International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

M.3020

(07/2008)

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.

Source

Recommendation ITU-T M.3020 was approved on 29 July 2008 by ITU-T Study Group 4 (2005-2008) under Recommendation ITU-T A.8 procedure.

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/.

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 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.x] Recommendation ITU-T M.3050.x-series (2007), *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 X.680] Recommendation ITU-T X.680 (2002) | ISO/IEC 8824-1:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

[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.*

[OMG UML] OMG: *Unified Modelling Language Specification, Version 1.5*. <http://www.omg.org/spec/UML/1.5/>

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 Terms defined in [ITU-T M.3010]:

– user;
– management service;
– management function set.

3.1.2 Terms defined in [OMG UML]:

– activity diagram;
– actor;
– association;
– class;
– class diagram;
– classifier;
– collaboration diagram;
– composition;
– modelElement;
– sequence diagram;
– state diagram;
– stereotype;
– use case.

3.1.3 Term defined in [ITU-T M.3060]:

– reference point.

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

3.2.4 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].

3.2.5 integration reference point: An architectural concept that is described by a set of specifications for the definition of a certain aspect of the Itf-N, 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 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].

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

3.2.12 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.

3.2.13 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].

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

4 Abbreviations

This Recommendation uses the following abbreviations:

3GPP	3rd Generation Partnership Project
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
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

MISM Management Interface Specification Methodology

NA Not Applicable

NE Network Element

NON Non-functional (usage: requirements category)

OMG Object Management Group

OO Object Oriented

OSI Open Systems Interconnection

SDL Specification and Description Language

SOA Service Oriented Architecture

SS Solution Set

TS Technical Specification

UML Unified Modelling Language

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 SG 4 use are specified in [b-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 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 subclauses 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 a great impact on concepts and data designed in the subsequent phases. For more detail, see Appendix III, and see the [b-ITU-T M.1400]-series Recommendations on data design for human-computer interfaces.

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 M.3200 and M.3050 approaches is described in [ITU-T M.3050.x].

Management functions must be grouped and supported within applications that address specific business needs, so the linkage between the eTOM processes, the M.3200 management services, the 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.

An example requirements definition is available in Appendix I.

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 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, SDL [b-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 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 B.

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., 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 subclauses.

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, behaviour definition of 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 CORBA based management, the information model is defined using IDL.

7.3.4.3 GDMO

In the context of the paradigm based on 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 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 an 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 sections.

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) subsection 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. An example of the use of this template can be found in Appendix I.

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.

Table A.1 – Requirements concepts

Management concept	UML notation	Comment
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 following conventions and templates 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 telecommunications 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.	
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".	

Table A.2 – Use case template

Use case stage	Evolution/Specification	<<Uses>> Related use
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.

A.2 Requirements template

A.2.1 Concepts and background

"A.2.1" represents a clause number in the actual Recommendation|Technical Specification.

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 part should be separated. Supporting background information, where required, should be placed in an appendix.

A.2.1.a SubSectionTitle

SubSectionTitle 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.

A.2.2 Business level requirements

A.2.2.1 Requirements

A.2.2.1.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 clause A.2.2.1.

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 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.

A.2.2.2 Actor roles

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

A.2.2.3 Telecommunications resources

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

A.2.2.4 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.

A.2.2.4.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 clause 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 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)>

A.2.3 Specification level requirements

A.2.3.1 Requirements

A.2.3.1.a SubSetTitle

The high-level use cases are further refined 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.

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

A.2.3.2 Actor roles

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

A.2.3.3 Telecommunications resources

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

A.2.3.4 Use cases

A.2.3.4.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.

Annex B

Analysis

(This annex forms an integral part of this Recommendation)

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

The analysis template is based on the 3GPP information service [b-3GPP TS 32.151] and augmented to meet additional requirements on the methodology (e.g., traceability).

For a management interface specification, both clauses B.2.2 and B.2.3 shall be used. For an information model (e.g., a network resource model), only clause B.2.2 shall be used.

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

B.1	*Conventions*
B.1.1	*Mandatory, optional and conditional qualifiers*
B.2	*Analysis template*
B.2.1	*Concepts and background*
B.2.2	*Information object classes*
B.2.2.1	*Imported information entities and local labels*
B.2.2.2	*Class diagram*
B.2.2.2.1	*Attributes and relationships*
B.2.2.2.2	*Inheritance*
B.2.2.3	*Information object class definitions*
B.2.2.3.a	*InformationObjectClassName*
B.2.2.4	*Information relationship definitions*
B.2.2.4.a	*InformationRelationshipName (supportQualifier)*
B.2.2.5	*Information attribute definitions*
B.2.2.5.1	*Definition and legal values*
B.2.2.5.2	*Constraints*
B.2.2.6	*Common notifications*
B.2.2.7	*System state model*
B.2.3	*Interface definition*
B.2.3.1	*Class diagram representing interfaces*
B.2.3.2	*Generic rules*
B.2.3.b	*Interface InterfaceName (supportQualifier)*
B.2.3.b.a	*Operation OperationName (supportQualifier)*
B.2.3.b.b	*Notification NotificationName (supportQualifier)*
B.2.3.c	*Scenario*

B.1 Conventions

B.1.1 Mandatory, optional and conditional qualifiers

This subclause 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, together with a reason, shall be returned to the manager.	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.
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

B.2.1 Concepts and background

"B.2.1" represents a clause number in the actual Recommendation|IS.

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

B.2.1.a SubSectionTitle

SubSectionTitle 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.

B.2.2 Information object classes

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

B.2.2.1 Imported information entities and local labels

This clause identifies a list of information entities (e.g., information object class, information relationship, information attribute) that have been defined in other specifications and that are imported in the present document. This includes information entities from other specifications imported for inheritance purpose. Each element of this list is a pair (label reference, local label). The label reference contains the name of the specification where it is defined, the type of the information entity and its name. 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.

Label reference	Local label

Imported elements should be from protocol neutral definitions based on this methodology but may import elements from other specifications, if necessary, in the interest of migration of protocol specific specifications over time.

B.2.2.2 Class diagram

B.2.2.2.1 Attributes and relationships

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

Characteristics (relationships) of imported information object classes need not be repeated in the diagram. Information object classes should be defined using the stereotype <<InformationObjectClass>>.

B.2.2.2.2 Inheritance

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

Characteristics (attributes, relationships) of imported information object classes need not be repeated in the diagram. Information object classes should be defined using the stereotype <<InformationObjectClass>>.

NOTE – Some inheritance relationships presented in clause B.2.2.2.2 can be repeated in clause B.2.2.2.1 to enhance readability.

B.2.2.3 Information object class definitions

Class name	Qualifier	Requirement IDs

Each information object class is defined using the following structure.

Inherited items (attributes etc.) shall not be shown, as they are defined in the parent IOC(s) and thus valid for all subclasses.

B.2.2.3.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 an information object class.

B.2.2.3.a.1 Definition

The <Definition> subclause is written in natural language. The <Definition> subclause refers to the information object class itself. The characteristics related to the relationships that the object class can have with other object classes cannot be found in the definition. The reader has to refer to relationships definition to find such kind of information. Information related to inheritance shall be precised here.

B.2.2.3.a.2 Attributes

The <Attributes> subclause presents the list of attributes, which are the manageable properties of the object class. Each element is a tuple (attributeName, supportQualifier, readQualifier, writeQualifier):

– *The supportQualifier indicates whether the attribute is Mandatory (M), Optional (O), Conditional-Mandatory (CM), Conditional-Optional (CO), SS-Conditional (C) or Not supported (–). Mandatory, Optional, Conditional or not supported ("M","O","C", or "–", respectively).*

– *The readQualifier indicates whether the attribute shall be readable by the Manager is Mandatory (M), Optional (O), Conditional-Mandatory (CM), Conditional-Optional (CO), SS-Conditional (C) or not supported (–). Allowed values are: Mandatory (M), Optional (O) and Not supported (–).*

– *The writeQualifier indicates whether the attribute shall be writeable by the Manager. The semantics for writeQualifier is identical to supportQualifier, for "M", "O", and "–". Allowed values are: Mandatory (M), Optional (O) and Not supported (–).*

There is a dependency relationship between the supportQualifier,readQualifier, and writeQualifier. The supportQualifier indicates the requirements for the support of the attribute. For any given attribute, regardless of the value of the supportQualifier, at least one of the readQualifier or writeQualifier must be "M". The implication of the "O" supportQualifier is that the attribute is optional; however, the read and write qualifiers indicate how the optional attribute shall be supported, should the optional attribute be supported.

Private or Agent Internal attributes are per definition not readable by the IRPManager. Their readQualifier is hence always "–".

Private or Agent Internal attributes are per definition not writable by the IRPManager. Their writeQualifier is hence always "–".

The readQualifier and writeQualifier of a supported attribute, that is public, may not be both "–".

The use of "–" in supportQualifier is reserved for documenting support of attributes defined by an "Archetype" IOC. Attributes with a supportQualifier of "–" are not implemented by the IOC that is realizing a subset of the attributes defined by the "Archetype". The readQualifier and writeQualifier are of no relevance in this case. However, a not supported attribute is neither readable nor writable. For this reason, the readQualifier and writeQualifier shall be "–" for unsupported attributes.

For any IOC that uses one or more attributes from an "Archetype", a separate table shall be used to indicate the supported attributes. This table is absent if no "Archetype" attributes are supported. For example, if a particular IOC has defined attributes (i.e., attributes not defined by an "Archetype") and encapsulates attributes from two "Archetype"s, then the totality of the attributes of the said IOC will be contained in three separate tables.

This information is provided in a table.

Attribute name	Support Qualifier	Read Qualifier	Write Qualifier	Requirement IDs

B.2.2.3.a.3 Attribute constraints

The <Attribute constraints> subclause presents constraints between attributes that are always held to be true. Those properties are always held to be true during the lifetime of the attributes and in particular do not need to be repeated in pre- or post-conditions of operations or notifications.

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

B.2.2.3.a.4 Relationships

The <Relationship> subclause presents the list of relationships in which this class is involved. Each element is a relationshipName.

The relationships will be listed in a table as follows:

Relationship	Requirement IDs

And each relationship name should be a reference (and preferably also a hyperlink) to the appropriate clause of B.2.2.

NOTE – This subclause is optional and may be avoided since all relationships are represented in the class diagram in clause B.2.2.2.1.

B.2.2.3.a.5 State diagram

The <State diagram> 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.

NOTE – This subclause does not need to be present when there is no state diagram to define.

B.2.2.3.a.6 Notifications

The <Notifications> subclause, for this IOC, presents:

a) *optionally, a reference to the common notifications defined in subclause B.2.2.6 as valid for this IOC; and*

b) *optionally, a list of notifications that shall be excluded from the list of common notifications (defined in subclause B.2.2.6) for this IOC;*

NOTE 1 – Inherited notifications from the parent IOC(s) cannot be excluded.

and

c) *optionally, a list of notifications applicable to this IOC, and which may or may not be defined in the common notifications in subclause B.2.2.6.*

The notifications identified in this subclause are notifications that can be emitted across the Itf-N, where the "object class" and "object instance" parameters of the notification header (see Note 3) of these notifications identifies an instance of the IOC defined by the encapsulating subclause (i.e., clause B.2.2.3.a).

The notifications identified in this subclause may originate from implementation object(s) whose identifier is mapped in the implementation, to the object instance identifier used over the Itf-N. Hence, the presence of notifications in this clause (i.e., clause B.2.2.3.a.6) does not imply nor identify those notifications as being originated from an instance of the IOC defined by the encapsulating subclause (i.e., clause B.2.2.3.a).

The information related to option c) above is provided in a table. An example of such a table is given below:

Name	Qualifier	Requirement IDs	Notes

NOTE 2 – This subclause and table do not need to be present when there are no additional notifications to those in clause B.2.2.6.

NOTE 3 – The notification header is defined in the notification IRP Information service [b-3GPP TS 32.302].

B.2.2.4 Information relationship definitions

This clause first lists all the relationships supported by this Recommendation|Specification in the following table. Support qualifier is defined as for attributes in clause B.1.

Relationship	Support Qualifier	Requirement IDs

Each information relationship is defined using the following structure.

Inherited relationships shall not be shown, as they are defined by the parent IOC(s) and thus valid for all subclasses.

B.2.2.4.a InformationRelationshipName (supportQualifier)

InformationRelationshipName is the name of the information relationship followed by a qualifier (see clause B.1).

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

B.2.2.4.a.1 Definition

The <Definition> subclause is written in natural language.

B.2.2.4.a.2 Roles

The <Roles> subclause identifies the roles played in the relationship by object classes. Each element is a pair (roleName, roleDefinition).

This information is provided in a table.

Name	Definition

B.2.2.4.a.3 Constraints

The <Constraints> subclause contains the list of properties specifying the semantic invariants that must be preserved on the relationship. Each element is a pair (propertyName, propertyDefinition). Those properties are always held to be true during the lifetime of the relationship and do not need to be repeated in pre- or post-conditions of operations or notifications.

This information is provided in a table.

Name	Definition

B.2.2.5 Information attribute definitions

Each information attribute is defined using the following structure.

Inherited attributes shall not be shown, as they are defined in the parent IOC(s) and thus valid for all subclasses.

B.2.2.5.1 Definition and legal values

This subclause contains for each attribute being defined its name, its definition written in natural language and a list of legal values supported by the attribute. The definition may also contain a formal definition of the type of the attribute.

In the case where the legal values can be enumerated, each element is a pair (legalValueName, legalValueDefinition), unless a legalValueDefinition 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 information is provided in a table.

Attribute Name	Definition	Legal Values

B.2.2.5.2 Constraints

The <Constraints> subclause 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.

Name	Affected attribute(s)	Definition

B.2.2.6 Common notifications

This <Common Notifications> subclause presents a list of notifications that can be referred to by any IOC defined by this management interface specification. These notifications are only applicable to IOCs referring to this subclause in subclause B.2.2.3.a.6.

This information is provided in a table.

Name	Qualifier	Notes

NOTE – This subclause does not need to be present when there are no common notifications.

B.2.2.7 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 clause 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.

B.2.3 Interface definition

"B.2.3" represents a number, immediately following "B.2.2". This clause shall be used for all management interface specifications and optional for information model only specifications.

B.2.3.1 Class diagram representing interfaces

Each interface is defined in the diagram. This shall be a UML compliant class diagram (see also Annex C).

Interfaces are defined using a stereotype <<Interface>>. Each interface contains a set of either operations or notifications which are mandatory or either a single operation or a single notification which is optional. Stereotypes (see Annex C) are used to specify optional or mandatory interfaces. On the class diagram, each operation and notification in an interface shall be qualified as "public" by the addition of a symbol "+" before each operation and notification.

B.2.3.2 Generic rules

The following rules are relevant for 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 studies.

B.2.3.b Interface InterfaceName (supportQualifier)

InterfaceName is the name of the interface followed by a qualifier (see 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 as defined here below.

Operation name	Qualifier	Requirement IDs

OperationName is the name of the operation followed by a qualifier (see clause B.1). Conditions must be defined in the text below this table.

Each operation is defined using the following structure.

NOTE – Grouping of operations/partitioning of interface contents and naming of interfaces is for further study.

B.2.3.b.a Operation OperationName (supportQualifier)

OperationName is the name of the operation followed by a qualifier (see clause B.1).

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

B.2.3.b.a.1 Definition

The <Definition> subclause is written in natural language.

B.2.3.b.a.2 Input parameters

List of input parameters of the operation. Each element is a tuple (inputParameterName, supportQualifier, InformationType, inputParameterComment). Legal values for the supportQualifier are specified in clause B.1.

This information is provided in a table.

Parameter Name	Qualifier	Information type	Comment

B.2.3.b.a.3 Output parameters

List of output parameters of the operation. Each element is a tuple (outputParameterName, supportQualifier, MatchingInformation, outputParameterComment). Legal values for the supportQualifier are specified in clause B.1.

This information is provided in a table.

Parameter Name	Qualifier	Matching Information	Comment

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

B.2.3.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.

Each assertion is defined by a pair (propertyName, propertyDefinition). All assertions constituting the pre-condition are provided in a table.

Assertion Name	Definition

B.2.3.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.

Each assertion is defined by a pair (propertyName, propertyDefinition). All assertions constituting the post-condition are provided in a table.

Assertion Name	Definition

B.2.3.b.a.6 Exceptions

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

B.2.3.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.

Exception Name	Definition	
	Condition	
	Return info	
	Exit state	
	Condition	
	Return info	
	Exit state	

B.2.3.b.a.7 Constraints

The <Constraints> subclause presents constraints for the operation or its parameters.

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

B.2.3.b.b Notification NotificationName (supportQualifier)

NotificationName is the name of the notification followed by a qualifier (see clause B.1).

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

B.2.3.b.b.1 Definition

The <Definition> subclause is written in natural language.

B.2.3.b.b.2 Input parameters

List of input parameters of the notification. Each element is a tuple (inputParameterName, supportQualifier and filteringQualifier, matchingInformation, inputParameterComment).

The column "Qualifiers" contains the two qualifiers, supportQualifier (see clause B.1) and filteringQualifier, separated by a comma. The filteringQualifier indicates whether the parameter of the notification can be filtered or not. Values are Yes (Y) or No (N). The matchingInformation refers to information in the state "toState".

This information is provided in a table.

Parameter Name	Qualifiers	Matching Information	Comment

B.2.3.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.

B.2.3.b.b.3.1 From state

This subclause is a collection of assertions joined by AND, OR, and NOT logical operators.

Each assertion is defined by a pair (propertyName, propertyDefinition). All assertions constituting the state "from state" are provided in a table.

Assertion Name	Definition

B.2.3.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 (propertyName, propertyDefinition). All assertions constituting the state "to state" are provided in a table.

Assertion Name	Definition

B.2.3.b.b.4 Constraints

The <Constraints> subclause presents constraints for the notification or its parameters.

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

B.2.3.c Scenario

"B.2.3.c" represents a number, immediately following "B.2.3".

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, inheritance and import

B.3.1 Property

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 clause 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

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

To facilitate reuse of IOC definitions among IRP specifications, an import mechanism is used by one IRP specification (called the subject IRP specification) to reuse IOC definition defined in another IRP specification. When the subject IRP specification imports an IOC, it cannot change the imported IOC property. If it requires changes to the imported IOC, it must use inheritance to define its own new class.

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 32.152].

C.1 Introduction

UML provides a rich set of concepts, notations and model elements to model distributive 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.

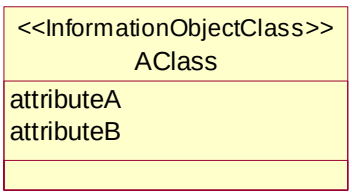
C.2 Basic model elements

C.2.1 General

UML defined a number of basic model elements. This subclause lists the selected subset for use in the repertoire. The semantics of the selected ones are defined in [OMG UML].

C.2.2 Attribute (subclause 3.25 of [OMG UML])

This sample shows two attributes, listed as strings in the attribute compartment of the class AClass.

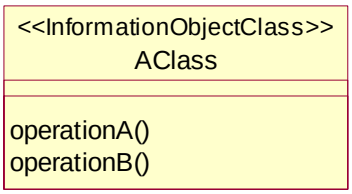


C.2.3 Aggregation (subclause 3.43.2.5 of [OMG UML])

This sample shows a hollow diamond attached to the end of a path to indicate aggregation. The diamond is attached to the class that is the aggregate.

C.2.4 Operation (subclause 3.26 of [OMG UML])

This sample shows two operations, shown as strings in the operation compartment of class AClass, that the instance of AClass may be requested to perform. The operation has a name, e.g., operationA and a list of arguments (not shown).

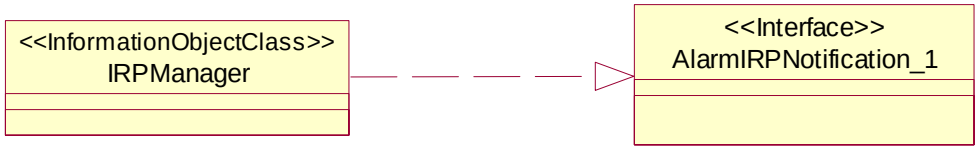


C.2.5 Association (subclause 3.41 of [OMG UML])

This sample shows a binary association between exactly two model elements. An association can relate a model element to itself. This sample shows a bidirectional association in that one model element is aware of the other. Association can be unidirectional (shown with an open arrow at one association end) in that only the source model element is aware of the target model element and not vice versa.

C.2.6 Realization relationship (subclause 2.5.2.1 of [OMG UML])

This sample shows the realization relationship between an AlarmIRPNotification_1 (the supplier) and a model element, IRPManager, that implements it.



C.2.7 Generalization relationship (subclause 3.50 of [OMG UML])

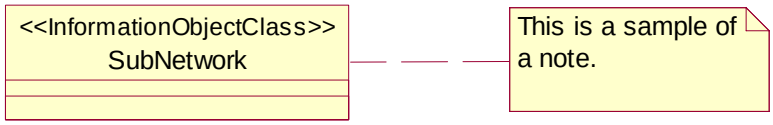
This sample shows a generalization relationship between a more general element (the agent) and a more specific element (the Agent_vendor_A) that is fully consistent with the first element and that adds additional information.

C.2.8 Dependency relationship (subclause 3.51 of [OMG UML])

This sample shows that BClass instances have a semantic relationship with AClass instances. It indicates a situation in which a change to the target element will require a change to the source element in the dependency.

C.2.9 Note (subclause 3.11 of [OMG UML])

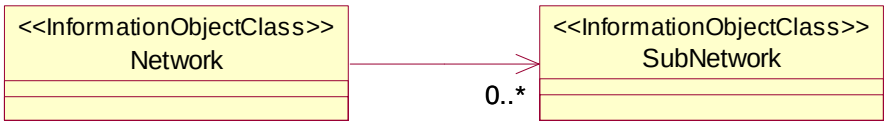
This sample shows a note, as a rectangle with a "bent corner" in the upper right corner. The note contains arbitrary text. It appears on a particular diagram and may be attached to zero or more modelling elements by dashed lines.



C.2.10 Multiplicity, a.k.a. cardinality (subclause 3.44 of [OMG UML])

This sample shows a multiplicity attached to the end of an association path. The meaning of this multiplicity is that one network instance is associated with zero, one or more subnetwork instances.

In previous versions of [b-3GPP TS 32.152], the cardinality zero can indicate that the IOC has the so-called "transient state" characteristics. For example, it indicates that the instance is not yet created but it is in the process of being created. From this version of the methodology, the cardinality zero will not be used to indicate these characteristics since such characteristics are considered inherent in all IOCs (all IOCs defined are considered to have such inherent "transient state" characteristics).



C.3 Entity stereotypes

C.3.1 General

This subclause defines all allowable entity stereotypes that are summarized in Table C.3-1. Except <<Interface>>, <<Type>> (which are defined in [OMG UML]), all other stereotypes are extensions specifically designed for use in Recommendations based on the methodology.

Table C.3-1 – Entity stereotypes

Stereotype	Base class	Affected metamodel elements
Interface	Class	
Type	Class	
ProxyClass	Class	
Notification	Class	
Archtetype	Classifier (subclause 2.5.2.10 of [OMG UML])	
InformationObjectClass	Classifier	
opt (alternatively "optional")	ModelElement	Attribute, Parameter and Operation

C.3.2 <<Interface>>

Subclause 2.5.2.25 of [OMG UML]:

"An interface is a named set of operations that characterize the behaviour of an element. In the metamodel, an Interface contains a set of Operations that together define a service offered by a Classifier realizing the Interface. A Classifier may offer several services, which means that it may realize several Interfaces, and several Classifiers may realize the same Interface.

...

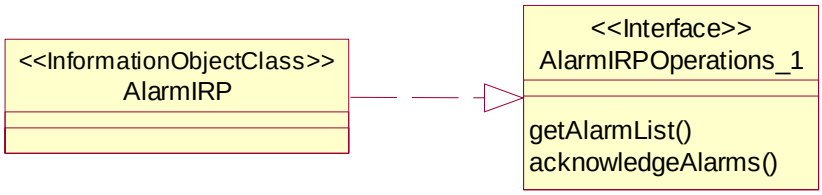
Interfaces [may or] may not have Attributes, Associations, or Methods. An Interface may participate in an Association provided the Interface cannot see the Association; that is, a Classifier (other than an Interface) may have an Association to an Interface that is navigable from the Classifier but not from the Interface."

From subclause 2.5.4.6 of [OMG UML]: "The purpose of an interface is to collect a set of operations that constitute a coherent service offered by classifiers. Interfaces provided a way to partition and characterize groups of operations. An interface is only a collection of operations with a name. It cannot be directly instantiateC.".

From subclause 2.5.4.6 of [OMG UML]: "Several classifiers may realize the same interface. All of them must contain at least the operations matching those contained in the interface. The specification of an operation contains the signature of the operation (i.e., its name, the types of the parameters and the return type). An interface does not imply any internal structure of the realizing classifier. For example, it does not include which algorithm to use for realizing an operation. An operation may, however, include a specification of the effects [e.g., with pre and post-conditions] of its invocation."

C.3.2.1 Sample

This sample shows an AlarmIRPOperations_1 <<Interface>> that has two operations. The input and output parameters of the operations are hidden (i.e., not shown). The AlarmIRP has a unidirectional mandatory realization relationship with the <<Interface>>.



<<Interface>> Notation

C.3.3 <<Type>>

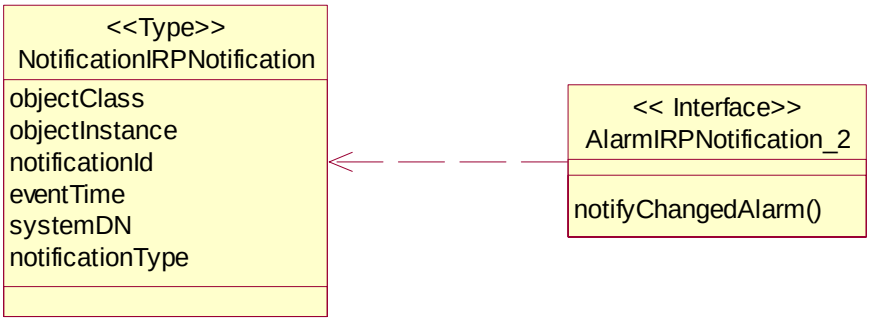
C.3.3.1 General

Subclause 3.28 of [OMG UML]: "[A Type is] a domain of objects together with the operations applicable to the objects, without defining the physical implementation of those objects. A Type may not contain any methods, maintain its own thread of control, or be nested. However, it may have Attributes and Associations. The Associations of a Type are defined solely for the purpose of specifying the behaviour of the Type's operations and do not represent the implementation of state data".

C.3.3.2 Sample

This sample shows the NotificationIRPNotification <<Type>> that specifies the five parameters (the notification header of Notification IRP). The AlarmIRPNotification_2 <<Interface>> depends

(see the dependency relationship, a dashed open arrow line) on this <<Type>> for the construction of the notification emitted via the operation notifyChangedAlarm().



<<Type>> Notation

C.3.4 <<ProxyClass>>

C.3.4.1 General

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 behaviours 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>>.

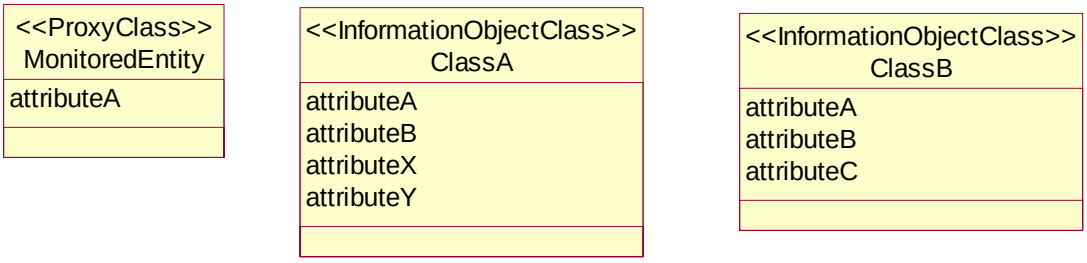
A particular <<InformationObjectClass>> can be represented by zero, one or more <<ProxyClass>> or <<Archetype>>. For example, the ManagedElement <<InformationObjectClass>> can have MonitoredEntity <<ProxyClass>> and ManagedEntity <<ProxyClass>>.

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

C.3.4.2 Sample

This shows a <<ProxyClass>> named MonitoredEntity. It represents all NRM <<InformationObjectClass>> (e.g., GgsnFunction <<InformationObjectClass>>) whose instances are being monitored for alarm conditions.

Note that <<MonitoredEntity>> does not define attributeA. The attributeA is already defined by all <<InformationObjectClass>> represented by the <<MonitoredEntity>>, i.e., ClassA and ClassB.



<<ProxyClass>> (sample 1)

See Appendix V for more samples that use <<ProxyClass>>.

C.3.5 <<Archetype>>

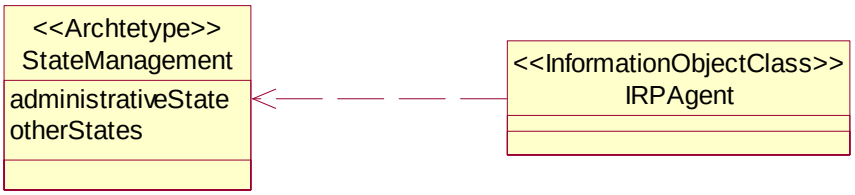
C.3.5.1 General

This represents a number of <<InformationObjectClass>>. It encapsulates attributes, links, operations, and interactions that are typical of the represented <<InformationObjectClass>>.

The semantics of an <<Archetype>> is that all attributes, links operations and interactions encapsulated by the <<Archetype>> may or may not be present in the represented <<InformationObjectClass>>. The <<Archetype>> represents a placeholder class that is most useful in technology neutral analysis models that will require further specification and/or mapping within a more complete construction model.

C.3.5.2 Sample

This shows a <<Archetype>> named StateManagement. It also shows a <<InformationObjectClass>> agent that depends on this StateManagement. Note that the StateManagement has defined a number of attributes, the classes that depend on this StateManagement may or may not use all of the StateManagement attributes. In other words, at least one of the attributes of StateManagement is present in the agent. The precise set of StateManagement attributes used by the agent is specified in the agent specification.



<<Archetype>>> Notation

C.3.6 <<InformationObjectClass>>

C.3.6.1 General

This represents an IOC. Each <<InformationObjectClass>> represents a set of instances with similar structure, behaviour and relationships.

This <<InformationObjectClass>> and other information classes such as <<Interface>> are mapped into technology-specific model elements such as GDMO managed object class for CMIP technology. The mapping of the protocol-neutral modelling constructs to technology-specific modelling constructs are captured in the corresponding protocol-specific specifications.

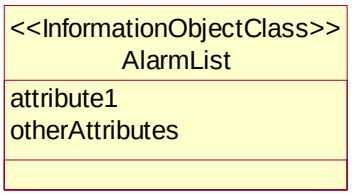
The name of a <<InformationObjectClass>> has scope within the Recommendation in which it is specified and the name must be unique among all <<InformationObjectClass>> names within that Recommendation. The Recommendation name is considered in a similar way as the UML Package-name.

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

Subclause 3.22.1 of [OMG UML]: "A *class* represents a concept within the system being modelled. Classes have data structure and behaviour and relationships to other elements."

C.3.6.2 Sample

This sample shows an AlarmList <<InformationObjectClass>>.

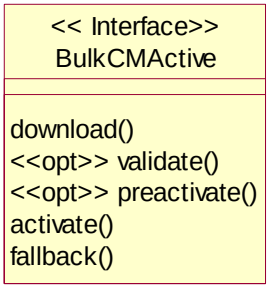


<<InformationObjectClass>>> Notation

C.3.7 <<opt>>

The <<opt>> (alternatively <<optional>>) enables the indication of optionality of attributes, parameters and operations (respectively) within the UML diagrams. The semantics of optionality is clearly defined in Annex A.

In the absence of the stereotype, the attribute, parameter, or operation in question is mandatory.



Example of the use of optionality indicator for operations

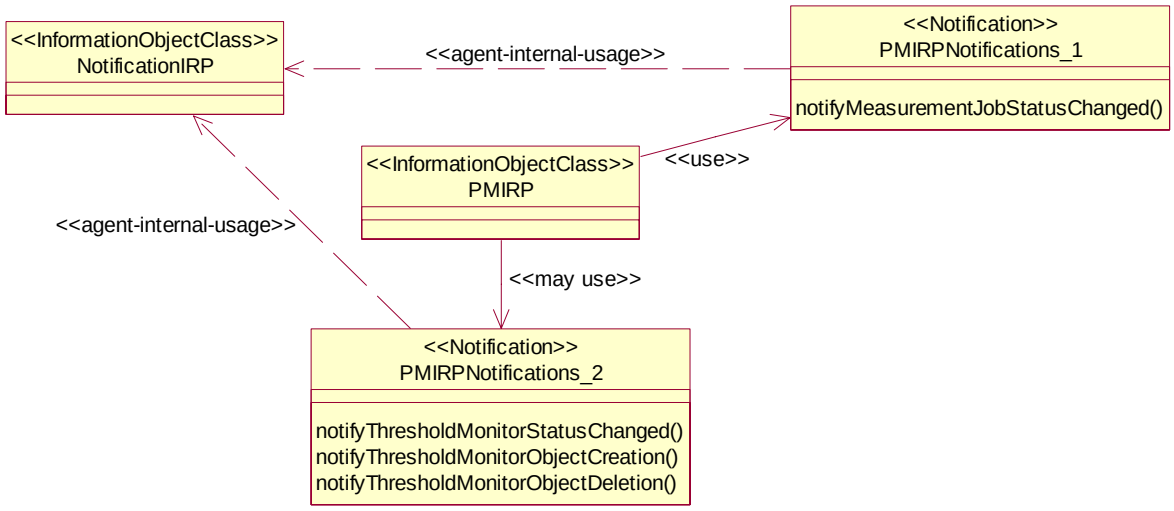
C.3.8 <<Notification>>

C.3.8.1 General

<<Notification>> is a named set of notifications. In the metamodel, a <<Notification>> contains a set of Notifications that together define a service offered by a Classifier realizing the <<Notification>>.

C.3.8.2 Sample

This sample shows a <<Notification>> named "PMIRPNotifications_1" that has one notification and a <<Notification>> named "PMIRPNotifications_2" that has three notifications.



C.4 Association stereotypes

C.4.1 General

This subclause defines all allowable association stereotypes that are summarized in Table C.4-1. Except <<use>> (which is defined in [OMG UML]), all other stereotypes are extensions specifically designed for use in Recommendations based on the methodology.

Table C.4-1 – Association stereotypes

Stereotype	Base class	Affected metamodel elements
Use	Association	
may use	Association	
may realize	Association	
Emits	Association	

C.4.2 <<use>> and <<may use>>

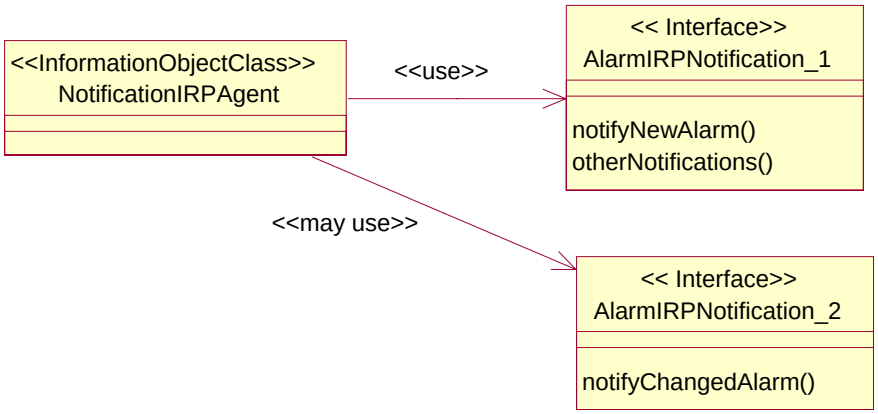
C.4.2.1 General

The <<use>> and <<may use>> are unidirectional associations. The target must be an <<Interface>>. The <<use>> states that the source class must have the capability to use the target <<Interface>> in that it can invoke the operations defined by the <<Interface>>. Support of the capability by the source entity is mandatory. The <<may use>> states that the source class may have the capability to use the target <<Interface>> in that it may invoke the operations defined by the <<Interface>>. Support of the capability by the source entity is optional.

The operations defined by the <<Interface>> are visible across the interface/reference point.

C.4.2.2 Sample

This shows that the NotificationAgent shall use the notifyNewAlarm and otherNotifications of AlarmIRPNotification_1 and may use the notifyChangedAlarm of AlarmIRPNotification_2.



<<use>> and <<may use>> Notation

C.4.3 Relationship realize and <<may realize>>

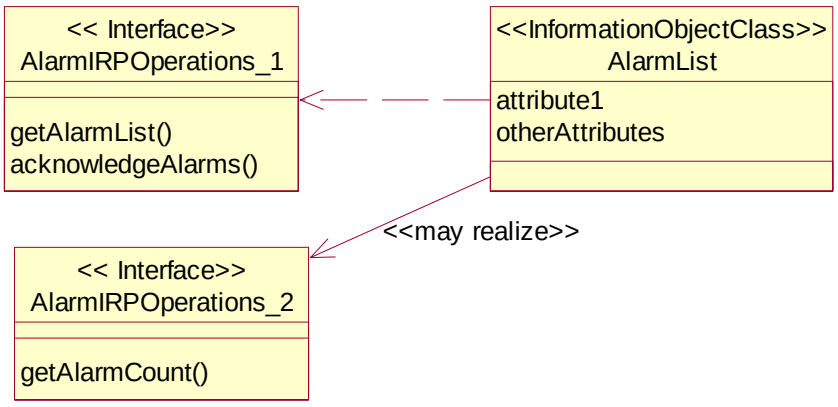
C.4.3.1 General

The relationship realize and <<may realize>> are unidirectional associations. The target must be an <<Interface>>. The relationship <<realize>> shows that the source entity must realize the operations defined by the target <<Interface>>. Realization of operations by the source entity is mandatory. The <<may realize>> shows the source entity may realize the operations defined by the target <<Interface>>. Realization of the <<Interface>> by the source entity is optional.

The operations defined by <<Interface>> are visible across the interface/reference point.

C.4.3.2 Sample

This shows that the AlarmList shall realize (or support, implement) the two operations of AlarmIRPOperations_1 and may realize the operation of AlarmIRPOperations_2.



Relationship realize and <<may realize>> Notations

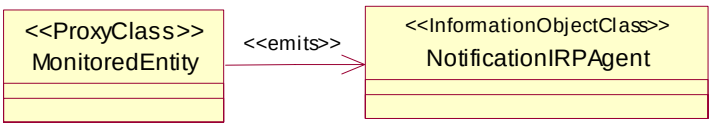
C.4.4 <<emits>>

C.4.4.1 General

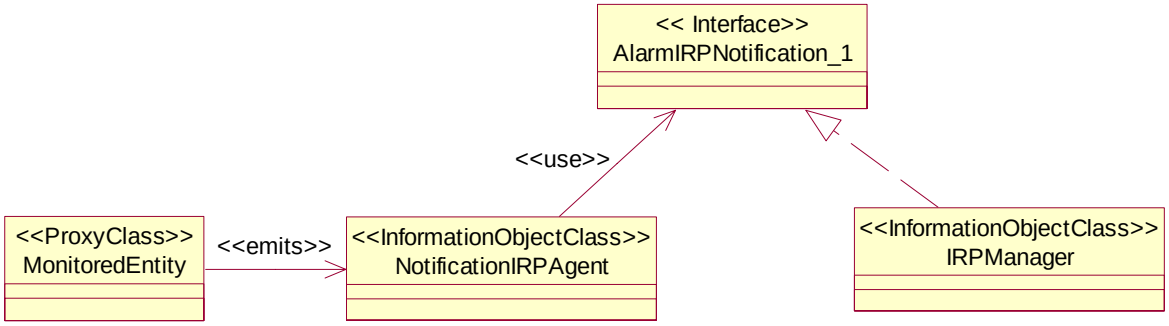
This is a unidirectional association. The source sends information to the target.

C.4.4.2 Sample

This shows the MonitoredEntity emits notifications that are received by the Notification Agent. The emission is not visible across the interface.



<<emits>> Notation



<<use>>, <<emits>> and realize relationship Notation

C.4.5 <<names>>

C.4.5.1 General

It specifies a unidirectional relationship. The target instance is uniquely identifiable, within the namespace of the source entity, among all other targeted instances of the same target classifier and among other targeted instances of other classifiers that have the same <<names>> composition with the source.

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

By convention, the name of the attribute in the target model element to hold part of the unique identification shall be formed by the name of the target class concatenated with "Id". There are two presentation options for the unique identification attribute of the class being named.

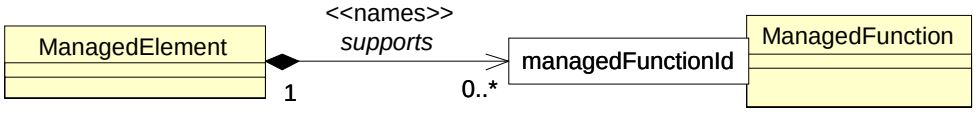
1) The use of the role qualifier allows the unique identification attribute to be attached to the target end of the <<names>> association (see the following figure).

2) The unique identification attribute may also be indicated as a normal attribute within the class attribute compartment.

   NOTE – The use of a single attribute for identification may be too restrictive. This issue is for further study.

C.4.5.2 Sample

This shows that all instances of ManagedFunction are uniquely identifiable within the ManagedElement namespace. Note the use of the label supports in specifications is optional.



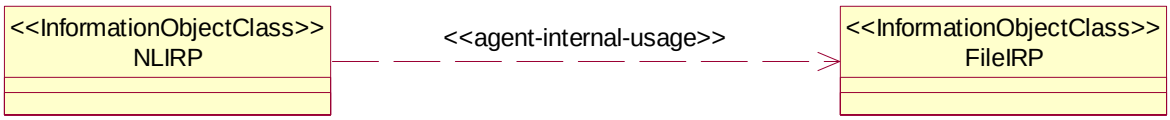
<<names>> Notation, Composition and explicit Qualifier

C.4.6 <<agent-internal-usage>>

This is a unidirectional association. The source passes network management information to target. The source and target are entities or processes running in different IRP instances, such as AlarmIRP, PMIRP. The instances may be name-contained by the same IRPAgent or different IRPAgent instances. The precise network management information passed and the information transfer mechanism are not standardized and are vendor-specific.

C.4.6.1 Sample

This shows that NLIRP (NotificationLog IRP) can pass some network management information to FileIRP.



<<agent-internal-usage>> Notation

C.5 Void

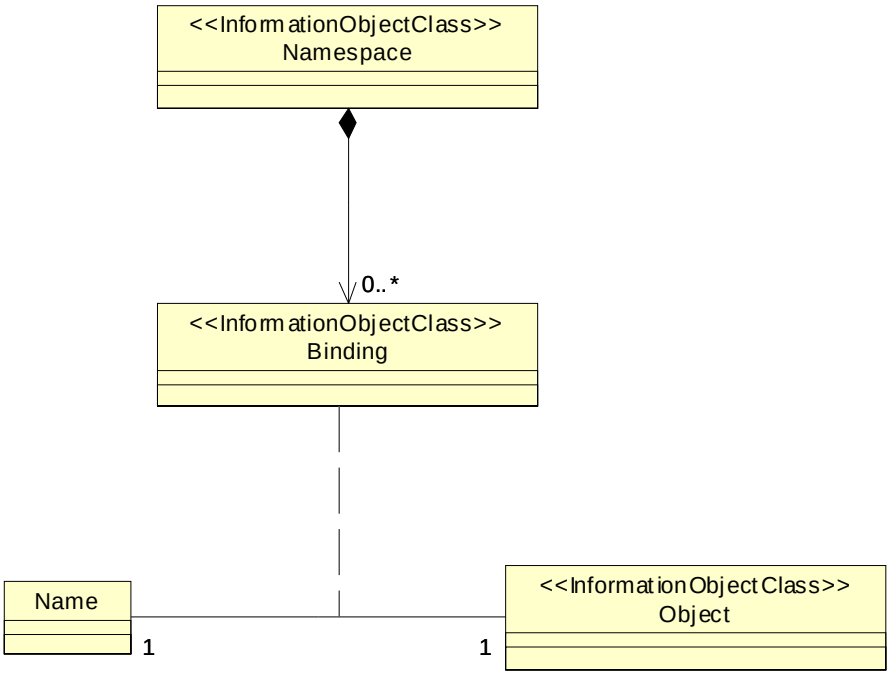
This clause is intentionally empty.

C.6 Association classes

Subclause 3.46 of [OMG UML] defines an association class as:

"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."

Association classes are appropriate for use when an "InformationObjectClass" needs to maintain associations to several other "InformationObjectClass"'s 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 object. A Binding "IOC" can be modelled as an Association Class that provides the binding semantics to the relationship between a name and some other "InformationObjectClass". This is depicted in the following figure (exemplary only, not taken from another Recommendation).



Example of an Association Class

C.7 Abstract class

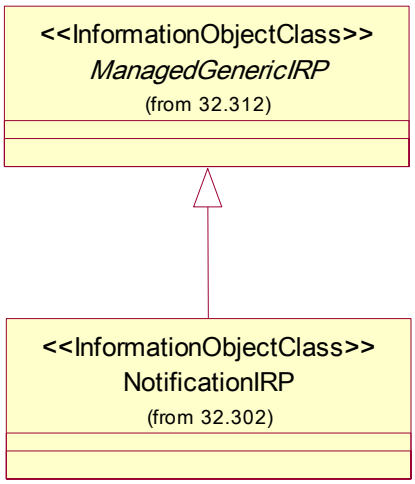
C.7.1 General

It specifies an <<InformationObjectClass>> as a base class to be inherited by subclasses. An abstract class cannot be instantiated.

Abstract class notation is the use of italics in the class name of the corresponding <<InformationObjectClass>> in the diagram.

C.7.2 Sample

This shows that ManagedGenericIRP is an abstract <<InformationObjectClass>>.



Abstract Class Notation

Annex D

Design

(This annex forms an integral part of this Recommendation)

This annex provides guidelines for specification of protocol specific designs.

For further study.

Appendix I

Requirements example

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

NOTE – The following example is based on alarm management, but is used for illustrative purposes only and it is not intended to be a complete or correct set of requirements for alarm management.

I.1 Concepts and background

Any evaluation of the NEs' and the overall network health status requires the detection of faults in the network and, consequently, the notification of alarms to the OS (EM and/or NM).

I.2 Business level requirements

I.2.1 Requirements

Faults that may occur in the network can be grouped into one of the following categories:

– Hardware failures, i.e., the malfunction of some physical resource within a NE.

– Software problems, e.g., software bugs, database inconsistencies.

I.2.1.1 Fault detection

REQ-FM-FUN-01 The majority of the faults should have well-defined conditions for the declaration of their presence or absence, i.e., fault occurrence and fault clearing conditions. Any such incident shall be referred to in this appendix as an ADAC fault. The network entities should be able to recognize when a previously detected ADAC fault is no longer present, i.e., the clearing of the fault, using similar techniques as they use to detect the occurrence of the fault.

I.2.1.2 Clearing of alarms

The alarms originated in consequence of faults need to be cleared. To clear an alarm, it is generally necessary to repair the corresponding fault.

...

REQ-FM-FUN-02 Each time an alarm is cleared, the agent shall generate an appropriate clear alarm event. A clear alarm is defined as an alarm.

I.2.1.3 Alarm forwarding and filtering

REQ-FM-FUN-03 For each detected fault, appropriate alarms (notifications of the fault) shall be generated by the faulty network entity.

...

I.2.2 Actor roles

Managed system The entity performing an agent role.

Managing system The entity performing the manager role.

I.2.3 Telecommunications resources

The managed network equipment is viewed as relevant telecommunications resources in this Recommendation.

I.2.4 High level use case diagrams

I.2.4.1 Report alarm

The first overview use case diagram in Figure I.1 shows the overall interaction of the alarm interface.

The first overview use case diagram shows the interactions involved in reporting a detected failure.

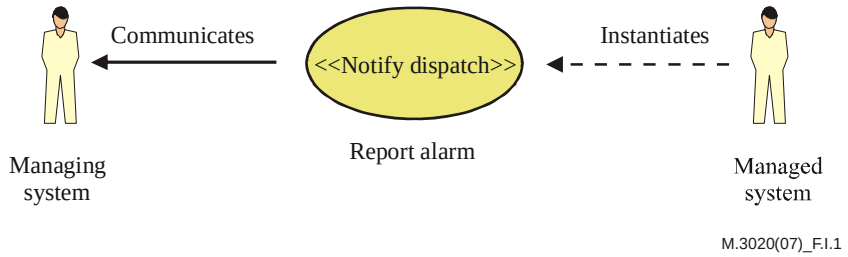


Figure I.1 – Report alarm

I.3 Specification level requirements

I.3.1 Requirements

There are no specification level requirements.

I.3.2 Actor roles

See clause I.2.2.

I.3.3 Telecommunications resources

See clause I.2.3.

I.3.4 Use cases

I.3.4.1 Fault notification

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal (*)	Upon detection of a failure condition, the managed system sends an alarm report notification, through interface Q, of the relevant type to the managing system.	
Actors and Roles (*)	The managing system is a consumer of notifications from the managed system.	
Telecom resources	Any managed entity	
Assumptions	A fault condition is detected.	
Pre-conditions	There is an open communication channel between the managing system and the managed system.	
Begins when	A fault condition is detected.	
Step 1 (*)	Upon detection of a failure condition, an appropriate alarm report or security alarm report is created.	
Ends when	Alarm report or security alarm report is emitted by the agent.	

Use case stage	Evolution/Specification	<<Uses>> Related use
Exceptions	Communication or process failure could result in a failure to deliver the alarm report to the managing system. The alarm synchronization use case covers this situation.	
Post-conditions	The managing system is informed of the fault condition in the managed system.	
Traceability (*)	REQ-FM-FUN-01, REQ-FM-FUN-02, ...	

I.3.4.2 Alarm clear

...

I.3.4.3 Acknowledge alarm

...

Appendix II

Analysis example

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

NOTE – The following example is based on alarm management, but is used for illustrative purposes only, and it is not intended to be a complete or correct set of requirements for alarm management.

II.1 Concepts and background

Any evaluation of the NEs' and the overall network health status requires the detection of faults in the network and, consequently, the notification of alarms to the OS (EM and/or NM).

...

II.2 Information object classes

II.2.1 Information entities imported and local label

Label reference	Local label
3GPP TS 32.302, information object class, NotificationIRP	NotificationIRP
3GPP TS 32.302, interface, notificationIRPNotification	NotificationIRPNotification
3GPP TS 32.622, information object class, IRPAgent	IRPAgent
3GPP TS 32.312, information object class, ManagedGenericIRP	ManagedGenericIRP

II.2.2 Class diagram

This clause introduces the set of information object classes (IOCs) that encapsulate information within the agent. The intent is to identify the information required for the AlarmAgent implementation of its operations and notification emission. This clause provides the overview of all support object classes in UML. Subsequent clauses provide more detailed specification of the various aspects of these support object classes.

II.2.2.1 Attributes and relationships

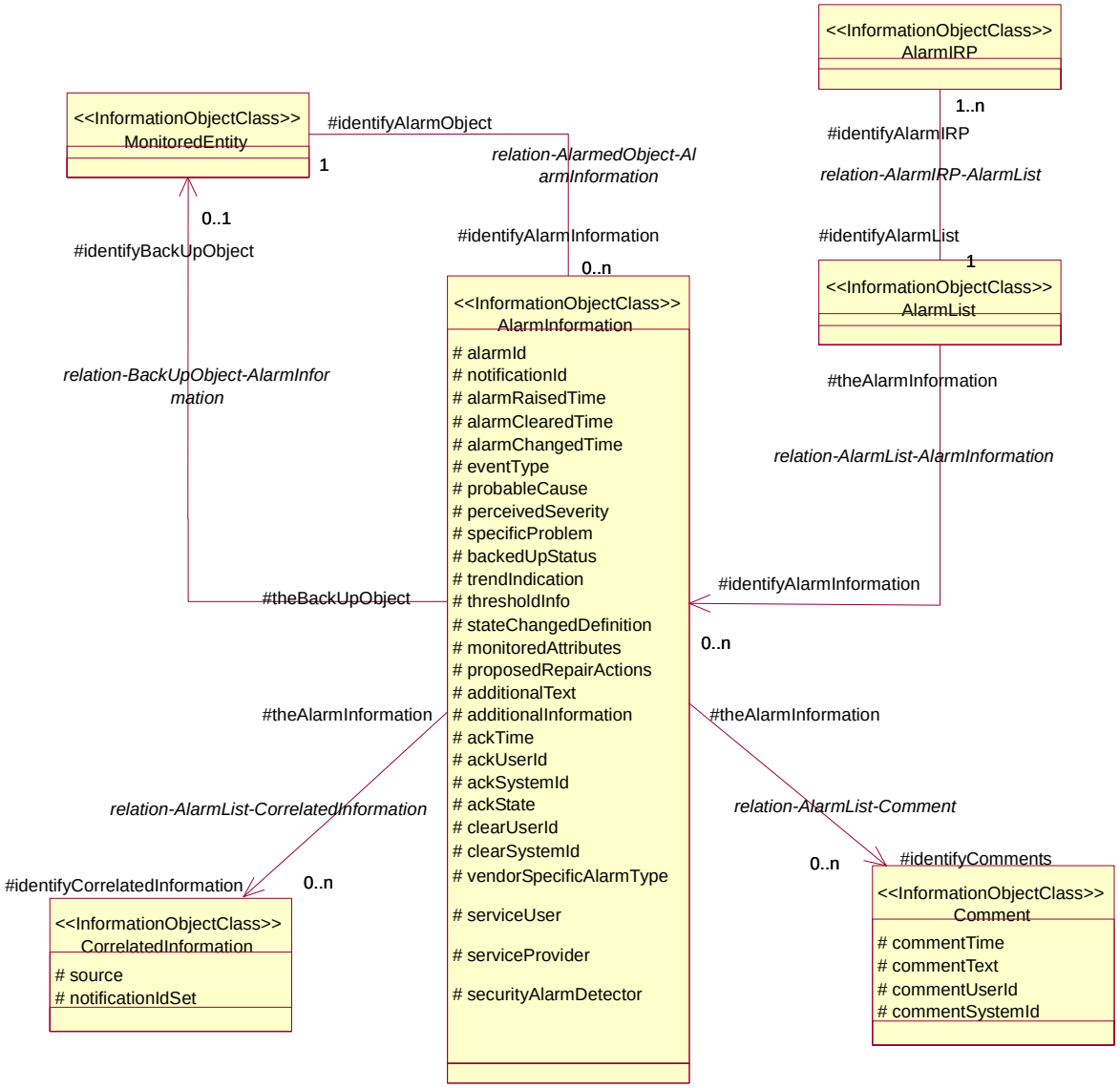


Figure II.1 – Alarm management information object classes

II.2.2.2 Inheritance

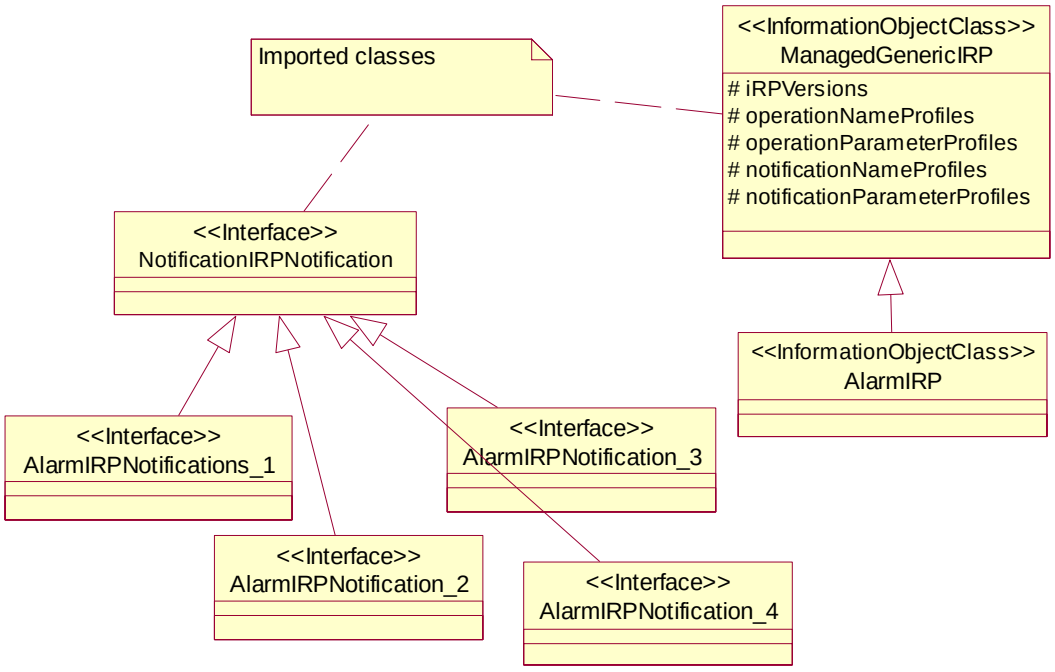


Figure II.2 – Alarm management IOC inheritance

II.2.3 Information object class definitions

Class name	Qualifier	Requirement IDs
`AlarmInformation`	M	REQ-FM-FUN-01, REQ-FM-FUN-02, …
`AlarmList`	M	REQ-FM-FUN-n
...		

II.2.3.1 AlarmInformation

II.2.3.1.1 Definition

AlarmInformation contains information about alarm condition of an alarmed MonitoredEntity.

...

II.2.3.1.2 Attributes

Attribute name	Support Qualifier	Read Qualifier	Write Qualifier	Requirement IDs
alarmed	M	M	M	
probableCause	C	M	C	
structuredProbableCause	C	M	C	
perceivedSeverity	M	M	M	
specificProblem	O	O	O	
...				
...				

II.2.3.1.3 State diagram

Alarms have states.

...

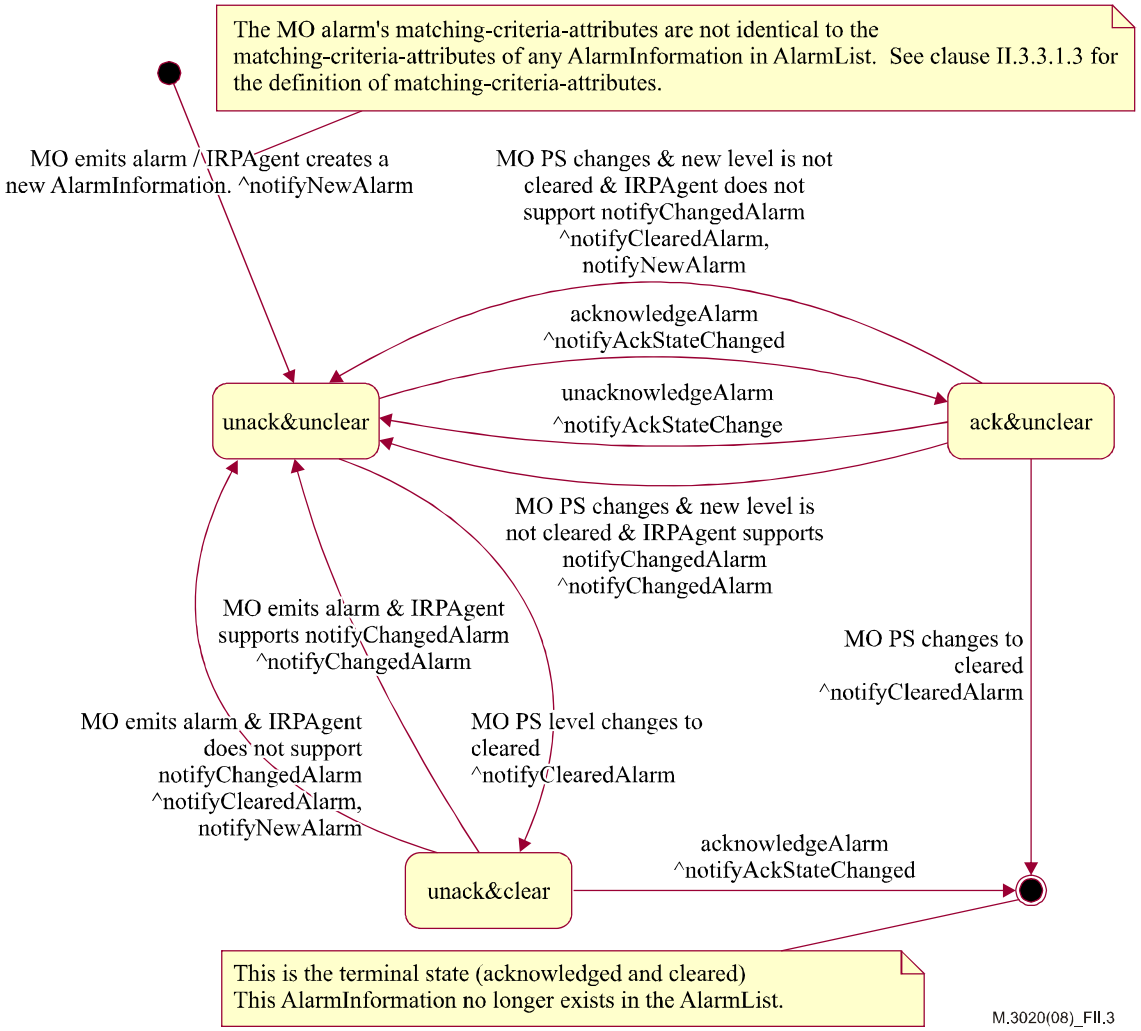


Figure II.3 – Alarm information state diagram

II.2.3.2 AlarmList

II.2.4 Information relationships definition

Relationship	Support Qualifier	Requirement IDs
relation-AlarmIRP-AlarmList	M	REQ-FM-FUN-x
...		

II.2.4.1 relation-AlarmIRP-AlarmList (M)

II.2.4.1.1 Definition

This represents the relationship between AlarmIRP and AlarmList.

II.2.4.1.2 Roles

Name	Definition
identifyAlarmIRP	It represents the capability to obtain the identities of one or more AlarmIRP.
identifyAlarmList	It represents the capability to obtain the identity of one AlarmList.

II.2.4.1.3 Constraint

There is no constraint for this relationship.

II.2.4.2 relation-AlarmList-AlarmInformation (M)

...

II.2.5 Information attribute definition

II.2.5.1 Definition and legal values

Name	Definition	Legal Values
alarmed	It identifies one AlarmInformation in the AlarmList.	
notificationId	It identifies the notification that carries the AlarmInformation.	

II.2.5.2 Constraints

Name	Affected attribute(s)	Definition
inv_notificationId	notificationId	NotificationIds shall be chosen to be unique across all notifications of a particular managed object (representing the NE) throughout the time that alarm correlation is significant. The algorithm by which alarm correlation is accomplished is outside the scope of this IRP.

II.3 Interface definition

II.3.1 Class diagram representing interfaces

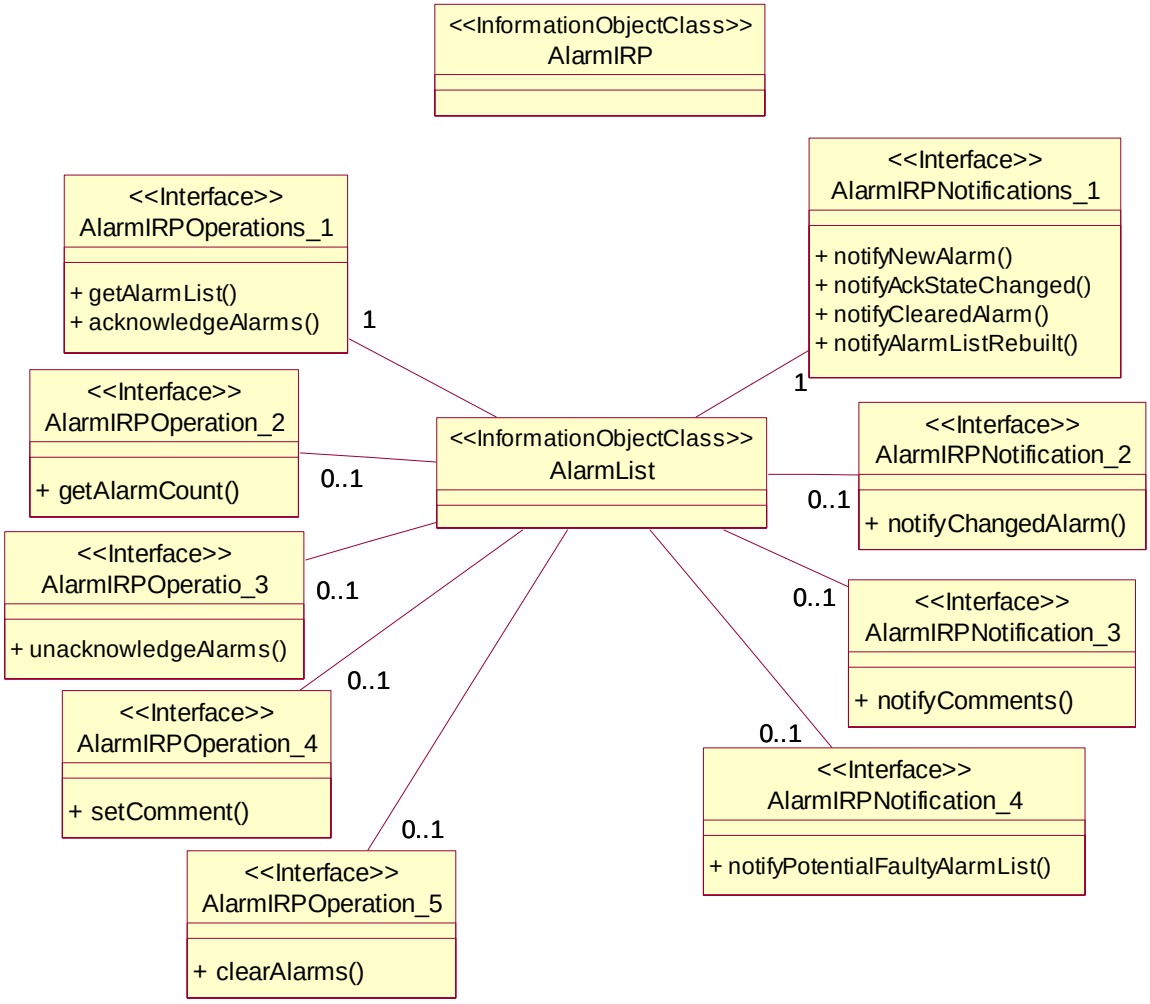


Figure II.4 – Alarm management IRP class diagram

II.3.2 Generic rules

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 that is raised when an internal problem occurs and that the operation cannot be completed. The exception has the same entry and exit state.

II.3.3 Interface AlarmIRPOperations_1 (O)

Operation Name	Qualifier	Requirement IDs
acknowledgeAlarms	M	REQ-FM-FUN-x, REQ-FM-FUN-y
getAlarmList	M	...

II.3.3.1 Operation acknowledgeAlarms (M)

II.3.3.1.1 Definition

The `Manager` invokes this operation to acknowledge one or more alarms.

II.3.3.1.2 Input parameters

Name	Qualifier	Information Type	Comment
...			
ackUserId	M	`AlarmInformation.ackUserId`	It identifies the user acknowledging the alarm.
ackSystemId	O	`AlarmInformation.ackSystemId`	It identifies the processing system on which the subject `IRPManager` runs. It may be absent implying that `IRPManager` does not wish this information be kept in `AlarmInformation` in `AlarmList`.

II.3.3.1.3 Output parameters

Name	Qualifier	Matching Information	Comment
...			
Status	M	ENUM (OperationSucceeded, OperationFailed, OperationPartiallySucceeded)	If someAlarmAcknowledged is true, status = OperationPartiallySucceeded. If allAlarmsAcknowledged is true, status = OperationSucceeded. If operation_failed is true, status = OperationFailed.

II.3.3.1.4 Pre-condition

`atLeastOneValidId`.

Assertion Name	Definition
atLeastOneValidId	The AlarmInformationReferenceList contains at least one identifier that identifies one AlarmInformation in AlarmList and that this identified AlarmInformation shall have its ackState indicating "unacknowledged" and, if provided, an equal perceivedSeverity.

II.3.3.1.5 Post-condition

`someAlarmAcknowledged OR allAlarmsAcknowledged.`

Assertion Name	Definition
someAlarmAcknowledged	...
allAlarmsAcknowledged	...

II.3.3.1.6 Exceptions

Name	Definition
operation_failed	**Condition**: Pre-condition is false or post-condition is false. **Returned Information**: The output parameter status. **Exit state**: Entry state.

II.3.3.2 Operation getAlarmList (M)

...

Appendix III

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], Data Architecture, that is used for the development of Recommendations in the M.1400 series of ITU-T Recommendations.

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 IV

Issues for further study

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

This appendix identified known issues that are subject for further study.

IV.1 SOA

The approval of [ITU-T M.3060], Principles for the management of next generation networks, signalled a change from an object-oriented to a service-oriented approach to management. The impact of this change will need to be studied to identify any changes required in future revisions of this Recommendation (M.3020).

IV.2 UML

This version of M.3020 references UML version 1.5 in order to maintain alignment with the corresponding 3GPP specifications. A revised M.3020 should reference later versions of UML:

– The OMG MOF meta-meta model integrates UML 2.x as a meta-model which is supported by the mainstream industry tool vendors. Prior to UML 2.0, there was no overarching meta-meta model and UML itself was not standard. MOF supports the addition and creation of other new meta-models defined in a precise way via OCL which is a predicate calculus language.

– Both industry (telecoms, governments and military) and tool vendors are converging on the OMG MOF model.

– The benefits of the MOF meta-meta model are that it supports a family of meta-models which can be used to define object models, HCI relationships, various technology-specific implementations and allows transforms between models to be undertaken in a standard way. This is not achievable in UML 1.5 since UML 1.5 exists in isolation of a higher meta-model.

IV.3 Visibility

It has been suggested that the default visibility should be private for attributed and public for operations in order to promote data encapsulation and reduce time and effort in defining the implementation model.

IV.4 Type definitions

When writing a new specification based on this methodology, it is necessary to specify the types of parameters and attributes. Formal type definitions are absent from the current version of M.3020, so the definition of types might be different and inconsistent for the same meaning in different specifications, e.g., for an array of integer, it might be defined as a list of integers, or a sequence of integers, or a set of integers.

It is suggested to add a new clause in Annex B with type definitions that can be used to define attributes and parameters. The new clause should also handle the definition of complex types.

Appendix V

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

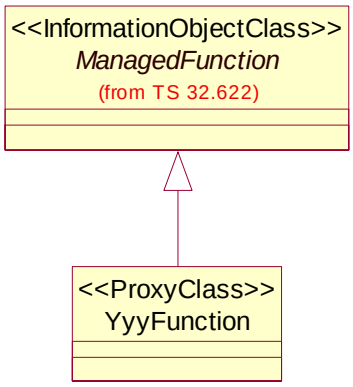
This appendix contains additional samples on use of the UML described in Annex C.

V.1 Proxy class

V.1.1 First sample

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 sample, inherit from ManagedFunction IOC.

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.

<<ProxyClass>> Notation Sample V.1

V.1.2 Second sample

This 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 subsection 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 subsection 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].

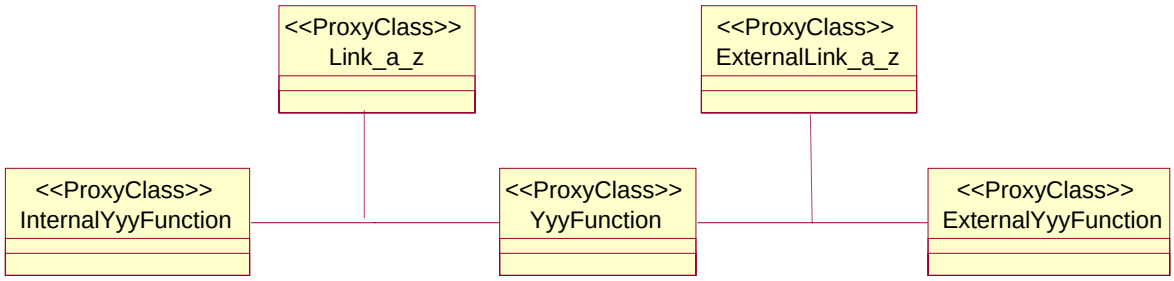
The linked InternalYyyFunction <<ProxyClass>> represents SlsFunction, CscfFunction, HlrFunction ...

The linked ExternalYyyFunction <<ProxyClass>> represents ...

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

The ExternaLink_a_z <<ProxyClass>> represents ...

"



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

`<<ProxyClass>> Notation Sample V.2`

Bibliography

This Bibliography contains 3GPP specifications used for the production of this Recommendation but they are not normative references.

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

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

[b-ITU-T Z.100] Recommendation ITU-T Z.100 (2007), *Specification and Description Language*.

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

[b-3GPP TS 23.002] 3GPP TS 23.002, in force, *Network architecture*. <http://www.3gpp.org/ftp/specs/html-info/23002.htm>

[b-3GPP TS 32.101] 3GPP TS 32.101 V8.1.0 (2007), *Telecommunication management; Principles and high level requirements*. <http://www.3gpp.org/ftp/specs/html-info/32101.htm>

[b-3GPP TS 32.150] 3GPP TS 32.150 V8.1.0 (2007), *Telecommunication management; Integration Reference Point (IRP) Concept and definition.* <http://www.3gpp.org/ftp/Specs/html-info/32150.htm>

[b-3GPP TS 32.151] 3GPP TS 32.151 V8.0.0 (2007), *Telecommunication management; Integration Reference Point (IRP) Information Service (IS) template*. <http://www.3gpp.org/ftp/Specs/html-info/32151.htm>

[b-3GPP TS 32.152] 3GPP TS 32.152 V8.0.0 (2007), *Telecommunication management; Integration Reference Point (IRP) Information Service (IS) Unified Modelling Language (UML) repertoire*. <http://www.3gpp.org/ftp/Specs/html-info/32152.htm>

[b-3GPP TS 32.302] 3GPP TS 32.302 V7.0.0 (2007), *Telecommunication management; Configuration Management (CM); Notification Integration Reference Point (IRP): Information Service (IS)*. <http://www.3gpp.org/ftp/specs/html-info/32302.htm>

[b-3GPP TS 32.312] 3GPP TS 32.312, in force, *Telecommunication management, generic Integration Reference Point (IRP) management; Information Service (IS).* <http://www.arib.or.jp/IMT-2000/V720Mar09/5_Appendix/Rel4/32/32312-410.pdf>

[b-3GPP TS 32.622] 3GPP TS 32.622, in force, *Telecommunication management; Configuration Management (CM); generic network resources Integration Reference Point (IRP): Network Resource Model*. <http://www.3gpp.org/ftp/Specs/html-info/32622.htm>

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
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	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
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 and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2009