 3) is included here.

2.3 Telecommunication resources

Textual description of the relevant resources (see clause 3) required to support the use cases are presented here.

2.4 High-level use cases

A high-level use case diagram may be presented. In order to understand the use case by subject matter experts, they should be augmented with a textual description for each use case. The description should serve two purposes: to capture the domain experts' knowledge and to validate the models in analysis and design phases with respect to the requirements. An example of a high-level use case diagram is given in Appendix I.

2.4.a UseCaseName

UseCaseName is the name of the use-case.

"a" represents a number, starting at 1 and increasing by 1 with each new definition of a use case.

This subclause is repeated for each high-level use case defined for the interface specification requirements.

The high-level use cases may identify the various function sets defined in [ITU-T M.3400] or the management processes defined in [ITU-T M.3050.x]. These use cases may be further refined as described in the specification level requirement subclause below by using stereotypes such as "include" and "extend".

If appropriate, sequence diagrams may be used. However, at the high-level requirements these diagrams are not expected to be used. When the use cases at this level are further decomposed in the next level of requirements, these diagrams may be more suitable.

The traceability of the next level of requirements from this level may be identified by how each function set is further refined with new use cases.

A set of use case tables, using the template defined in Table A.2, may be used to represent the significant capabilities studied at a level of abstraction appropriate to the problem being analysed.

The level of detail, and extent of coverage provided in the use cases is dependent upon the authoring team's familiarity with the subject matter and is therefore subjective. The lower levels of details are most likely an indication of analysis rather than requirements capture.

It is permitted to develop successively more detailed analysis of each step of a higher abstraction level use case by referring to the more detailed use case in the table cell reserved for this purpose. It is emphasized this does not have to be done, and is subjective depending upon the need of the author/group.

The following list is provided to aid the initial identification of suitable use cases:

– *What is the main purpose of the system?*
– *What types of people/system need to interact with the system?*
– *How can these people/systems be grouped or abstracted to roles?*
– *What are the start up, normal running, failure and recovery aspects of the system?*
– *What types of reports or data may be needed from the system?*
– *Which special activities are required (e.g., based on times of day and network loads)?*

It is useful to document use cases in a common manner. The following structure is suggested:

– *<use case table> (see Table A.2)*
– *<optional sequence diagram(s)>*
– *<optional state chart(s)>*

3 Specification level requirements

3.1 Requirements

The business level requirements are further refined here using management functions from [ITU-T M.3400]. Since [ITU-T M.3400] is not exhaustive enough to address all management services for all managed areas, it is expected that new functions will be required. The new functions should be included in the requirements as described below.

3.1.a SubSetTitle

SubSetTitle represents the name of a subset of specification level requirements.

"a" represents a number, starting at 1 and increasing by 1 with each new sub-set.

The use of sub-sets is optional and all specification level requirements can be stated in subclause 3.1 (requirements).

List major detailed and concrete requirements in text, and identify use cases with actor/role and resources. The use cases in subclause 3.4 should bring out specification level requirements with lower level details and be more implementation-oriented compared to the business level use case requirements. Numbering the requirements is required for traceability.

Requirements should be specified as described in clause A.1.3. Within a requirements specification, it is suggested that requirements be written in the sequence of clause A.1.3 (either for the entire specification or for each sub-set).

Use of requirements categories is optional, and – when used – a subset of the categories can be applied.

As an example, functional requirement number 33 in a Recommendation tagged 'OM' would be specified as follows:

Identifier	Definition
REQ-OM-FUN-33	A pending operation can be cancelled by the initiator.

One or more tables can be used with supportive text between tables as necessary.

Specification level requirements should follow the conventions and templates defined in clause A.1.

3.2 Actor roles

A list of all actors and textual description of actors not already defined in the business level requirements is included here.

3.3 Telecommunication resources

A list of all passive resources and textual description of resources not already defined in the business level requirements is presented here.

3.4 Use cases

The high-level use cases are further refined here using several specification level use cases, each of which will be further explained in detail in a subclause as described below.

3.4.a UseCaseName

UseCaseName is the name of the use-case.

"a" represents a number, starting at 1 and increasing by 1 with each new definition of a use case.

If appropriate, sequence and state chart diagrams may be used.

NOTE – Guidelines and criteria for use of sequence diagrams and state chart diagrams are for further study.

Use case specifications should follow the conventions and templates defined in clause A.1.

A.3 Simplified requirements template

The simplified requirements template is an alternative template for use in cases when only the textual requirements are required. A separate template is defined to avoid ambiguity that would result by adding options in the full-form template described in clause A.2.

1 Concepts and background

Define major goals and objectives and the applicable management interfaces (and reference points) for this specification. Use [ITU-T M.3200] categorization as a source for identifying the management service(s) supported by this interface.

This clause should give a clear description of the users' benefit, i.e., the reason for performing this management service. Background and context should be added as necessary, but the explanatory and descriptive parts should be separated. Supporting background information, where required, should be placed in an appendix.

1.a SubClauseTitle

SubClauseTitle is the name of the subclause.

"a" represents a number, starting at 1 and increasing by 1 with each new subclause.

The use of subclauses is optional.

2 Requirements

2.a SubSetTitle

SubSetTitle is the name of a sub-set of the business level requirements.

"a" represents a number, starting at 1 and increasing by 1 with each new sub-set.

The use of sub-sets is optional and all business level requirements can be stated in clause 2 (requirements).

List major requirements in text, and identify use cases with actor/role and resources. The use cases should bring out high-level requirements and are distinguished from the specification requirements by not refining to lower levels. Policy-related information (e.g., security, persistence) are candidates for inclusion at this level. Numbering the requirements is required for traceability.

Requirements should be specified as described in clause A.1.3. Within a requirements specification, it is suggested that requirements are written in the sequence of clause A.1.3 (either for the entire specification or for each sub-set).

Use of requirements categories is optional, and – when used – a subset of the categories can be applied.

As an example, conceptual requirement number 23 in a Recommendation tagged 'SM' would be specified as follows:

Identifier	Definition
REQ-SM-CON-23	A Service Order consists of a name, address, phone number, service description and an optional FAX number for contacts {T1M1.5 Document 246 11/96}

One or more tables can be used with supportive text between tables as necessary.

Annex B

Analysis

(This annex forms an integral part of this Recommendation.)

B.1 Conventions

B.1.1 Mandatory, optional and conditional qualifiers

B.2 Analysis template

1 Concepts and background

1.a SubClauseTitle

2 Model

2.1 Imported and associated information entities

2.1.1 Imported information entities and local labels

2.1.2 Associated information entities and local labels

2.2 Class diagram

2.2.1 Relationships

2.2.2 Inheritance

2.3 Class definitions

2.3.a InformationObjectClassName

2.3.a.1 Definition

2.3.a.2 Attributes

2.3.a.3 Attribute constraints

2.3.a.4 Notifications

2.3.a.5 State diagram

2.4 Attribute definitions

2.4.1 Attribute properties

2.4.2 Constraints

2.5 Common notifications

2.5.1 Alarm notifications

2.5.2 Configuration notifications

2.6 System state model

3 Interface definition

3.1 Class diagram representing interfaces

3.2 Generic rules

3.b Interface InterfaceName (supportQualifier)

3.b.a Operation OperationName (supportQualifier)

3.b.a.1 Definition

3.b.a.2 Input parameters

3.b.a.3 Output parameters

3.b.a.4 Pre-condition

The following are guidelines for specification of the results of the analysis phase.

The analysis template is based on the 3rd Generation Partnership Project (3GPP) information service [b-3GPP TS 32.157] and augmented to meet additional requirements on the methodology (e.g., traceability).

For a management interface specification, both subclauses 2.2 and 2.3 of "Analysis" template indicated in clause B.2 shall be used. For an information model (e.g., a network resource model), only subclause 2.2 shall be used.

The analysis template uses Information Type as one characteristic to describe IOC attributes and operation/notification parameters. The valid Information Type(s) that can be used and their semantics are defined in Annex E.

An example of the use of this template can be found in Appendix II.

The constructs "Analysis|Information Service" and "Design|Solution" sets are used to denote the equivalent, but differently named, specifications developed by ITU-T and 3GPP.

B.1 Conventions

B.1.1 Mandatory, optional and conditional qualifiers

This clause defines a number of terms used to qualify the relationship between the Analysis|Information service, the Design|Solution sets and their impact on the interface implementations. The qualifiers defined in this clause are used to qualify agent behaviour only. This is considered sufficient for the specification of the management interfaces.

Analysis specification|IS specifications define IOC attributes, interfaces, operations, notifications, operation parameters and notification parameters. They can have the following support/read/write qualifiers: M, O, CM, CO, C.

Definition of qualifier M (Mandatory):

- Used for items that shall be supported.

Definition of qualifier O (Optional):

- Used for items which may or may not be supported.

Definition of qualifier CM (Conditional-Mandatory):

- Used for items that are mandatory under certain conditions, specifically:
  - All items having the support qualifier CM shall have a corresponding constraint defined in the Recommendation|IS specification. If the specified constraint is met, then the items shall be supported.

Definition of qualifier CO (Conditional-Optional):

- Used for items that are optional under certain conditions, specifically:
  - All items having the support qualifier CO shall have a corresponding constraint defined in the Recommendation|IS specification. If the specified constraint is met, then the items may be supported.

Definition of qualifier C (SS-Conditional):

- Used for items that are only applicable for certain but not all Designs|Solutions sets (SSs).

Design|SS specifications define the SS-equivalents of the IOC attributes, operations, notifications, operation parameters and notification parameters. These SS-equivalents can have the following support/read/write qualifiers: M, O, CM and CO.

The mapping of the qualifiers of Analysis|IS-defined constructs to the qualifiers of the corresponding SS-constructs is defined as follows:

- For qualifier M, O, CM and CO, each IS-defined item (operation and notification, input and output parameter of operations, input parameter of notifications, information relationship and information attribute) shall be mapped to its equivalent(s) in all SSs. Mapped equivalent(s) shall have the same qualifier as the IS-defined qualifier.
- For qualifier C, each IS-defined item shall be mapped to its equivalent(s) in at least one SS. Mapped equivalent(s) can have support qualifier M or O.

Table B.1 defines the semantics of qualifiers of the equivalents, in terms of support from the agent perspective.

Table B.1 – Semantics for qualifiers used in Design|Solution sets

Mapped SS equivalent	**Mandatory**	**Optional**	**Conditional-Mandatory (CM)**	**Conditional-Optional (CO)**
Mapped notification equivalent	The agent shall generate the notification.	The agent may or may not generate it.	The agent shall generate this notification if the constraint for this item is satisfied.	The agent may choose whether or not to generate it. If the agent chooses to generate it, the constraint for this notification must be satisfied.
Mapped operation equivalent	The agent shall support it.	The agent may or may not support this operation. If the agent does not support this operation, the agent shall reject the operation invocation with a reason indicating that the agent does not support this operation. The rejection,	The agent shall support this operation if the constraint for this item is satisfied.	The agent may support this operation if the constraint for this item is satisfied.

Table B.1 – Semantics for qualifiers used in Design|Solution sets

Mapped SS equivalent	Mandatory	Optional	Conditional-Mandatory (CM)	Conditional-Optional (CO)
		together with a reason, shall be returned to the manager.		
Input parameter of the mapped operation equivalent	The agent shall accept and behave according to its value.	The agent may or may not support this input parameter. If the agent does not support this input parameter and if it carries meaning (i.e., it does not carry no-information semantics), the agent shall reject the invocation with a reason (that it does not support the parameter). The rejection, together with the reason, shall be returned to the manager.	The agent shall accept and behave according to its value if the constraint for this item is satisfied.	The agent may accept and behave according to its value if the constraint for this item is satisfied.
Input parameter of mapped notification equivalent AND output parameter of mapped operation equivalent	The agent shall supply this parameter.	The agent may supply this parameter.	The agent shall supply this parameter if the constraint for this item is satisfied.	The agent may supply this parameter if the constraint for this item is satisfied.
Mapped IOC attribute equivalent	The agent shall support it.	The agent may support it.	The agent shall support this attribute if the constraint for this item is satisfied.	The agent may support this attribute if the constraint for this item is satisfied.

B.2 Analysis template

1 Concepts and background

This clause should provide an introduction to the management interface specification analysis.

1.a SubClauseTitle

SubClauseTitle is the name of a subclause.

"a" represents a number, starting at 1 and increasing by 1 with each new subclause.

The use of subclauses is optional.

2 Model

This clause shall be used for all specifications (both management interface specifications and information model only specifications).

2.1 Imported and associated information entities

2.1.1 Imported information entities and local labels

This subclause identifies a list of information entities (e.g., information object class, interface, attribute) that have been defined in other specifications and that are imported in the present (target) specification. All *imported entities shall be treated as defined locally in the target specification. One usage for import is for inheritance purpose.*

Each element of this list is a pair (label reference, local label). The local label contains the name of the information entity that appears in the target specification, and the entity name in the local label shall be kept identical to the name defined in the original specification. The local label of imported information entities can then be used throughout the specification instead of the label reference.

This information is provided in a table as shown below.

Label reference	Local label

Guidelines on entity import as well as IOC properties and inheritance can be found in Annex F.

2.1.2 Associated information entities and local labels

This clause identifies a list of information entities (e.g., information object class, interface, attribute) that have been defined in other specifications and that are associated with the information entities defined in the present (target) specification. For the associated information entity, only its properties, attribute of an instance of the associated information entity) used as associated information needs to be supported locally in the target specification.

Each element of this list is a pair (label reference, local label). The label reference contains the name of the original specification where the information entity is defined, the information entity type and its name. The local label contains the name of the information entity that appears in the target specification. The local label can then be used throughout the target specification instead of that which appears in the label reference.

This information is provided in a table as shown below.

Label reference	Local label

2.2 Class diagram

2.2.1 Relationships

This first set of diagrams represents all classes defined in this specification with all their relationships and all their attributes, including relationships with imported and associated information entities (if any). These diagrams shall contain information object class cardinalities (for associations as well as containment relationships) and may also contain role names. These shall be UML compliant class diagrams (see also Annex C).

Characteristics (relationships) of imported and associated information object classes need not be repeated in the diagram.

Allowable classes are specified in Annex C.

*Use this as the first paragraph: "*This clause depicts the set of classes (e.g., IOCs) that encapsulates the information relevant for this management specification. This clause provides an overview of the relationships between relevant classes in UML. Subsequent clauses provide more detailed specification of various aspects of these classes.*"*

2.2.2 Inheritance

This second set of diagrams represents the inheritance hierarchy of all information object classes defined in this specification. These diagrams do not need to contain the complete inheritance hierarchy but shall at least contain the parent classes of all classes defined in the present document. By default, a class inherits from the class "top".

Characteristics (attributes, relationships) of imported classes need not be repeated in the diagram.

NOTE 1 – Some inheritance relationships presented in subclause 2.2.2 can be repeated in subclause 2.2.1 to enhance readability.

*Use "*This subclause depicts the inheritance relationships.*" as the first paragraph.*

2.3 Class definitions

Each class is defined using the following structure.

Inherited items (attributes, etc.) shall not be shown, as they are defined in the parent classes(es) and thus valid for the subclass.

2.3.a InformationObjectClassName

InformationObjectClassName is the name of the information object class.

"a" represents a number, starting at 1 and increasing by 1 with each new definition of a class.

2.3.a.1 Definition

This subclause is written in natural language. This subclause refers to the class itself.

Optionally, information on traceability back to one or more requirements supported by this class can be defined here, in the following form:

Referenced specification	Requirement label	Comment

2.3.a.2 Attributes

This clause presents the list of attributes, which are the manageable properties of the class.

Each attribute is characterised by some of the attribute properties (see Table C.1), i.e., supportQualifier, isReadable, isWritable, isInvariant and isNotifyable.

The legal values and their semantics for attribute properties are defined in Annex C.

This information is provided in a table as shown below.

Attribute name	Support qualifier	isReadable	isWriteable	isInvariant	isNotifyable

The attributeName indicates the name of the attribute. An attributeName with an "" sign indicates that this attribute is a naming attribute that will be used in the DN/RDN naming tree. The value of the naming attribute in each object instance shall be unique under its parent object instance.*

In case there is one or more attributes related to role (see clause 2.10 of Annex C), the attributes related to role shall be specified at the bottom of the table with a divider "Attribute related to role", as shown in the following example:

Attribute name	Support qualifier	isReadable	isWriteable	isInvariant	isNotifyable
…					
…					
Attribute related to role					
…					
…					

2.3.a.3 Attribute constraints

This clause presents constraints for the attributes, and one usage is to present the predicates for conditional qualifiers (CM/CO).

This information is provided in a table as shown below.

Name	Definition

This subclause shall state "None." when there is no attribute constraint to define.

2.3.a.4 Notifications

The <Notifications> subclause, for this class, presents one of the following options:

a) *The class defines (and independent from those inherited) the support of a set of notifications that is identical to that defined in clause 2.5. In such case, use "The common notifications defined in clause 2. 5 are valid for this class, without exceptions or additions." as the lone sentence of this clause.*

b) *The class defines (and independent from those inherited) the support of a set of notifications that is a superset of that defined in clause 2.5. In such case, use "The common notifications defined in clause 2.5 are valid for this class. In addition, the following set of notification is also valid." as the lone paragraph of this clause. Then, define the 'additional' notifications in a table. See clause 2.5 for the notification table format.*

c) *The class defines (and independent from those inherited) the support of a set of notifications that is not identical to, nor a superset of, that defined in clause 2.5. In such case, use "The common notifications defined in clause 2. 5 are not valid for this class. The set of notifications defined in the following table is valid." as the lone paragraph of this clause. Specify the set of notifications in a table. See clause 2.5 for the notification table format.*

d) *The class does not define (and independent from those inherited) the support of any notification. In such case, use "There is no notification defined." as the lone sentence of this clause.*

The notifications identified (options a-c above) in this subclause are notifications that can be emitted across the management interface, where the "object class" and "object instance" parameters of the notification header (see Note 2) of these notifications identify an instance of the IOC defined by the encapsulating subclause (i.e., subclause 2.3.a).

The notifications identified (options a-c above) in this subclause may originate from implementation object(s) whose identifier is mapped in the implementation, to the object instance identifier used over the management interface may or may not be the same as that carried in the notification parameters "object class" and "object instance". Hence, the identification of notifications in this subclause does not imply nor identify those notifications as being originated from an instance of the class (or its direct or indirect derived class) defined by the encapsulating subclause (i.e., subclause 2.3.a).

NOTE 1 – This clause shall state "This class does not support any notification." (see option-c) when there is no notification defined for this class. (Note that if its parent class has defined some notifications, the implementation of this class is capable of emitting those inherited defined notifications.)

NOTE 2 – The notification header is defined in the notification integration reference point (IRP) information service [b-3GPP TS 32.302].

NOTE 3 – The qualifier of a notification, specified in Notification Table, indicates if an implementation can generate a notification carrying the DN of the subject class. The qualifier of a notification, specified in a management specification, indicates if an implementation of the management specification can generate such notification in general.

A Manager can receive notification-XYZ that carries DN (the "object class" and "object instance") of class-ABC instance if and only if:

1) *The class-ABC Notification Table defines the notification-XYZ and*
2) *The class-ABC instance implementation supports this notification-XYZ and*
3) *A management interface defines the notification-XYZ and*
4) *The management interface implementation supports this notification-XYZ.*

2.3.a.5 State diagram

This subclause contains state diagrams. A state diagram of an information object class defines permitted states of this information object class and the transitions between those states. A state is expressed in terms of individual attribute values or a combination of attribute values or involvement in relationships of the information object class being defined. This shall be a UML compliant state diagram.

This subclause shall state "None." when there is no State diagram defined.

2.4 Attribute definitions

2.4.1 Attribute properties

It has a lone paragraph "The following table defines the properties of attributes that are specified in the present document.".

Each information attribute is defined using the following structure.
Inherited attributes shall not be shown, as they are defined in the parent class(es) and thus valid for this class.
An attribute has properties (Table C.1). Some properties of an attribute are defined in 2.3.a.2 (e.g., Support Qualifier). The remaining properties of an attribute (e.g., documentation, default value) are defined here.
The information is provided in a table. In case a) attributes of the same name are specified in more than one class and b) the attributes have different properties, then the attribute names (first column) should be prefixed with the class name followed by a period.

An example is given below:

Attribute Name	Documentation and Allowed Values	Properties
xyzId	It identifies … allowedValues …	type: Integer multiplicity: … isOrdered: … isUnique: … defaultValue: … isNullable: False

In case there is one or more attributes related to role (see clause 2.10 of Annex C), the attributes related to role shall be specified at the bottom of the table with a divider "Attribute related to role". See example below.

Attribute Name	Documentation and Allowed Values	Properties
abc	It identifies … allowedValues …	type: Integer multiplicity: … isOrdered: … isUnique: … defaultValue: … isNullable: False
Attribute Related to Role		
aEnd	It identifies … allowedValues …	type: DN multiplicity: … isOrdered: … isUnique: … defaultValue: … isNullable: False

This clause shall state "None." if there is no attribute to define.

2.4.2 Constraints

This clause indicates whether there are any constraints affecting attributes. Each constraint is defined by a tuple (propertyName, affected attributes, propertyDefinition). PropertyDefinitions are expressed in natural language.
This information is provided in a table as shown below.

Name	Affected attribute(s)	Definition

This subclause shall state "None." if there is no constraint.

2.5 Common notifications

This clause presents a list of notifications that can be referred to by any class defined in the specification. This information is provided in a table as shown below.

Name	Qualifier	Notes

This subclause shall state "None." if there are no common notifications.

2.5.1 Alarm notifications

The following quoted text shall be copied as the only paragraph of this clause.

*"*This clause presents a list of notifications, defined in [x], that a manager can receive. The notification header attribute objectClass/objectInstance, defined in [y], shall capture the DN of an instance of a class defined in this specification.*"*

The information is provided in a table as shown below.

Name	Qualifier	Notes

2.5.2 Configuration notifications

The following quoted text shall be copied as the only paragraph of this clause.

*"*This clause presents a list of notifications, defined in [x], that IRPManager can receive. The notification header attribute objectClass/objectInstance, defined in [z], shall capture the DN of an instance of a class defined in this specification.*"*

The information is provided in a table as shown below.

Name	Qualifier	Notes

2.6 System state model

Some configurations of information are special or complex enough to justify the usage of a state diagram to clarify them. A state diagram in this subclause defines permitted states of the system and the transitions between those states. A state is expressed in terms of a combination of attribute values constraints or involvement in relationships of one or more information object classes.

3 Interface definition

This clause shall be used for all management interface specifications and optional for information model only specifications.

3.1 Class diagram representing interfaces

Each interface is defined in one or more UML-compliant class diagrams (see also Annex C).

3.2 Generic rules

The following rules are relevant to all specifications. They shall simply be copied as part of the specification.

Rule 1: Each operation with at least one input parameter supports a pre-condition valid_input_parameter which indicates that all input parameters shall be valid with regard to their information type. Additionally, each such operation supports an exception operation_failed_invalid_input_parameter which is raised when pre-condition valid_input_parameter is false. The exception has the same entry and exit state.

Rule 2: Each operation with at least one optional input parameter supports a set of pre-conditions supported_optional_input_parameter_xxx where "xxx" is the name of the optional input parameter and the pre-condition indicates that the operation supports the named optional input parameter. Additionally, each such operation supports an exception operation_failed_unsupported_optional_input_parameter_xxx which is raised when (a) the pre-condition supported_optional_input_parameter_xxx is false and (b) the named optional input parameter is carrying information. The exception has the same entry and exit state.

Rule 3: Each operation shall support a generic exception operation_failed_internal_problem which is raised when an internal problem occurs and that the operation cannot be completed. The exception has the same entry and exit state.

NOTE – Security considerations and resulting generic rules are for further study.

3.b Interface InterfaceName (supportQualifier)

InterfaceName is the name of the interface followed by a qualifier indicating whether the interface is Mandatory (M), Optional (O), Conditional-Mandatory (CM), Conditional-Optional (CO), or SS-Conditional (C) (see also clause B.1).

"b" represents a number, starting at 3 and increasing by 1 with each new definition of an interface.

Each interface is defined by its name and by a sequence of operations or notifications.

If the interface is related to operation(s), the following subclause 3.b.a "Operation OperationName (supportQualifier)" shall be applied.

If the interface is related to notification(s), subclause 3.b.b "Notification NotificationName (supportQualifier)" below shall be applied.

3.b.a Operation OperationName (supportQualifier)

OperationName is the name of the operation followed by a qualifier indicating whether the operation is Mandatory (M), Optional (O), Conditional-Mandatory (CM), Conditional-Optional (CO), or SS-Conditional (C) (see clause B.1).

"a" represents a number, starting at 1 and increasing by 1 with each new definition of an operation.

3.b.a.1 Definition

This subclause is written in natural language.

Information on traceability back to one or more requirements supported by this operation should also be defined here, in the following form:

Reference	Requirements label	Comment

3.b.a.2 Input parameters

List of input parameters of the operation. Each element is a tuple (Parameter Name, Support Qualifier, Information Type (see Annex E and Note in clause E.2) and an optional list of Legal Values supported by the parameter, Comment). Legal values for the Support Qualifier are: Legal Values for the Support Qualifier are: Mandatory (M), Optional (O), Conditional-Mandatory (CM), Conditional-Optional (CO), or SS-Conditional (C) (see also in clause B.1).

This information is provided in a table as shown below.

Parameter Name	Support Qualifier	Information Type/ Legal Values	Comment

NOTE – Information Type qualifies the parameter of Parameter Name. In the case where the Legal Values can be enumerated, each element is a pair (Legal Value Name, Legal Value Semantics), unless a Legal Value Semantics applies to several values in which case the definition is provided only once. When the Legal Values cannot be enumerated, the list of Legal Values is defined by a single definition.

3.b.a.3 Output parameters

List of output parameters of the operation. Each element is a tuple (Parameter Name, Support Qualifier, Matching Information / Information Type (see Annex E and Note in clause E.2) and an optional list of Legal Values supported by the parameter, Comment). Legal values for the Support Qualifier are: Mandatory (M), Optional (O), Conditional-Mandatory (CM), Conditional-Optional (CO), or SS-Conditional (C) (see also clause B.1).

This information is provided in a table as shown below.

Parameter name	Support qualifier	Matching information/ Information type/ Legal values	Comment

NOTE – Information Type qualifies the parameter of Parameter Name. In the case where the Legal Values can be enumerated, each element is a pair (Legal Value Name, Legal Value Semantics), unless a Legal Value Semantics applies to several values, in which case the definition is provided only once. When the Legal Values cannot be enumerated, the list of Legal Values is defined by a single definition.

This table shall also include a special 'parameter status' to indicate the completion status of the operation (success, partial success, failure reason, etc.).

3.b.a.4 Pre-condition

A pre-condition is a collection of assertions joined by AND, OR, and NOT logical operators. The pre-condition must be held to be true before the operation is invoked. An example is given here below:

`notificationCategoriesNotAllSubscribed` *OR* `notificationCategoriesParameterAbsentAndNotAllSubscribed`

Each assertion is defined by a pair (name, definition). All assertions constituting the pre-condition are provided in a table as shown below.

Assertion name	Definition
`notificationCategoriesNotAllSubscribed`	At least one `notificationCategory` identified in the `notificationCategories` input parameter is supported by `IRPAgent` and is not a member of the `ntfNotificationCategorySet` attribute of an `NtfSubscription` which is involved in a subscription relationship with the `NtfSubscriber` identified by the `managerReference` input parameter.
`notificationCategoriesParameterAbsentAndNotAllSubscribed`	The `notificationCategories` input parameter is absent and at least one `notificationCategory` supported by `IRPAgent` is not a member of the `ntfNotificationCategorySet` attribute of an `ntfSsubscription` which is involved in a subscription relationship with the `NtfSubscriber` identified by the `managerReference` input parameter.

3.b.a.5 Post-condition

A post-condition is a collection of assertions joined by AND, OR, and NOT logical operators. The post-condition must be held to be true after the completion of the operation. When nothing is said in a post-condition regarding an information entity, the assumption is that this information entity has not changed compared to what is stated in the pre-condition. An example is given here below:

`subscriptionDeleted` OR `allSubscriptionDeleted`

Each assertion is defined by a pair (name, definition). All assertions constituting the post-condition are provided in a table as shown below.

Assertion name	Definition
`subscriptionDeleted`	The `ntfSubscription` identified by `subscriptionId` input parameter is no more involved in a subscription relationship with the `ntfSubscriber` identified by the `managerReference` input parameter and has been deleted. If this `ntfSubscriber` has no more `ntfSubscription`, it is deleted as well.
`allSubscriptionDeleted`	In the case `subscriptionId` input parameter was absent, the `ntfSubscriber` identified by the `managerReference` input parameter is no longer involved in any subscription relationship and is deleted, the corresponding `ntfSubscription` have been deleted as well.

3.b.a.6 Exceptions

List of exceptions that can be raised by the operation. Each element is a tuple (exceptionName, condition, ReturnedInformation, exitState).

3.b.a.6.c exceptionName

ExceptionName is the name of an exception.

"c" represents a number, starting at 1 and increasing by 1 with each new definition of an exception.

This information is provided in a table as shown below.

Exception name	Definition	
	Condition	
	Return info	
	Exit state	
	Condition	
	Return info	
	Exit state	

3.b.a.7 Constraints

This subclause presents constraints for the operation or its parameters.

NOTE – This subclause does not need to be present when there are no constraints to be defined.

3.b.b Notification NotificationName (supportQualifier)

NotificationName is the name of the notification followed by a qualifier indicating whether the notification is Mandatory (M), Optional (O), Conditional-Mandatory (CM), Conditional-Optional (CO) or SS-Conditional (C) (see clause B.1).

"b" represents a number, starting at 1 and increasing by 1 with each new definition of a notification.

3.b.b.1 Definition

This subclause is written in natural language.

Information on traceability back to one or more requirements supported by this notification should also be defined here, in the following form:

Reference	Requirement label	Comment

3.b.b.2 Input parameters

List of input parameters of the notification. Each element is a tuple (Parameter Name, Qualifiers, Matching Information/Information Type (see Annex E and Note in clause E.2) and an optional list of Legal Values supported by the parameter, Comment).

The column "Qualifiers" contains the two qualifiers, Support Qualifier (see clause B.1) and Filtering Qualifier, separated by a comma. The Filtering Qualifier indicates whether the parameter of the notification can be filtered or not. Values are Yes (Y) or No (N).

This information is provided in a table as shown below.

Parameter name	Qualifiers	Matching information/ Information type/ Legal values	Comment
alarmType	M,Y	AlarmInformation.eventType / ENUMERATED / "Communications Alarm": a communication error alarm. "Processing Error Alarm": a processing error alarm. "Environmental Alarm": an environmental violation alarm. "Quality Of Service Alarm": a quality of service violation alarm. "Equipment Alarm": an alarm related to equipment malfunction.	

NOTE – Information Type qualifies the parameter of Parameter Name. In the case where the Legal Values can be enumerated, each element is a pair (Legal Value Name, Legal Value Semantics), unless a Legal Value Semantics applies to several values, in which case the definition is provided only once. When the Legal Values cannot be enumerated, the list of Legal Values is defined by a single definition.

3.b.b.3 Triggering event

The triggering event for the notification to be sent is the transition from the information state defined by the "from state" subclause to the information state defined by the "to state" subclause.

3.b.b.3.1 From state

This subclause is a collection of assertions joined by AND, OR, and NOT logical operators. An example is given here below:

alarmMatched AND alarmInformationNotCleared

Each assertion is defined by a pair (name, definition). All assertions constituting the state "from state" are provided in a table as shown below.

Assertion name	Definition

3.b.b.3.2 To state

This subclause is a collection of assertions joined by AND, OR and NOT logical operators. When nothing is said in a to-state regarding an information entity, the assumption is that this information entity has not changed compared to what is stated in the from state.

Each assertion is defined by a pair (name, definition). All assertions constituting the state "to state" are provided in a table as shown below.

Assertion name	Definition

3.b.b.4 Constraints

This subclause presents constraints for the notification or its parameters.

NOTE – This subclause does not need to be present when there are no constraints to be defined.

3.c Scenario

This subclause contains one or more sequence diagrams, each describing a possible scenario. These shall be UML-compliant sequence diagrams. This is an optional subclause.

B.3 IOC properties and inheritance

B.3.1 Property

The following guidelines are based on Annex G of [b-3GPP TS 32.150].

The properties of an IOC (excluding Support IOC) are specified in terms of the following:

a) An IOC attribute(s) including its semantics and syntax, its legal value ranges and support qualifications. The IOC attributes are not restricted to Configuration Management but also include those related to, for example, 1) Performance Management (i.e., measurement types), 2) Trace Management and 3) Accounting Management.

b) The non-attribute-specific behaviour associated with an IOC (see Note 1).

NOTE 1 – As an example, the Link between A and B is optional. It is mandatory if the A instance belongs to one ManagedElement instance while the B instance belongs to another ManagedElement instance. This Link behaviour is a non-attribute-specific behaviour. It is expected that this behaviour, like others, will be inherited.

c) An IOC relationship(s) with another IOC(s).

d) An IOC notification type(s) and their qualifications.

e) An IOC's relation with its parents (see Note 2). There are three mutually exclusive cases:

1) The IOC is abstract and no parents have yet been designated.
2) The IOC is abstract and all of the possible parent(s) have been designated and whether subclass IOCs can be designated as a root IOC.
3) The IOC is not abstract and all of the possible parent(s) have been designated and whether the IOC can be designated as a root IOC.

An IOC instance is either a root IOC or it has one and only one parent.

NOTE 2 – The parent and child relation in this subclause is the parent name-containing the child relation.

f) An IOC's relation with its children. There are three mutually exclusive cases:

1) An IOC shall not have any children (name-containment relation) IOCs.
2) An IOC can have children IOC(s). The maximum number of instances per children IOC can be specified. An IOC may designate that vendor-specific objects are not allowed as children IOCs.
3) An IOC can only have the specific children IOC(s) (or their subclasses). The maximum number of instances per children IOC can be specified. An IOC may designate that vendor-specific objects are not allowed as children IOCs.

g) Whether An IOC can be instantiated or not (i.e., whether an IOC is an abstract IOC).

h) An attribute for naming purpose.

B.3.2 Inheritance

The following guidelines are based on Annex G of [b-3GPP TS 32.150].

An IOC (the subclass) inherits from another IOC (the superclass) in that the subclass shall have all the properties of the superclass.

The subclass can change the inherited support-qualification(s) from optional to mandatory but not vice versa. The subclass can change the inherited support-qualification from conditional-optional to conditional-mandatory but not vice versa.

An IOC can be a superclass of many IOC(s). A subclass cannot have more than one superclass.

The subclass can:

a) Add (compared to those of its superclass) unique attributes including their behaviour, legal value ranges and support-qualifications. Each additional attribute shall have its own unique attribute name (among all added and inherited attributes).

b) Add non-attribute behaviour on an IOC basis. This behaviour may not contradict inherited superclass behaviour.

c) Add relationship(s) with IOC(s). Each additional relationship shall have its own unique name (among all added and inherited relations).

d) Add additional notification types and their qualifications.

e) Designate all of the possible parent(s) (and their subclasses) if the superclass has Property-e-1 such that an IOC will have Property-e-2 or Property-e-3. Restrict possible parent(s) (and their subclasses) and/or remove the capability of the subclass from being a root IOC, if the superclass has Property-e-2 or Property-e-3.

f) Add children IOC(s) if the superclass has Property-f-2 such that an IOC will have Property-f-3. Restrict the allowed children IOC(s) (or their subclasses) if the superclass has Property-f-3.

g) Specify whether an IOC can be instantiated or not (i.e., the IOC is an abstract IOC).

h) Restrict the legal value range of a superclass attribute that has a legal value range.

B.3.3 Import

The following guidelines are based on Annex I of [b-3GPP TS 32.150].

To facilitate re-use of entity definitions among IRP specifications, an import mechanism is used. When an IRP specification (the subject IRP specification) imports an entity defined in another IRP specification, the subject IRP specification is considered to have defined the imported entity in its specification. Furthermore, the subject IRP specification cannot change the properties of this imported entity. If it requires an entity that is not identical but similar to the imported entity, it should define a new entity that inherits the imported entity and introduce changes in the new entity definition.

Annex C

MISM UML repertoire

(This annex forms an integral part of this Recommendation.)

The following are guidelines for specification of the results of the analysis phase as based on 3GPP unified modelling language (UML) repertoire [b-3GPP TS 32.156].

C.1 Introduction

UML provides a rich set of concepts, notations and model elements to model distributed systems. Usage of all UML notations and model elements is not necessary for the purpose of analysis specifications. This annex documents the necessary and sufficient set of UML notations and model elements, including the ones built by the UML extension mechanism <<stereotype>>, for use by development of protocol-neutral specifications. Collectively, this set of notations and model elements is called the UML modelling repertoire.

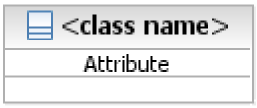
Recommendations following the methodology shall employ the UML notation and model elements of this repertoire and may also employ other UML notation and model elements considered necessary.

C.2 Basic model elements

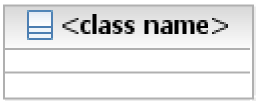
C.2.1 General

UML has defined a number of basic model elements. This clause lists the selected subset for use in specifications based on the repertoire. The semantics of these selected basic model elements are defined in [OMG UML1].
For each basic model element listed, there are three parts. The first part contains its description. The second part contains its graphical notation examples and the third part contains the rule, if any, recommended for labelling or naming it.
The graphical notation has the following characteristics:

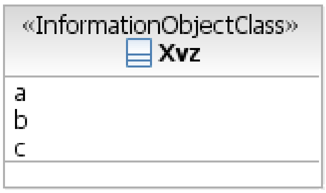
a) Subclause 7.2.7 of [OMG UML2] specifies "A class is often shown with three compartments. The middle compartment holds a list of attributes while the bottom compartment holds a list of operations" and "Additional compartments may be supplied to show other details". This repertoire only allows the use of the name (top) compartment and attribute (middle) compartment. The operation (bottom) compartment may be present but is always empty, as shown in the figure below.



b) Classes may or may not have attributes. The graphical notation of a class may show an empty attribute (middle) compartment even if the class has attributes, as shown in figure below.



c) The visibility symbol shall not appear along with the class attribute, as shown below.



d) The use of the decoration, i.e., the symbol in the name (top) compartment, is optional.

C.2.2 Attribute

C.2.2.1 Description

It is a typed element representing a property of a class. See 10.2.5 Property of [OMG-UML1].

An element that is typed implies that the element can only refer to a constrained set of values.

See 10.1.4 Type of [OMS-UML1] for more information on type.

See clauses C.3.5 and C.4.3 for predefined data types and user-defined data types that can apply type information to an element.

Table C.1 captures the properties of this modelled element.

Table C.1 – Attribute properties

Property name	Description	Legal values
documentation	Contains a textual description of the attribute. Should refer (to enable traceability) to the specific requirement.	Any
isOrdered	For a multi-valued multiplicity; this specifies if the values of this attribute instance are sequentially ordered. See section 7.3.44 and its Table 7.1 of [OMG-UML2].	True, False (default)
isUnique	For a multi-valued multiplicity, this specifies if the values of this attribute instance are unique (i.e., no duplicate attribute values). See section 7.3.44 and its Table 7.1 of [OMG-UML2].	True (default), False
isReadable	Specifies that this attribute can be read by the manager.	True (default), False
isWritable	Specifies that this attribute can be written by the manager under the conditions specified in Annex G.	True, False (default)
type	Refers to a predefined (see clause C.4.3) or user defined data type (see clause C.3.5. See also section 7.3.44 of [OMG-UML2]; inherited from StructuralFeature.	NA
isInvariant	Attribute value is set at object creation time and cannot be changed under the conditions specified in Annex G.	True, False (default)
allowedValues	Identifies the values the attribute can have.	Dependent on type
isNotifyable	Identifies if a notification shall be sent in case of a value change.[1,2]	True (default), False
defaultValue	Identifies a value at specification time that is used at object creation time under conditions defined in Annex G.	No value (default) or a value that is dependent on allowedValues

Table C.1 – Attribute properties

Property name	Description	Legal values
multiplicity	Defines the number of values the attribute can simultaneously have. See section 7.3.44 of [OMG-UML2]; inherited from StructuralFeature.	See clause C.2.9 Default is 1
isNullable	Identifies if an attribute can carry no information. The implied meaning of carrying "no information" is context sensitive and is not defined in this Model Repertoire.	True, False (default)
supportQualifier	Identifies the required support of the attribute. See also section 7.	M, O (default), CM, CO, C

NOTE 1 – Whether a client/manager can receive the notification depends on a) if the client/manager has subscribed or registered for reception of such notification and b) if a notification mechanism is supported.

NOTE 2 – If the attribute is a role-attribute and its property passedById is 'False', then changes in the navigable association target end instance alone shall not trigger a notification.

C.2.2.2 Example

This example shows three attributes, i.e., `a`, `b` and `c`, listed in the attribute (the second) compartment of the class `Xyz`.

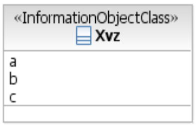


Figure C.1 – Attribute notation

C.2.2.3 Name style

An attribute name shall use the LCC style.

Well known abbreviation (WKA) is treated as a word if used in a name. However, WKA shall be used as is (its letter case cannot be changed) except when it is the first word of a name; and if so, its first letter must be in lower case.

C.2.3 Association relationship

C.2.3.1 Description

It shows a relationship between two classes and describes the reasons for the relationship and the rules that might govern that relationship.

It has ends. Its end, the association end(s), specifies the role that the object at one end of a relationship performs. Each end of a relationship has properties that specify the role (see clause C.2.10), multiplicity (see clause C.2.9), visibility and navigability (see the arrow symbol used in Figure C.3 – Unidirectional association relationship notation) and may have constraints. Note that visibility shall not be used in models based on this Repertoire (see paragraph 3 of clause C.2.1).

See 7.3.3 Association of [OMG-UML2].

The three examples given in Figures C.2 to C.4 show a binary association between two model elements. The association can include the possibility of relating a model element to itself.

The first example (Figure C.2) shows a bidirectional navigable association in that each model element has a pointer to the other. The second example (Figure C.3) shows a unidirectional association (shown with an open arrow at the target model element end) in that only the source model element has a pointer to the target model element and not vice-versa. The third example (Figure C.4) shows a bidirectional non-navigable association in that each model element does not have a pointer to the other; i.e., such associations are just for illustration purposes.

C.2.3.2 Example

An association shall have an indication of cardinality (see clause C.2.9).

It shall, except the case of non-navigable association, have an indication of the role name (see clause C.2.10). The model element involved in an association is said to be "playing a role" in that association. The role has a name such as `+class3` in the first example below. Note that the "+" character in front of the role name, indicating the visibility, is ignored.

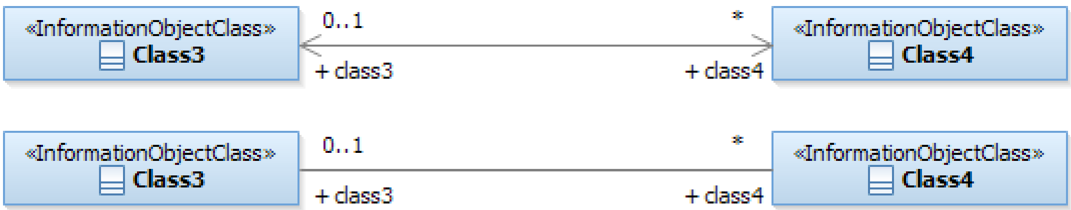


Figure C.2 – Bidirectional association relationship notation

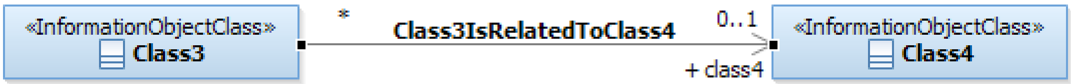


Figure C.3 – Unidirectional association relationship notation

Figure C.4 – Non-navigable association relationship notation

Note that some tools do not use arrows in the UML graphical representation for bidirectional associations. Therefore, absence of arrows is not, but absence of role names is, an indication of a non-navigable association.

C.2.3.3 Name style

An Association can have a name. The use of Association name is optional. Its name style is UCC style.

A role name shall use the LCC style.

C.2.4 Aggregation association relationship

C.2.4.1 Description

It shows a class as a part of or subordinate to another class.

An aggregation is a special type of association in which objects are assembled or configured together to create a more complex object. Aggregation protects the integrity of an assembly of objects by defining a single point of control called aggregate, in the object that represents the assembly.

See 7.3.2 AggregationKind (from Kernel) of [OMG-UML2].

C.2.4.2 Example

Figure C.5 shows that a hollow diamond attached to the end of a relationship is used to indicate an aggregation. The diamond is attached to the class that is the aggregate. The aggregation association shall have an indication of cardinality at each end of the relationship (see clause C.2.9).

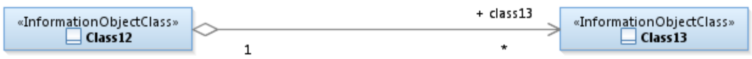


Figure C.5 – Aggregation association relationship notation

C.2.4.3 Name style

An Association can have a name. Use of Association name is optional. Its name style is UCC.

C.2.5 Composite aggregation association relationship

C.2.5.1 Description

A composite aggregation association is a strong form of aggregation that requires a part instance be included in at most one composite at a time. If a composite is deleted, all of its parts are deleted as well.

A composite aggregation shall contain a description of its use.

See 7.3.3 Association (from Kernel) of [OMG-UML2].

C.2.5.2 Example

A filled diamond attached to the end of a relationship (see Figure C.6) is used to indicate a composite aggregation. The diamond is attached to the class that is the composite. The composition association shall have an indication of cardinality at each end of the relationship (see clause C.2.9).

Figure C.6 – Composite aggregation association relationship notation

C.2.5.3 Name style

An Association can have a name. The use of Association name is optional. Its name style is UCC.

C.2.6 Generalization relationship

C.2.6.1 Description

Generalization indicates a relationship in which one class (the child) inherits from another class (the parent).

See 7.3.20 Generalization of [OMG-UML2].

C.2.6.2 Example

The example in Figure C.7 shows a generalization relationship between a more general model element (the `IRPAgent`) and a more specific model element (the `IRPAgentVendorA`) that is fully consistent with the first element and that adds additional information.

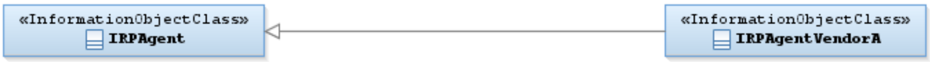


Figure C.7 – Generalization relationship notation

C.2.6.3 Name style

Generalization has no name, so there is no name style.

C.2.7 Dependency relationship

C.2.7.1 Description

A dependency is a relationship that signifies that a single or a set of model elements requires other model elements for their specification or implementation. This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s)...", an extract from 7.3.12 Dependency of [OMG-UML2].

C.2.7.2 Example

The example in Figure C.8 shows that the `BClass` instances have a semantic relationship with the `AClass` instances. It indicates a situation in which a change to the target element (the `AClass` in the example) will require a change to the source element (the `BClass` in the example) in the dependency.

Figure C.8 – Dependency relationship notation

C.2.7.3 Name style

A Dependency can have a name. Use of Dependency name is optional. Its name style is UCC.

C.2.8 Comment

C.2.8.1 Description

A comment is a textual annotation that can be attached to a set of elements.

See 7.3.9 Comment (from Kernel) from [OMG-UML2].

C.2.8.2 Example

The example in Figure C.9 shows a comment as a rectangle with a "bent corner" in the upper right corner. It contains text. It appears on a particular diagram and may be attached to zero or more modelling elements by dashed lines.

Figure C.9 – Comment notation

C.2.8.3 Name style

Comment notations have no name so there is no name style.

C.2.9 Multiplicity (also known as cardinality in relationships)

C.2.9.1 Description

"A multiplicity is a definition of an inclusive interval of non-negative integers beginning with a lower bound and ending with a (possibly infinite) upper bound. A multiplicity element embeds this information to specify the allowable cardinalities for an instantiation of this element…", an extract from 7.3.32 MultiplicityElement of [OMG-UML2].

Table C.2 – Multiplicity-string definitions

Multiplicity	Explanation
1	Attribute has one attribute value
m	Attribute has *m* attribute values
0..1	Attribute has zero or one attribute value
0..*	Attribute has zero or more attribute values
*	Attribute has zero or more attribute values
1..*	Attribute has at least one attribute value
m..n	Attribute has at least *m* but no more than *n* attribute values

The use of "0..n" and "0..*" is not recommended although it has the same meaning as "*".

The use of a standalone symbol zero (0) is not allowed.

C.2.9.2 Example

Figure C.10 shows a multiplicity attached to the end of an association path. The meaning of this multiplicity is one to many. One `Network` instance is associated with zero, one or more `SubNetwork` instances. Other valid examples can show the "many to many" relationship.

Figure C.10 – Cardinality notation

The cardinality zero is not used to indicate the IOC's so-called "transient state" characteristic. For example, it is not used to indicate that the instance is not yet created but it is in the process of being created. The cardinality zero will not be used to indicate this characteristic since such characteristic is considered inherent in all IOCs. All IOCs defined are considered to have such inherent "transient state" characteristics.

Note that the use of "0..*", "0..n" or '*' means "zero to many". The use of "0..*" is recommended. The following table shows some valid examples of multiplicity.

Table C.3 – Multiplicity-string examples

Multiplicity	Explanation
1	Attribute has exactly one attribute value
5	Attribute has exactly 5 attribute values
0..1	Attribute has zero or one attribute value
0..*	Attribute has zero or more attribute values
1..*	Attribute has at least one attribute value
4..12	Attribute has at least 4 but no more than 12 attribute values

C.2.9.3 Name style

Cardinality has no name so there is no name style.

C.2.10 Role

C.2.10.1 Description

A role indicates navigation, from one class to another class, involved in an association relationship. A role is named. The direction of navigation is to the class attached to the end of the association relationship with (or near) the role name.

The use of role name in the graphical representation is mandatory for bidirectional and unidirectional association relationship notations (see Figure C.2 – Bidirectional association relationship notation and Figure C.3 – Unidirectional association relationship notation). Role name shall not be used in non-navigable association relationship notation (see Figure C.4 – Non-navigable association relationship notation).

A role at the navigable end of a relationship becomes (or is mapped into) an attribute (called role-attribute) in the source class of the relationship. Therefore roles have the same behaviour (or properties) as attributes.

The role-attribute shall have all properties defined for attributes in clause C.2.2 and in addition the following property:

Table C.4 – passedById property

Property name	Description	Legal values
passedById	If True, the role-attribute (navigable association source end) contains a DN of the navigable association target end instance. If False, the role-attribute contains (a copy of) the whole target end instance (e.g., X). If X has a role-attribute whose "passedById==False", then the subject role-attribute contains (a copy of) X's target end instance as well. The above rule is applied repeatedly for all occurrences of "passedById==False". This application can result in a collection of instances where no ordering can be implied and no instances are duplicated. Use of "passedById==False" supports the efficient access of target end instances from a source end instance. The mechanism by which such access is achieved is operation model design specific (e.g., not related to resource model design).	True (default), False

C.2.10.2 Example

The example in Figure C.11 shows that a `Person` (say instance John) is associated with a `Company` (say whose DN is "Company=XYZ"). We navigate the association by using the opposite association-end such that John's `Person.theCompany` would hold the DN, i.e., "Company=XYZ".

Figure C.11 – Role notation

C.2.10.3 Name style

A role has a name. Use noun for the name. The name style follows the attribute name style; see clause C.2.2.3.

C.2.11 Xor constraint

C.2.11.1 Description

"A Constraint represents additional semantic information attached to the constrained elements. A constraint is an assertion that indicates a restriction that must be satisfied by a correct design of the system. The constrained elements are those elements required to evaluate the constraint specification…", an extract from 7.3.10 Constraint (from Kernel) of [OMG-UML2].

For a constraint that applies to two elements such as two associations, the constraint shall be shown as a dashed line between the elements labeled by the constraint string (in braces). The constraint string, in this case, is xor.

C.2.11.2 Example

Figure C.12 shows a `ServerObjectClass` instance that has relation(s) to multiple instances of a class from the choice of `ClientObjectCLass_Alternative1`, `ClientObjectClass_Alternative2` or `ClinetObjectCLass_Alternative3`.

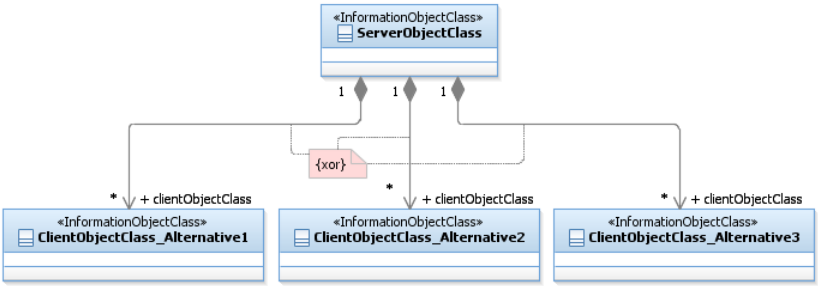


Figure C.12 – {xor} notation

C.2.11.3 Name style

The Xor constraint has no name so there is no name style.

C.3 Stereotypes

C.3.1 General

Clause C.2 lists the UML defined basic model elements. UML defined a stereotype concept allowing the specification of simple or complex user-defined model elements.

This clause lists all allowable stereotypes for this repertoire.

For each stereotype model element listed, there are three parts. The first part contains its description. The second part contains its graphical notation examples and the third part contains the rule, if any, recommended for labelling or naming it.

C.3.2 <<ProxyClass>>

C.3.2.1 Description

This represents a number of <<InformationObjectClass>>. It encapsulates attributes, links, methods (or operations), and interactions that are present in the represented <<InformationObjectClass>>.

The semantics of a <<ProxyClass>> is that all behaviour of the <<ProxyClass>> are present in the represented <<InformationObjectClass>>. Since this class is simply a representation of other classes, this class cannot define its own behaviour other than those already defined by the represented <<InformationObjectClass>>.

A particular <<InformationObjectClass>> can be represented by zero, one or more <<ProxyClass>>. For example, the ManagedElement <<InformationObjectClass>> can have MonitoredEntity <<ProxyClass>> and ManagedEntity <<ProxyClass>>.

The attributes of the <<ProxyClass>> are accessible by the source entity that has an association with the <<ProxyClass>>.

C.3.2.2 Example

Figure C.13 shows a <<ProxyClass>> named `MonitoredEntity`. It represents (or its constraints is that it represents) all NRM <<InformationObjectClass>> (e.g., `GgsnFunction` <<InformationObjectClass>>) whose instances are being monitored for alarm conditions. It is mandatory to use a Note to capture the constraint.

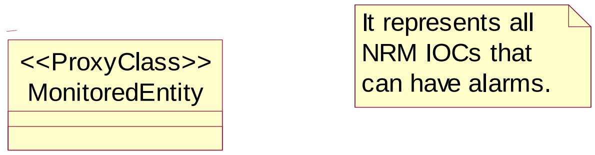


Figure C.13 – <<ProxyClass>> Notation

See Appendix II for more examples that use <<ProxyClass>>.

C.3.2.3 Name style

For <<ProxyClass>> name, use the same style as <<InformationObjectClass>> (see clause C.3.3).

C.3.3 <<InformationObjectClass>>

C.3.3.1 Description

The <<InformationObjectClass>> is identical to UML *class* except that it does not include/define methods or operations.

A UML *class* represents a capability or concept within the system being modelled. Classes have data structure and behaviour and relationships to other elements.

This class can inherit from zero, one or multiple classes (multiple inheritances).

See more on UML *class* in 10.2.1 of [OMG-UML1].

C.3.3.2 Example

The example in Figure C.14 shows an `AbcFunction` <<InformationObjectClass>>.

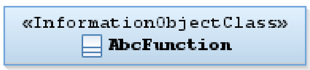


Figure C.14 – <<InformationObjectClass>> Notation

Table C.5 captures the properties of this modelled element.

Table C.5 – <<InformationObjectClass>> properties

Property name	Description	Legal values
documentation	Contains a textual description of this modelled element. Should refer (to enable traceability) to a specific requirement.	Any
isAbstract	Indicates if the class can be instantiated or is just used for inheritance.	True, False (default)
isNotifyable	Identifies the list of the supported notifications.	List of names of notification
supportQualifier	Identifies the required support of the class. See also clause 7.	M, O (default), CM, CO, C

C.3.4 <<names>>

C.3.4.1 Description

The <<names>> is modelled by a composition association where both ends are non-navigable. The source class is the composition and the target class is the component. The target instance is uniquely identifiable, within the namespace of the source entity, among all other targeted instances of the same target class and among other targeted instances of other classes that have the same <<names>> composition with the source.

The source class and target class shall each has its own naming attribute.

The composition aggregation association relationship is used as the act of name containment providing a semantic of a whole-part relationship between the domain and the named elements that are contained, even if only by name. From the management perspective access to the part is through the whole. Multiplicity shall be indicated at both ends of the relationship.

A target instance cannot have multiple <<names>> with multiple sources, i.e., a target instance cannot participate in or belong to multiple namespaces.

C.3.4.2 Example

Figure C.15 shows that all instances of `Class4` are uniquely identifiable within a `Class3` instance's namespace.

Figure C.15 – <<names>> notation

C.3.4.3 Name style

<<names>> has no name so there is no name style.

C.3.5 <<dataType>>

C.3.5.1 Description

<<dataType>> represents the general notion of being a data type (i.e., a type whose instances are identified only by their values) whose definition is defined by user (e.g., specification authors).

This repertoire uses two kinds of data types: predefined data types and user-defined data types. The former is defined in clause C.4.3. The latter is defined by the specifications authors using this <<dataType>> model element.

The user-defined data types support the modelling of structured data types (see <<dataType>> notations in clause C.3.5.3). When user-defined or predefined data type is used to apply type information to a class attribute (see clause C.2.2), the data type name is shown along with the class attribute. See user example of <<dataType>> in clause C.3.5.3

C.3.5.2 Example

The following examples are two user-defined data types. The left-most is named `PlmnId` that consists of mobile country code (MCC) and mobile network code (MNC), whose types are the predefined data types in clause C.4.3. The right-most is named `Xyz` that consists of two predefined data types (i.e., `String`, `Integer` and one user-defined data type `PlmnId`.

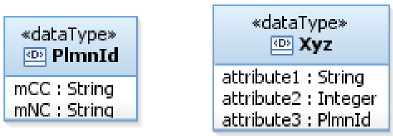


Figure C.16 – <<dataType>> notations

Figure C.17 shows an example of a `ZClass` using two user-defined data types and two predefined data types.

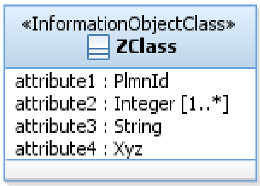


Figure C.17 – Usage example of <<dataType>>

C.3.5.3 Name style

For <<dataType>> name, use the same style as <<InformationObjectClass>> (see clause C.3.3).

For <<dataType>> attribute, use the same style as Attribute (see clause C.2.2).

C.3.6 <<enumeration>>

C.3.6.1 Description

An enumeration is a data type. It contains sets of named literals that represent the values of the enumeration. An enumeration has a name.

See 10.3.2 Enumeration of [OMG-UML1].

C.3.6.2 Example

The example in Figure C.18 shows an enumeration model element whose name is `Account` and it has four enumeration literals. The upper compartment contains the keyword <<enumeration>> and the name of the enumeration. The lower compartment contains a list of enumeration literals.

Note that the symbol to the right of <<enumeration>> Account in the figure below is a feature specific to a particular modelling tool. It is recommended that modelling tool features should be used when appropriate.

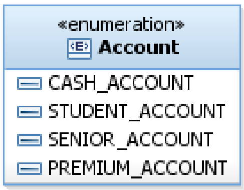


Figure C.18 – <<enumeration>> notation

C.3.6.3 Name style

For <<enumeration>> name, use the same style as <<InformationObjectClass>> (see clause C.3.3).

For <<enumeration>> attribute (the enumeration literal), use the following rules:

- Enumeration literal is composed of one or more words of upper case characters. Words are separated by the underscore character.

C.3.7 <<choice>>

C.3.7.1 Description

The «choice» stereotype represents one of a set of classes (when used as an information model element) or one of a set of data types (when used as an operations model element).

This stereotype property, e.g., one out of a set of possible alternatives, is identical to the {xor} constraint (see clause C.2.11).

C.3.7.2 Example

Sometimes the specific kind of class cannot be determined at model specification time. In order to support such scenario, the specification is done by listing all possible classes.

Figure C.19 lists 3 possible classes. It also shows a «choice, InformationObjectClass» named SubstituteObjectClass. This scenario indicates that only one of the three «InformationObjectClass» named Alternative1ObjectClass, Alternative2ObjectClass, Alternative3ObjectClass shall be realised.

The «choice» stereotype represents one of a set of classes when used as an information model element.

Figure C.19 – Information model element example using «choice» notation

Sometimes the specific kind of data type cannot be determined at model specification time. In order to support such scenario, the specification is done by listing all possible data types.

Figure C.20 lists 2 possible data types. It also shows a «choice» named ProbableCause. This scenario indicates that only one of the two «dataType» named IntegerProbableCause, StringProbableCause shall be realised.

The «choice» stereotype represents one of a set of data types when used as an operations model element.

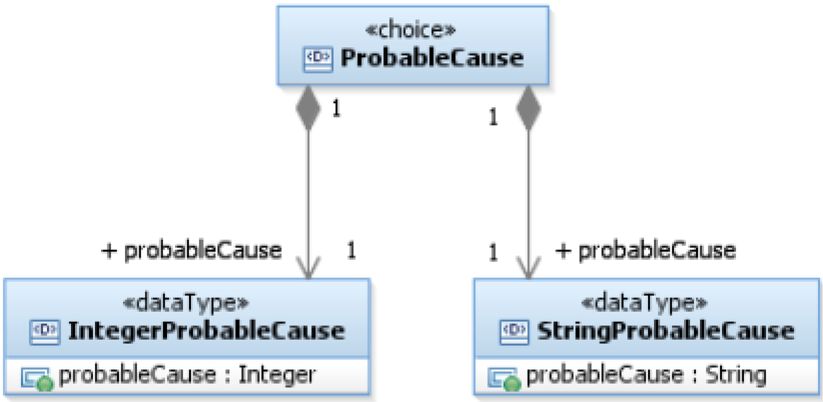


Figure C.20 – Operations model element example using «choice» notation

Sometimes models distinguish between sink/source/bidirectional termination points. A generic class which comprises these three specific classes can be modelled using the «choice» stereotype (see Figure C.21).

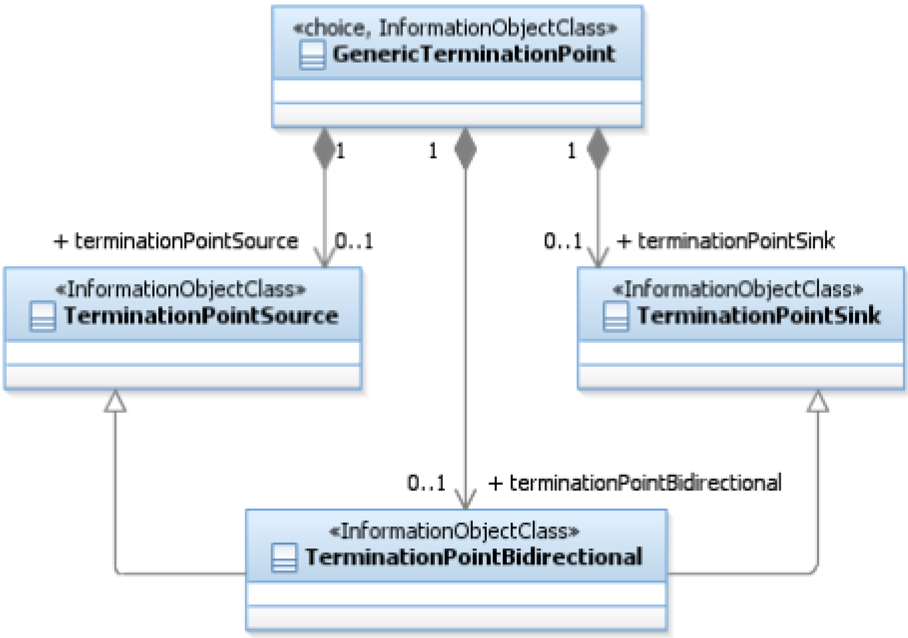


Figure C.21 – Sink/source/bidirectional termination points example using «choice» notation

C.7.3 Name style

For <<choice>> name, use the same style as <<InformationObjectClass>> (see clause C.3.3).

C.4 Others

C.4.1 Association class

C.4.1.1 Description

An association class is an association that also has class properties (or a class that has association properties). Even though it is drawn as an association and a class, it is really just a single model element.

See 7.3.4 AssociationClass of [OMG-UML2].

Association classes are appropriate for use when an «InformationObjectClass» needs to maintain associations to several other instances of «InformationObjectClass» and there are relationships between the members of the associations within the scope of the "containing" «InformationObjectClass». For example, a namespace maintains a set of bindings, a binding ties a name to an identifier. A NameBinding «InformationObjectClass» can be modelled as an Association Class that provides the binding semantics to the relationship between an identifier and some other «InformationObjectClass» such as Object in the figure. This is depicted in Figure C.22.

C.4.1.2 Example

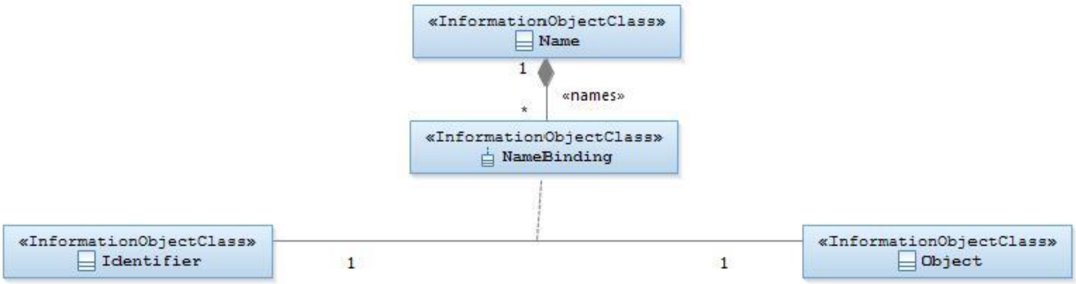


Figure C.22 – Association class notation

C.4.1.3 Name style

The name shall use the same style as in <<InformationObjectClass>> (see clause C.3.3).

C.4.2 Abstract class

C.4.2.1 Description

Abstract class specifies a special kind of <<InformationObjectClass>> as the general model element involved in a generalization relationship (see clause C.2.6). An abstract class cannot be instantiated.

This modelled element has the same properties as class. See clause C.3.3.

C.4.2.2 Example

Figure C.23 shows that *`Class5_`* is an abstract class. It is the base class for `SpecialisedClass5`.

Figure C.23 – Abstract class notation

C.4.2.3 Name style

For abstract class name, use the same style as <<InformationObjectClass>> (see clause C.3.3). The name shall be in italics.

C.4.3 Predefined data types

C.4.3.1 Description

It represents the general notion of being a data type (i.e., a type whose instances are identified only by their values) whose definition is defined by this specification and not by the user (e.g., specification authors).

This repertoire uses two kinds of data types: predefined data types and user-defined data types. The latter are defined in clauses C.3.5 and C.3.6.

Table C.6 lists the UML data types selected for use as predefined data type.

Table C.6 – UML defined data types

Name	Description and reference
Boolean	See Boolean type of [ITU-T X.680].
Integer	See Integer type of [ITU-T X.680].
String	See PrintableString type of [ITU-T X.680].

Table C.7 lists data types that are defined by this repertoire.

Table C.7 – Non-UML defined data types

Name	Description and reference
AttributeValuePair	This data type defines an attribute name and the attribute's value.
BitString	This data type is defined by Bit string of clause 3 and clause G.2.5 of [ITU-T X.680].
DateTime	This data type is defined by GeneralizedTime of [ITU-T X.680].
DN	This data type defines the distinguished name (DN) (see Distinguished Name of [ITU-T X.501].of an object contains a sequence of one or more name components. Each initial sub-sequence (note 1) of the object name is also the name of an object. The sequence of objects so identified, starting with the one identified by only the first name component and ending with the object being named, is such that each is the immediate superior (Note 2) of that which follows it in the sequence. NOTE 1 – Suppose an object's DN is composed of a sequence of 4 name components, i.e., 1st, 2nd, 3rd and 4th components. The "initial sub-sequence" is composed of the 1st, 2nd and 3rd components. NOTE 2 – Suppose object A is name-contained (see clause C.3.4) by object B, object B is said to be the immediate superior of object A.
External	This data type is defined by another organization.
OperationStatusAtomic	This enumeration defines the status values of an atomic operation. • SUCCESSFUL: The operation has been successfully completed as a whole; • NOT_SUCCESSFUL: The operation has not been successfully completed as a whole; i.e., the states of the involved object instances are the same as before the operation (roll back is necessary).
OperationStatusBestEffort	This enumeration defines the status values of a best effort operation. • SUCCESSFUL: The operation has been completed successfully as a whole; • PARTIALLY_SUCCESSFUL: The operation has been completed partially successfully. Further definition what this means for a specific operation is to be specified by the interface specification author; • NOT_SUCCESSFULThe operation has not been completed at all, i.e., the state of the involved object instances is unchanged.
Real	This data type is defined by Real type of [ITU-T X.680].

C.4.3.2 Example

Figure C.24 shows an example of predefined data types usage.

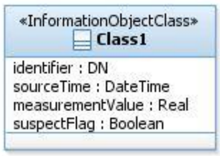


Figure C.24 – Predefined data types usage

NOTE – Use of this is optional. Uses of other means, to specify Predefined data types, are allowed.

C.4.3.3 Name style

It shall use the UCC style.

C.5 Qualifiers

This clause defines the qualifiers applicable for model elements specified in this document, e.g., the IOC (see clause C.3.3), the Attribute (see clause C.2.2). The qualifications are M, O, CM, CO, C and 'SS'. Their meanings are specified in this section. This type of qualifier is called Support Qualifier (see supportQualifier of IOC in Table C.3 and supportQualifier of attribute in Table C.1).

This clause also defines the qualifiers applicable to various properties of a model element, e.g., see the IOC properties excepting 'supportQualifier' in Table C.3 and attributes properties excepting supportQualifier in Table C.1. The qualifications are M, O, CM, CO, C and '-'. Their meanings are specified in this section. This type of qualifier is simply called Qualifier.

Definition of M (Mandatory) qualification:

- The capability (e.g., the Attribute named `abc` of an IOC named `Xyz`; the write property of Attribute named `abc` of an IOC named `Xyz`; the IOC named `Xyz`) shall be supported.

Definition of O (Optional) qualification:

- The capability may or may not be supported.

Definition of CM (Conditional-Mandatory) qualification:

- The capability shall be supported under certain conditions, specifically:
  – When qualified as CM, the capability shall have a corresponding constraint defined in the specification. If the specified constraint is met then the capability shall be supported.

Definition of CO (Conditional-Optional) qualification:

- The capability may be supported under certain conditions, specifically:
  – When qualified as CO, the capability shall have a corresponding constraint defined in the specification. If the specified constraint is met then the capability may be supported.

Definition of C (Conditional) qualification:

- Used for items that have multiple constraints. Each constraint is worded as a condition for one kind of support such as mandatory support, optional support or "no support". All constraints must be related to the same kind of support. Specifically:
  – Each item with C qualification shall have the corresponding multiple constraints defined in the specification. If all specified constraints are met and are related to mandatory, then the item shall be supported. If all the specified constraints are met and are related to optional, then the item may be supported. If all the specified constraints are met and are related to "no support", then the item shall not be supported.
- NOTE – This qualifier should only be used when absolutely necessary, as it is more complex to implement.

Definition of SS (SS Conditional) qualification:

- The capability shall be supported by at least one but not all solutions.

Definition of '-' (no support) qualification:

- The capability shall not be supported.

C.6 UML diagram requirements

Classes and their relationships shall be presented in class diagrams.

It is recommended to create:

- An overview class diagram containing all classes related to a specific management area (Class Diagram).
  - The class name compartment should contain the location of the class definition (e.g., "Qualified Name")
  - The class attributes should show the "Signature". (see section 7.3.45 of [OMG UML2]for the signature definition);
- A separate inheritance class diagram in case the overview diagram would be overloaded when showing the inheritance structure (Inheritance Class Diagram);
- A class diagram containing the user defined data types (Type Definitions Diagram);
- Additional class diagrams to show specific parts of the specification in detail;
- State diagrams for complex state attributes.

Annex D

Design

(This annex forms an integral part of this Recommendation.)

This annex provides guidelines for the specification of protocol-specific designs. It is for further study.

Annex E

Information type definitions – type repertoire

(This annex forms an integral part of this Recommendation.)

This annex defines a repertoire of types that shall be used to specify type information in the conceptual model (analysis model/information service).

The repertoire is defined as a subset of types defined by ASN.1 [ITU-T X.680] combined with types derived from the types defined by ASN.1 (clause E.4).

The keywords to be used for each type are summarized in Table E.1.

E.1 Basic types

Basic types are types that can be used directly to define attributes and parameters. Basic types can also be used to construct complex types. Basic types include the following ASN.1 types:

E.1.1 **integer type** clause 19 of [ITU-T X.680]

E.1.2 **real type** clause 21 of [ITU-T X.680]

E.1.3 **boolean type** clause 18 of [ITU-T X.680]

E.1.4 **bitstring type** clause 22 of [ITU-T X.680]

E.1.5 **null type** clause 24 of [ITU-T X.680]

E.1.6 **generalized time type** clause 38 of [ITU-T X.680]

E.2 Enumerated type

Enumerated type clause 20 of [ITU-T X.680] represents enumerated values. All values that may be used by a specific attribute or parameter shall be listed in the legal value columns. Only the listed names style is applicable for the conceptual model, i.e., the identification of concrete values (numbers or strings) are left for the concrete design models.

NOTE – If the number of these values is more than 50, it is recommended to define them in an appendix or an independent document.

E.3 Complex types

Complex types can be defined using the following concepts:

E.3.1 **sequence types** clause 25 of [ITU-T X.680]

E.3.2 **choice types** clause 29 of [ITU-T X.680]

E.3.3 **set types** clause 27 of [ITU-T X.680]

In addition, lists and sets of complex types are supported using:

E.3.4 **sequence-of types** clause 26 of [ITU-T X.680]

E.3.5 **set-of types** clause 28 of [ITU-T X.680]

E.4 Useful types

E.4.1 String type

String represents a string of characters, the character set is not restricted, i.e.:

String ::= UnrestrictedCharacterStringType clause 44 of [ITU-T X.680]

E.4.2 Name type

Name represents an exclusive name of an object instance in name space. It might include object containment tree hierarchy information, but it is implementation dependent and is out of the scope of this Recommendation. Formally, the name type is defined as:

Name ::= TYPE-IDENTIFIER Annex A of [ITU-T X.681]

E.5 Keywords

Table E.1 defines the list of keywords to be used in the analysis template (see Annex B) for definition of information type, e.g.:

Parameter Name	Support Qualifier	Information Type/Legal Values	Comment
…			
eventIdList	M	SET OF INTEGER/–	The list of alarms to be acknowledged.

Table E.1 – Keywords

Type	Keyword
integer type	INTEGER
real type	REAL
boolean type	BOOLEAN
bitstring type	BIT STRING
null type	NULL
generalized time type	GeneralizedTime
enumerated type	ENUMERATED
sequence type	SEQUENCE
choice type	CHOICE
set type	SET
sequence-of type	SEQUENCE OF
set-of type	SET OF
string type	String
name type	Name

Annex F

Guidelines on IOC properties, inheritance and entity import

(This annex forms an integral part of this Recommendation.)

The following guidelines are based on [b-3GPP TS 32.150].

F.1 IOC property

The properties of an IOC (including Support IOC) are specified in terms of the following:

a) An IOC attribute(s) including its semantics and syntax, its legal value ranges and support qualifications. The IOC attributes are not restricted to Configuration Management but also include those related to, for example, 1) Performance Management (i.e., measurement types), 2) Trace Management and 3) Accounting Management.

b) The non-attribute-specific behaviour associated with an IOC.

NOTE 1 – As an example, the Link between MscServerFunction and CsMgwFunction is optional. It is mandatory if the MscServerFunction instance belongs to one ManagedElement instance while the CsMgwFunction instance belongs to another ManagedElement instance. This Link behaviour is a non-attribute-specific behaviour. It is expected that this behaviour, like others, will be inherited.

c) An IOC relationship(s) with another IOC(s).

d) An IOC notification type(s) and their qualifications.

e) An IOC's relation with its parents (see Note 2). There are three mutually exclusive cases:

1) The IOC can have any parent. In UML diagram, the class has a parent Any.

2) The IOC is abstract and all of the possible parent(s) have been designated and whether subclass IOCs can be designated as a root IOC. In UML diagram, the class has zero or more possible parents of specific classes (except Any).

3) The IOC is concrete and all of the possible parent(s) have been designated and whether the IOC can be designated as a root IOC. In UML diagram, the class has one or more possible parents of specific classes (except Any).

An IOC instance is either a root IOC or it has one and only one parent. Only 3GPP SA5 may designate an IOC class as a potential root IOC. Currently, only SubNetwork, ManagedElement or MeContext IOCs can be root IOCs.

NOTE 2 – The parent and child relation in this subclause is the parent name-containing the child relation.

f) An IOC's relation with its children. There are three mutually exclusive cases:

1) An IOC shall not have any children (name-containment relation) IOCs. In UML diagram, the class has no child.

2) An IOC can have children IOC(s). The maximum number of instances per children IOC can be specified. An IOC may designate that vendor-specific objects are not allowed as children IOCs. In UML diagram, the class has a child Any.

3) An IOC can only have the specific children IOC(s) (or their subclasses). The maximum number of instances per children IOC can be specified. An IOC may designate that vendor-specific objects are not allowed as children IOCs. In UML diagram, the class has one or more children of specific classes (except Any).

g) Whether an IOC can be instantiated or not (i.e., whether an IOC is an abstract IOC).

h) An attribute for naming purpose.

F.2 Inheritance

An IOC (the subclass) inherits from another IOC (the superclass) in that the subclass shall have all the properties of the superclass.

The subclass can change the inherited support-qualification(s) from optional to mandatory but not vice versa. The subclass can change the inherited support-qualification from conditional-optional to conditional-mandatory but not vice versa.

An IOC can be a superclass of many IOC(s). A subclass cannot have more than one superclass.

The subclass can:

a) Add (compared to those of its superclass) unique attributes including their behaviour, legal value ranges and support-qualifications. Each additional attribute shall have its own unique attribute name (among all added and inherited attributes).

b) Add non-attribute behaviour on an IOC basis. This behaviour may not contradict inherited superclass behaviour.

c) Add relationship(s) with IOC(s). Each additional relationship shall have its own unique name (among all added and inherited relations).

d) Add additional notification types and their qualifications.

e) Designate all of the possible parent(s) (and their subclasses) if the superclass has Property-e-1 such that an IOC will have Property-e-2 or Property-e-3. Restrict possible parent(s) (and their subclasses) and/or remove the capability of the subclass from being a root IOC, if the superclass has Property-e-2 or Property-e-3.

f) Add children IOC(s) if the superclass has Property-f-2 such that an IOC will have Property-f-3. Restrict the allowed children IOC(s) (or their subclasses) if the superclass has Property-f-3.

g) Specify whether an IOC can be instantiated or not (i.e., the IOC is an abstract IOC).

h) Restrict the legal value range of a superclass attribute that has a legal value range.

F.3 Entity (interface, IOC and attribute) import

Management interface specifications define entities (e.g., IOCs, interfaces and attribute). To facilitate the reuse of entity definitions among interface specifications, an import mechanism is used. When a management interface specification (the subject specification) imports an entity defined in another management interface specification, the subject specification is considered to have defined the imported entity in its specification. Furthermore, the subject specification cannot change the properties of this imported entity. If it requires an entity that is not identical but similar to the imported entity, it should define a new entity that inherits the imported entity and introduce changes in the new entity definition.

Annex G

Attribute Properties

(This annex forms an integral part of this Recommendation.)

The following guidelines are based on Annex B of [b-3GPP TS 32.156].

isInvariant	write	defaultValue	manager must provide a value when manager requests object creation	Meaning
☑	☑	☑	☑	Not valid.
☑	☑	☑		May be set by the manager only during object creation time; if no value is provided by the manager, the default value is used.
☑	☑		☑	Must be set by the manager during object creation time.
☑	☑			May be set by the manager only during object creation time; if no value is provided by the manager, the agent must provide a value.
☑		☑	☑	Not valid.
☑		☑		Valid but not useful.
☑			☑	Not valid.
☑				Must be set by the agent during object creation time.
	☑	☑	☑	Not valid.
	☑	☑		May be set by the manager anytime; if no value is provided by the manager at object creation time, it is set to the default value.
	☑		☑	Must be set by the manager at object creation time and may be changed anytime.
	☑			May be set by the manager at object creation time and may be changed anytime.
		☑	☑	Not valid.
		☑		Must be set by the agent to the default value at object creation time; may be changed by the agent anytime.
			☑	Not valid.
				May be set by the agent at object creation time and may be changed by the agent anytime.

Annex H

Design patterns

(This annex forms an integral part of this Recommendation.)

The following guidelines are based on Annex C of [b-3GPP TS 32.156].

H.1 Intervening class and association class

H.1.1 Concept and definition

Classes may be related via simple direct associations or via associations with related association classes.

However, in situations where the relationships between a number of classes is complex and especially where the relationships between instances of those classes are themselves interrelated there may be a need to encapsulate the complexity of the relationships within a class that sits between the classes that are to be related. The term "intervening class" is used here to name the pattern that describes this approach. The name "intervening class" is used as the additional class "intervenes" in the relationships between other classes.

The "intervening class" differs from the association class as the intervening class does break the association between the classes whereas the association class does not but instead sits to one side. This can be seen in the Figure H.1. A direct association between class A and C appears the same at A and C regardless of the presence or absence of an association class where as in the case of the "intervening class" there are associations between A and the "intervening class" B and C and the "intervening class" B.

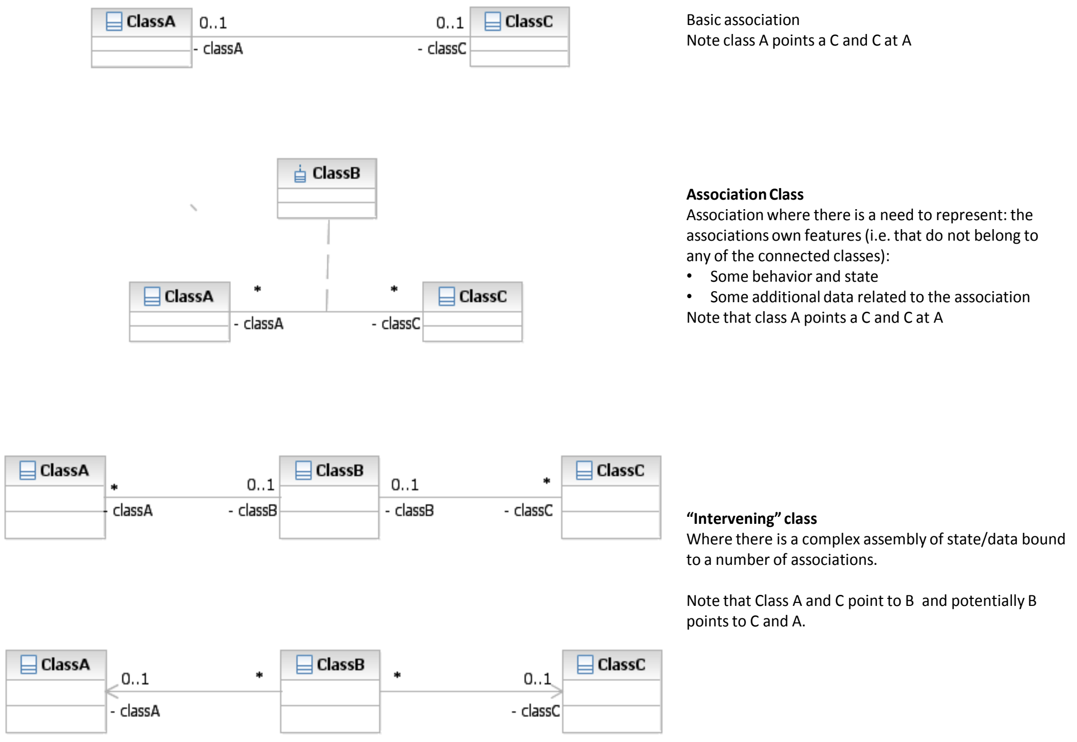


Figure H.1 – Various association forms

The "intervening class" is essentially no different to any other class in that it may encapsulate attributes, complex behaviour etc.

Figure H.2 shows an instance view of both an association class form and an "intervening class" form for a complex interrelationship.

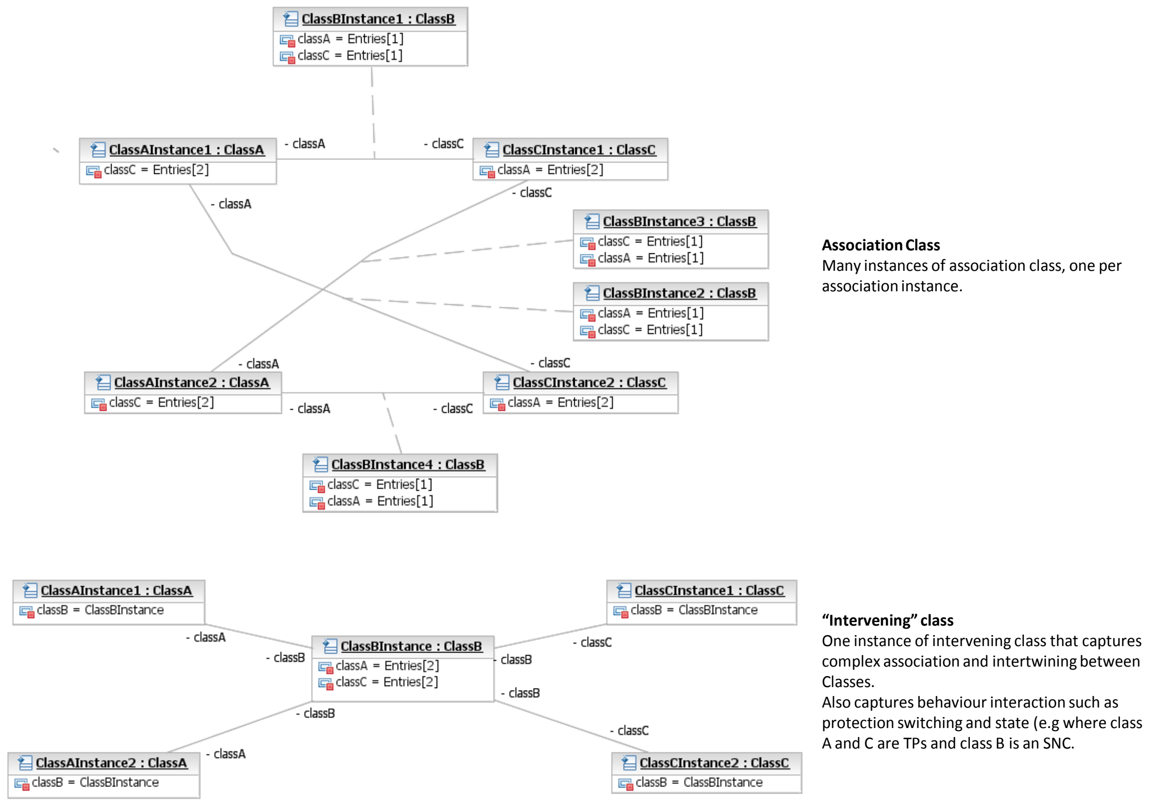


Figure H.2 – Instance view of "intervening class"

The case depicted above does not show interrelationships between the relationships. A practical case from modelling of the relationships between termination points in a fixed network does show this relationship interrelationship challenge. In this case the complexity of relationship is between instances of the same class, the termination point (TP). The complexity is encapsulated in a SubNetworkConnection (SNC) class.

For an example of an SNC intervening in TP-TP relationship see Figure H.3.

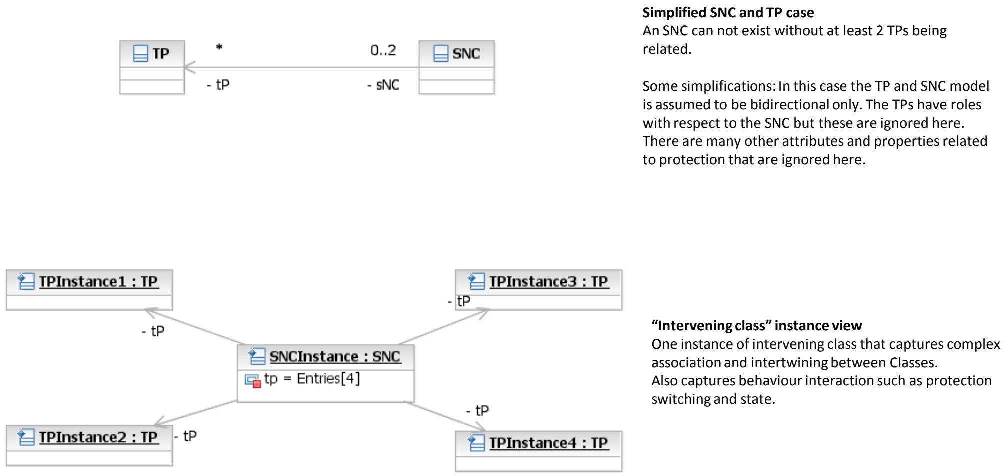


Figure H.3 – SNC intervening in TP-TP relationship

The SNC also encapsulates the complex behaviour of switching and path selection as depicted in Figure H.4.

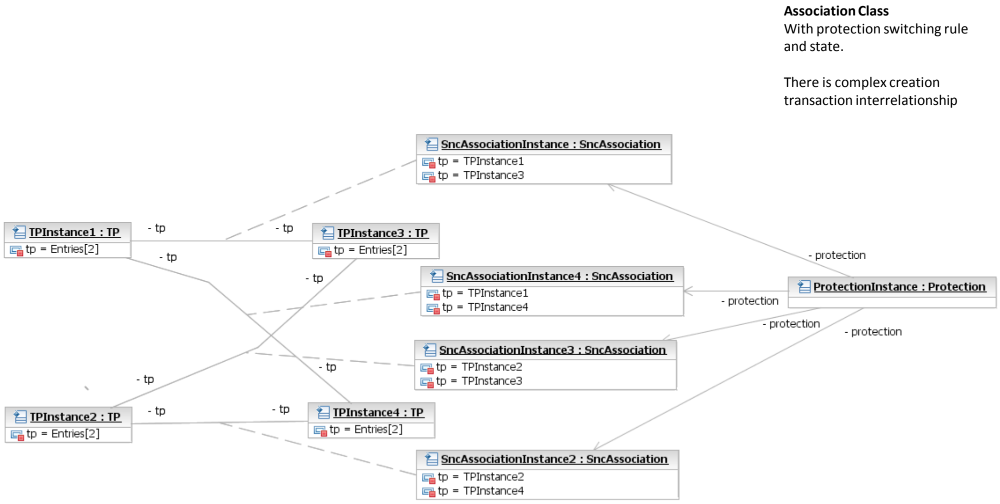


Figure H.4 – Complex relationship interrelationships

H.1.2 Usage in the non-transport domain

The choice of association class pattern or intervening class pattern is on a case-by-case basis.

The transport domain boundary is highlighted in the Figure H.5.

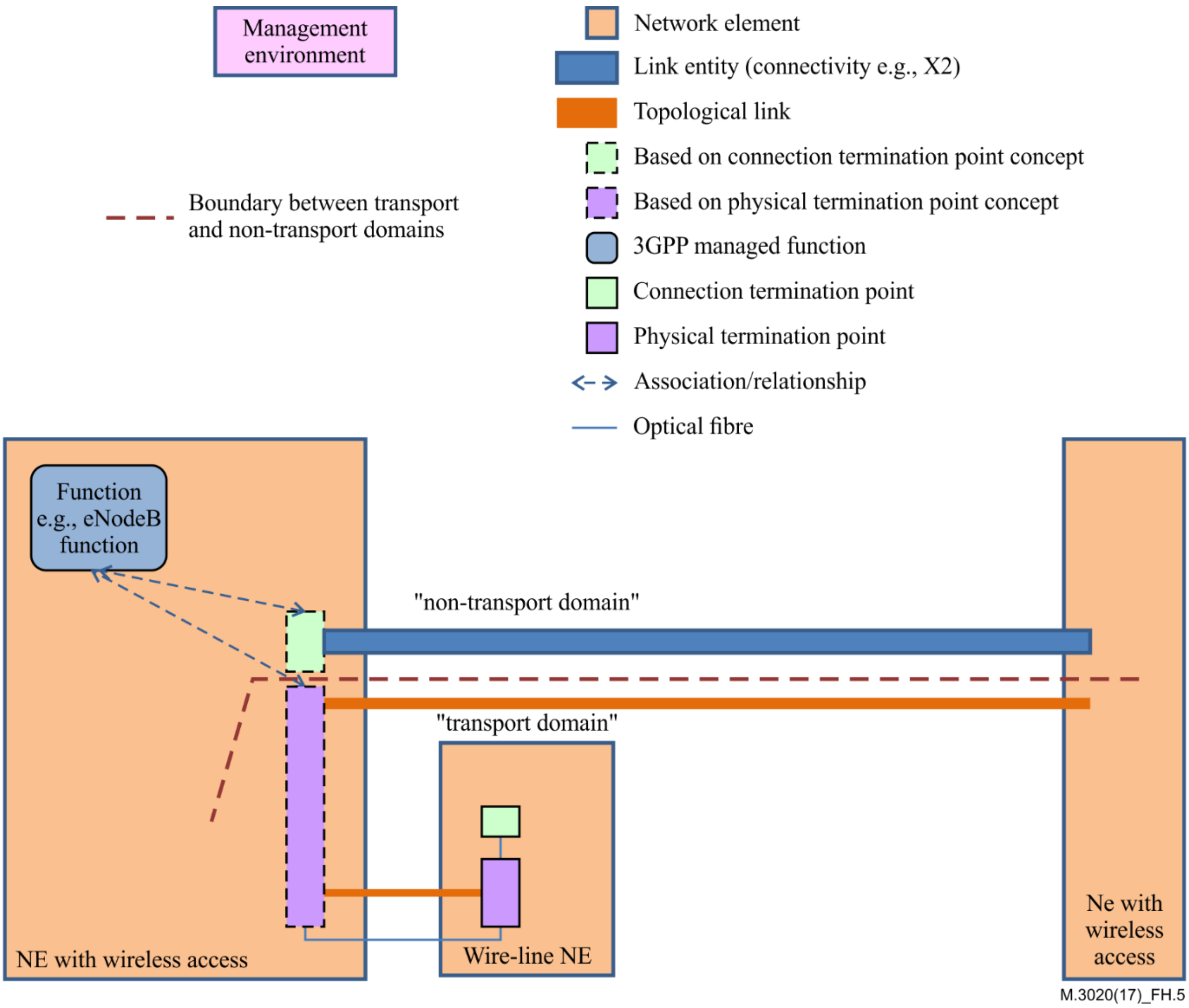


Figure H.5 – Highlighting the boundary between transport and non-transport domains

H.1.3 Usage in the transport domain

The following guidelines must be applied to the models of the "transport domain".

When considering interrelationships between classes the following guidelines should be applied:

- If considering all current and recognised potential future cases it is expected that the relationship between two specific classes will be 0..1:0..1 then a simple association should be used
  - This may benefit from an association class to convey rules and parameters about the association behaviour in complex cases.
- If there is recognised potential for cases currently or in future where there is a 0..*:0..* between two specific classes then intervening classes should be used to encapsulate the groupings etc. so as to convert it to 0..1:n..*.
  - Note that the 0..1:n..* association may benefit from an association class to convey rules and parameters about the association behaviour in complex cases but in the instance form this can probably be ignored or folded into the intervening class

- In general it seems appropriate to use an association class when the properties on the relationship instance cannot be obviously or reasonably folded into one of the classes at either end of the association and when there is no interdependency between association instances between a set of instances of the classes.

An example of usage of intervening class is the case of the TP-TP (TerminationPoint) relationship (0..*:0..*) where the SNC (SubNetworkConnection) is added as the intervening class between multiple TPs, i.e., TP-SNC. Note that TP-SNC actually becomes 0..2:n..* due to directionality encapsulation.

Considering the case of the adjacency relationship between PTPs it is known that although the current common cases are 1:1 there are some current and many potential future case of 0..*:0..* and hence a model that has an intervening class, i.e., the TopologicalLink, should be used.

For a degenerate instance cases of 0..*:0..* that happens to be 0..1:0..1 the intervening class pattern should still be used:

- Using the 0..1:0..1 direct association in this degenerate case brings unnecessary variety to the model and hence to the behaviour of the application (the 0..1:n..* model covers the 0..1:0..1 case with one single code form clearly)
- An instance of the 0..1:0..1 model may need to be migrated to 0..1:n..* as a result of some change in the network forcing an unnecessary administrative action to transition the model form where as in the 0..1:n..* form requires no essential change.

H.2 Use of "ExternalXyz" class

For further study.

Appendix I

Comparison with Recommendation ITU-T Z.601

(This appendix does not form an integral part of this Recommendation.)

This appendix provides information on the relationship between this Recommendation and [b-ITU-T Z.601] that is used for the development of ITU-T M.1400.x series of Recommendations (see Bibliography).

While this Recommendation provides a methodology for specifying management interfaces between two physical systems, [b-ITU-T Z.601] provides a framework for the development of one system. This data architecture identifies candidate interfaces within one system as well as the interfaces on the boundary of this system. These interfaces at the boundary will be between systems.

The methodology specified by this Recommendation is primarily aimed at the development of a set of management interface Recommendations rather than of individual systems. The data architecture prescribes no requirements capture similar to the requirements phase, as it prescribes the specification of individual systems only, not their purpose relative to an organization.

[b-ITU-T Z.601] focuses on specification of the external terminology and grammar as perceived by the end users. This Recommendation focuses on specification of management interfaces, which may not be perceived by the end users.

In this Recommendation, the requirements for the problem being solved fall into two classes. The first class of requirements is referred to as business requirements; the second class is referred to as specification requirements. The specification requirements may include requirements to support end-user interaction at their human-computer interfaces. Some of these requirements may specify syntactical requirements to be supported over any management interface. Syntactical requirements correspond to external terminology schemata of the data architecture as described in [b-ITU-T Z.601].

The output of the analysis phase will be an information model. This corresponds to a concept schema of the data architecture as described in [b-ITU-T Z.601]. If the information models from the analysis phase do not convey all the necessary information from the syntactical requirements, the implementation design may need to include a mapping from the syntactical requirements.

The documentation from the implementation design phase will consist of two parts:

1) A technology-dependent data specification common for several interfaces, e.g., using GDMO or CORBA IDL, corresponding to an internal terminology schema according to the data architecture in [b-ITU-T Z.601].
2) A technology-dependent specification of each interface, e.g., using CMIP or CORBA IDL, corresponding to a distribution schema according to the data architecture in [b-ITU-T Z.601].

Appendix II

Additional UML usage examples

(This appendix does not form an integral part of this Recommendation.)

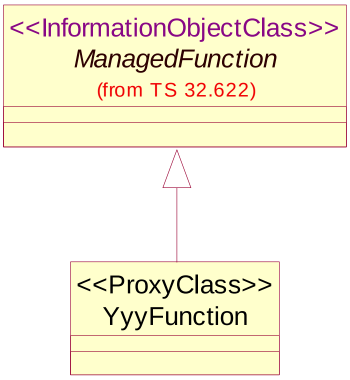
This appendix contains additional examples on the use of the UML described in Annex C.

II.1 Proxy class

II.1.1 First example

This shows a <<ProxyClass>> named YyyFunction. It represents all IOCs listed in the Note under the UML diagram. All the listed IOCs, in the context of this example, inherit from ManagedFunction IOC (see Figure II.1).

The use of <<ProxyClass>> eliminates the need to draw multiple UML <<InformationObjectClass>> boxes, i.e., those whose names are listed in the Note, in the UML diagram.



NOTE – The YyyFunction <<ProxyClass>> represents AsFunction, AucFunction, BgFunction, etc.

Figure II.1 – <<ProxyClass>> Notation example II.1

II.1.2 Second sample

Figure II.2 shows a <<ProxyClass>> named YyyFunction. It represents all IOCs listed in the Note right under the UML diagram. All the listed IOCs, in the context of this sample, have link (internal and external) relations.

The actual names of the IOC represented by InternalYyyFunction <<ProxyClass>> and by the ExternalYyyFunction <<ProxyClass>> are listed under the subclause of X.Y of the associated YyyFunction. For example, under X.Y.1 for AsFunction, two paragraphs are added to list all peer internal entities and external entities that are linked with AsFunction. See sample in quotation below that is using AsFunction as a sample for YyyFunction.

The actual names of the IOC represented by Link_a_z <<ProxyClass>> and by ExternalLink_a_z <<ProxyClass>> are listed under the subclause of X.Y of the associated YyyFunction. For example, under X.Y.1 for AsFunction, two paragraphs are added to list the names of the IOCs represented by Link_a_z and by ExternalLink_a_z. See the quoted text below that is using AsFunction as a sample for YyyFunction.

"

X.Y.1 AsFunction

X.Y.1.1 Definition

This IOC represents As functionality. For more information about the As, see [b-3GPP TS 23.002].

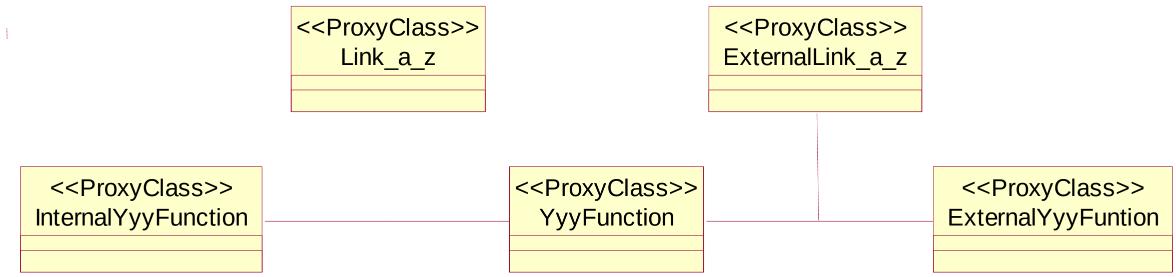
The linked InternalYyyFunction <<ProxyClass>> represents SlsFunction, CscfFunction, HlrFunction ...

The linked ExternalYyyFunction <<ProxyClass>> represents …

The Link_a_z <<ProxyClass>> represents Link_As_Scscf, Link_Bgcf_Scscf …

The ExternalLink_a_z <<ProxyClass>> represents …

"



NOTE – The 'Yyy' of YyyFunction <<ProxyClass>> represents AsFunction, AucFunction, etc.

Figure II.2 – <<ProxyClass>> Notation sample II.2

Appendix III

Guidelines on requirements numbering

(This appendix does not form an integral part of this Recommendation.)

The format for requirements numbering is the following:

REQ-Label-Category-Number

where "Label" is an abbreviation for the Recommendation (or part thereof). The set of labels is not finite and not subject for standardization. The set of categories is defined in this Recommendation.

Some issues:

– How to structure the label in a large requirements specification?

– How to handle deletion and addition of requirements?

The following guidelines are found to be useful:

– Requirements should never be renumbered. The only exception to this case is the first publication of a specification, but even in this case it may be better to avoid renumbering as the specification may have been used also in its draft form.

– Given that requirements are not to be renumbered, it cannot be expected that the requirements are numbered sequentially throughout the specification.

– The label can be used to divide the numbering into logical partitions. As an example, the style of "A_B" is recommended to identify "B" as a logical partition of "A". However, other styles can be used as long as the structure with "-" separating the fields of the requirements number is maintained.

– Use of postfix or prefix notations, i.e., adding something in front of "Number" or following "Number", are not recommended since the "Number" part is not intended to convey semantic information.

– As an alternative to the "A_B" style, the authors of a specification may choose to assign a number range to a group of requirements. This approach should be allowed.

Appendix IV

Stereotypes for naming purposes

(This appendix does not form an integral part of this Recommendation.)

Figure IV.1 illustrates the various stereotypes for naming purposes.

a) The <<names>> with solid-diamond (see clause C.3.3) identifies:
   - The naming class (close to the solid diamond) and a named class;
   - The naming scheme is DN;
   - The container (close to the solid diamond) and the content.

b) The <<names>> with other types of associations (and excluding those labelled "Not Allowed") identifies:
   - The naming class (close to the hollow diamond or the source with regard to arrow direction) and a named class (the target);
   - The naming scheme is DN.

c) The <<namedBy>> with dependency (dotted arrowed line) identifies:
   - The naming class (target with regard to arrow direction) and a named class (the source);
   - The naming scheme is DN.

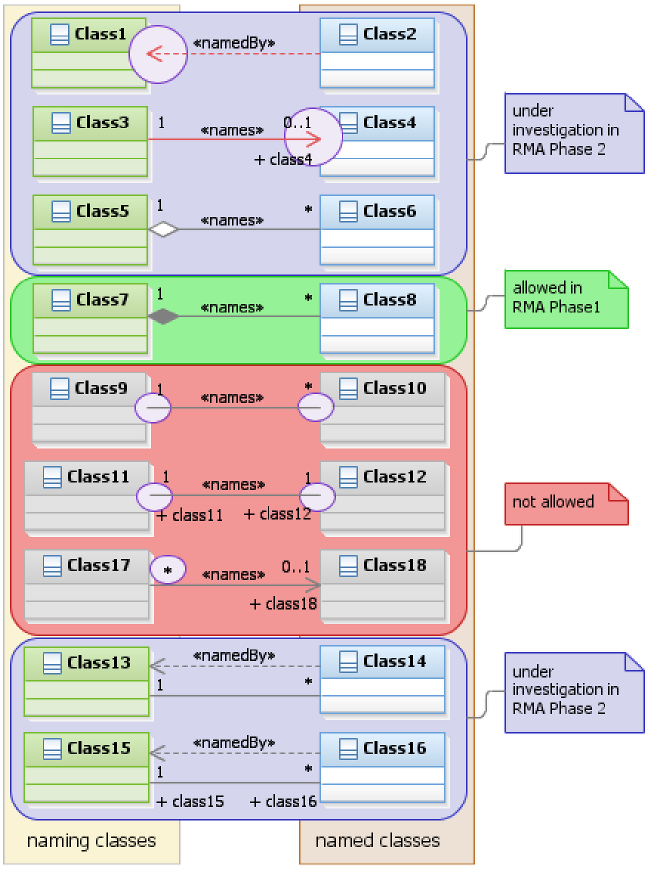


Figure IV.1 – Various forms of naming stereotypes

Bibliography

[b-ITU-T M.1401] Recommendation ITU-T M.1401 (2006), *Formalization of interconnection designations among operators' telecommunication networks*.

[b-ITU-T M.1402] Recommendation ITU-T M.1402 (2012), *Formalization of data for service management*.

[b-ITU-T M.1403] Recommendation ITU-T M.1403 (2007), *Formalization of generic orders*.

[b-ITU-T M.1404] Recommendation ITU-T M.1404 (2007), *Formalization of orders for interconnections among operators' networks*.

[b-ITU-T M.1405] Recommendation ITU-T M.1405 (2007), *Formalization of orders for service management among operators*.

[b-ITU-T Z.601] Recommendation ITU-T Z.601 (2007), *Data architecture of one software system*.

[b-3GPP TS 23.002] 3GPP TS 23.002 V14.1.0 (2017), *Network architecture*.

[b-3GPP TS 32.101] 3GPP TS 32.101 V14.0.0 (2017), *Telecommunication management; Principles and high level requirements*.

[b-3GPP TS 32.150] 3GPP TS 32.150 V14.1.1 (2017-04), *Telecommunication management; Integration ReferencePoint (IRP) Concept and definitions*.

[b-3GPP TS 32.156] 3GPP TS 32.156 V14.0.1 (2017-04), *Telecommunication management; Fixed Mobile Convergence (FMC) model repertoire*

[b-3GPP TS 32.157] 3GPP TS 32.157 V14.0.1 (2017-04), *Telecommunication management; Integration Reference Point (IRP) Information Service (IS) template*.

[b-3GPP TS 32.302] 3GPP TS 32.302 V13.0.0 (2016), *Telecommunication management; Configuration Management (CM); Notification Integration Reference Point (IRP); Information Service (IS)*.

[b-3GPP TS 32.300] 3GPP TS 32.300 V13.1.0 (2016), *Telecommunication management; Configuration Management (CM); Name convention for Managed Objects.*

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	**Telecommunication management, including TMN and network maintenance**
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2017