International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.819

(01/2022)

SERIES Q: SWITCHING AND SIGNALLING, AND ASSOCIATED MEASUREMENTS AND TESTS

Q3 interface

REST-based management services

Recommendation ITU-T Q.819

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING, AND ASSOCIATED MEASUREMENTS AND TESTS

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4, 5, 6, R1 AND R2	Q.120–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.799
Q3 INTERFACE	**Q.800–Q.849**
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
SPECIFICATIONS OF SIGNALLING RELATED TO BEARER INDEPENDENT CALL CONTROL (BICC)	Q.1900–Q.1999
BROADBAND ISDN	Q.2000–Q.2999
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR THE NGN	Q.3000–Q.3709
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR SDN	Q.3710–Q.3899
TESTING SPECIFICATIONS	Q.3900–Q.4099
PROTOCOLS AND SIGNALLING FOR PEER-TO-PEER COMMUNICATIONS	Q.4100–Q.4139
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2020	Q.5000–Q.5049
COMBATING COUNTERFEITING AND STOLEN ICT DEVICES	Q.5050–Q.5069

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

Recommendation ITU-T Q.819

REST-based management services

Summary

Recommendation ITU-T Q.819 defines a set of services required to support REST-based interfaces and, with Recommendation ITU-T X.785, composes a framework for REST-based network management interfaces. It specifies protocol requirements and defines some network management-specific support services, which are notification service, heartbeat service and containment service. The JSON/YAML interface definitions for the network management-specific support services are also provided.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Q.819	2022-01-13	2	11.1002/1000/14835

Keywords

5G, AI, energy saving, telecom operation management.

---

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

FOREWORD

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

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

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

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

NOTE

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

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

INTELLECTUAL PROPERTY RIGHTS

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

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents/software copyrights, 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 appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Recommendation ITU-T Q.819

REST-based management services

1 Scope

The network management architecture defined in [ITU-T M.3010] introduces the use of multiple management protocols. So far, GDMO/CMIP, CORBA GIOP/IIOP, SMI/SNMP and Web Service/SOAP are possible choices at the application layer. Based on the management interface specification methodology defined in [ITU-T M.3020], more technology-based paradigms can be introduced into network management interfaces, and representational state transfer/hyper text transfer protocol (REST/HTTP) is now an additional paradigm for network management.

This Recommendation, together with [ITU-T X.785], sets out to define a framework for defining how interfaces supported by management systems and network elements should be modelled using the REST / JavaScript object notation (JSON) schema. It is within the scope of this Recommendation to provide the following guidelines or instructions:

– protocol requirements for using REST in network management;

– how a REST-based notification service is used in network management interfaces;

– how to monitor the availability of a REST notification forwarding mechanism;

– how to access containment information through a REST containment service;

– compliance and conformance requirements.

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.3020] Recommendation ITU-T M.3020 (2017), *Management interface specification methodology*.

[ITU-T M.3702] Recommendation ITU-T M.3702 (2010), *Common management services – Notification management – Protocol neutral requirement and analysis*.

[ITU-T X.701] Recommendation ITU-T X.701 (1997*), Information technology – Open Systems Interconnection – Systems management overview*.

[ITU-T X.703] Recommendation ITU-T X.703 (1997), *Information technology – Open Distributed Management Architecture*.

[ITU-T X.721] Recommendation ITU-T X.721 (1992), *Information technology – Open Systems Interconnection – Structure of management information: Definition of management information*.

[ITU-T X.785] Recommendation ITU-T X.785 (2021), *Guidelines for defining REST-based managed objects and management interfaces*.

[IETF RFC 3986] IETF RFC 3986 (2005), *Uniform Resource Identifier (URI): Generic Syntax*.

[IETF RFC 6585] IETF RFC 6585 (2012), *Additional HTTP Status Codes*.

[IETF RFC 7230] IETF RFC 7230 (2014), *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*.

[IETF RFC 7231] IETF RFC 7231 (2014), *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*.

[IETF RFC 7232] IETF RFC 7232 (2014), *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*.

[IETF RFC 7540] IETF RFC 7540 (2015), *Hypertext Transfer Protocol Version 2 (HTTP/2)*.

[IETF RFC 8259] IETF RFC 8259 (2017), *The JavaScript Object Notation (JSON) Data Interchange Format (JSON)*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 agent [ITU-T M.3020]: 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.

3.1.2 managed object class [ITU-T X.701]: A named set of managed objects sharing the same (named) sets of attributes, notifications, management operations (packages), and which share the same conditions for presence of those packages.

3.1.3 manager [ITU-T M.3020]: 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.

3.1.4 notification [ITU-T X.703]: An interaction for which the contract between the invoking object (client) and the receiving object (server) is restricted to the ability of the server to receive the contents of information sent by the client.

3.2 Terms defined in this Recommendation

None.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

CMIP Common Management Information Protocol

CORBA Common Object Request Broker Architecture

CRUD Create, Read, Update and Delete

DN Distinguished Name

GDMO Guidelines for the Definition of Managed Objects

GIOP General Inter-ORB Protocol

HTTP Hyper Text Transfer Protocol

IIOP Internet Inter-ORB Protocol

JSON JavaScript Object Notation

MO	Managed Object
REST	Representational State Transfer
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
TMN	Telecommunications Management Network
URI	Uniform Resource Identifier
URL	Unified Resource Location
XML	extensible Markup Language
YAML	YAML Ain't Markup Language

5 Conventions

A few conventions are followed in this Recommendation to make the reader aware of the purpose of the text. Two qualifiers are used, which are:

M, which stands for mandatory and means that the attribute or field is mandatory for an operation or notification.

O, which stands for optional and means that the attribute or field is optional for an operation or notification.

Examples of YAML and JSON are included in this Recommendation and normative YAML or JSON schema examples specifying the data types, base classes and other modelling constructs of the framework are included in Annex A. The JSON/YAML schemas are written in a 10 point courier typeface:

A JSON schema example	<pre>{
	    "title": "root",
	    "items": {
	        "title": "array item"
	    }
	}</pre>

A YAML schema example	<pre>SomeType:
	  type: object
	    properties:
	      attr1:
	        type: string
	      attr2:
	        type: string
	        enum:
	          - e1
	          - e2</pre>

6 REST-based network management services – goals and requirements

This clause describes the key goals of the services framework and the requirements that help the REST-based network management services support these goals. Clause 6.1 introduces the goals of the REST-based services framework. Clause 6.2 then provides terminology and requirements. The requirements in this clause are requirements that the framework must satisfy and that are based on telecommunications management needs. Clauses 7, 8 and 9 then describe a framework that meets these needs and defines how to achieve the requirements of this clause by using REST-based services in a certain way.

6.1 Goals

This Recommendation sets out a framework for defining how interfaces supported by management systems and network elements should be modelled. Some key goals of the framework are identified here:

– application interoperability;

– common REST-based support services.

This clause elaborates on these two goals.

6.1.1 Application interoperability

A key goal of the network management architecture, and in particular the information architecture, is to promote a standard framework for providing interoperability and information exchange between systems from a diverse set of network management system suppliers. Interoperability between systems involves many aspects of development. At its lowest layer, a common communication mechanism must be in place to support a common syntax, the establishment of connectivity and the exchange of operation requests/replies between systems. This aspect of interoperability is inherently supported by the REST/HTTP specifications.

For network management, there is a need to provide application interoperability. That is, management systems from diverse suppliers will be utilized within a single administration's telecommunications management network (TMN) to support the different functions necessary to support the management of its networks. To simplify the integration of these various suppliers' systems, they must agree on the semantics of the information being exchanged. This is accomplished with the specification of an information model. Guidelines for the definition of REST-based information models are specified in [ITU-T X.785], but the services defined here should also support those guidelines.

6.1.2 Common REST-based support services

A second aspect of this framework is the definition of the common usage and profiling of the distributed processing environment of choice. This aspect of the framework should indicate the reasonable expectations network management system suppliers may have of one another. Rather than redefining the interface capabilities needed to support common network management functions such as a notification service with each information model, the modelling guidelines in [ITU-T X.785] rely upon a set of support services. These support services enable the information models to be simpler, and also enhance interoperability.

6.2 Information modelling dependencies

As described in the previous clause, the explicit modelling of resources that are manageable across an interface is central to application interoperability. The guidelines for defining REST-based managed objects (MOs) detailed in [ITU-T X.785] describe the rules for modelling manageable resources. They also embody several decisions that must be supported by the network management REST-based services framework. This clause summarizes those points.

6.2.1 Access granularity

REST-based service interface granularity refers to the relationship between the resources that are modelled on an interface and the means by which they are accessed using REST-based services. The objects that represent manageable resources are called MOs. Recommendation [ITU-T X.785] uses a service-grain modelling approach which means each modelled resource is only accessible through either the generic MO access service, or a user defined REST service specific for various MO classes.

6.2.2 Representation of containment and naming

Containment is a logical representation of how modelled resources contain other modelled resources. Containment has traditionally been a very important relationship in network management applications because it is a convenient means of identifying the large number of resources that typically must be managed. The [ITU-T X.785] guidelines require that a unique name be assigned to each MO, based in part on the name of the object that contains it. The REST-based management services must provide the means to store these names (and hence the containment relationships they represent).

6.2.3 Object creation and deletion

The REST-based platforms do not provide clients with the particular means to create or delete objects on remote systems. Instead, these functionalities are provided by remote systems for clients to create or delete objects on the remote system. Recommendation [ITU-T X.785] provides a common object accessing service, so object creation and deletion can be model independent. When deleting an object that contains other objects, the containment relationship shall be retained, so that there will be no orphan objects existing in the remote systems. The containment information should be stored in the remote system and be maintained when creation and deletion take place.

6.3 Notifications

The framework needs to support the ability to:

– deliver notifications;

– subscribe for notification types;

– forward notifications to multiple destinations;

– uniquely identify the resource that emits the notification.

7 Framework overview and protocol requirements

7.1 Framework overview

This framework for REST-based management interfaces is a collection of capabilities. The framework defines the support services to be standards on network management interfaces conforming to this framework. The REST interfaces for these services will be defined in Annex A.

To support the software objects representing manageable resources, the framework requires that they implement some common basic capabilities. Therefore, one base class (YAML/JSON-based) is defined in [ITU-T X.785] for use in modelling network management resources. MO classes in information models must inherit and implement a basic set of capabilities from the base class in order to operate within this framework. Finally, some rules and conventions are defined for information modellers developing models for use with this framework. These consist of modelling guidelines and YAML/JSON style idioms. All of these are depicted graphically in Figure 1.

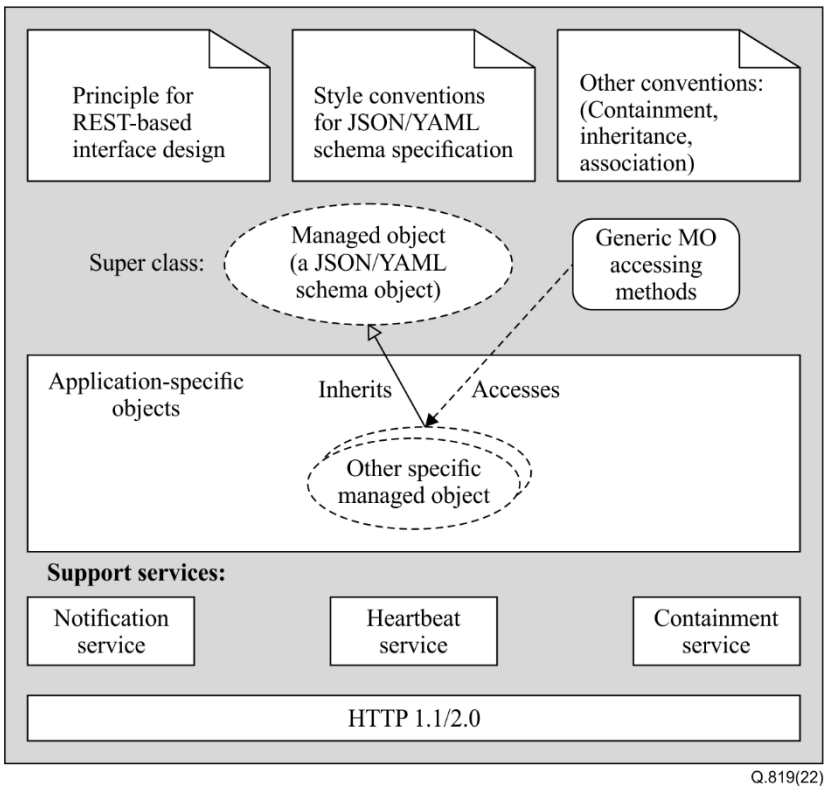

Figure 1 – Framework overview

Figure 1 shows the framework in grey. In the middle are the application-specific objects that are supported by the framework. Along the bottom is a box representing the communication protocol, HTTP 1.1 or 2.0. Above that are a number of boxes with names in them representing the services that compose this framework. Along the top of the figure are icons representing the super class ManagedObject, and the generic MO accessing methods. Each MO supported by this framework must ultimately inherit from the super class. Also shown in Figure 1 are squares with downturned corners representing standard object modelling conventions.

The framework services, represented as boxes with square corners, are defined in this Recommendation. The super class, object modelling conventions and other conversions are defined in [ITU-T X.785].

7.2 Protocol, languages and services requirements

This clause defines the versions of the languages and protocols that are required to support this framework. REST-based technology and protocol specifications are defined by IETF and OAI. Table 1 shows which version (including subsequent versions released up until the publication date of this Recommendation) of the applicable specifications must be supported to comply with this framework.

Table 1 – REST-based language, protocol and versions

Language and protocol	Version	References
YAML Schema (OAI)	3.0	[b-OAI-OAS3]
JSON Schema Core (IETF)	Draft 2020-12 (Draft-08-patch-1, December, 2020)	[IETF RFC 8259]
HTTP protocol (IETF)	1.1 (2014) 2.0 (August, 2017)	[IETF RFC 7230], [IETF RFC 7231],[IETF RFC 7232], [IETF RFC 7540]

8 REST-based notification service and notification format

8.1 REST-based notification service overview

REST-base notification service provides a standard approach for notification consumers and providers to transfer notification information using a topic-based publish/subscribe pattern. The contents of the notification service include standard message exchanges to be implemented by notification providers, operational requirements expected of service providers and requesters that participate in notifications and an JSON model that describes topics.

The REST-based notification service supports the asynchronous exchange of event messages between clients using a subscribe-and-publish paradigm.

8.2 Mapping of operations from [ITU-T M.3702]

Recommendation [ITU-T M.3702] provides protocol neutral requirements and analysis for the generic notification management service, which defines the generic functions that are to be implemented by a protocol-specific management paradigm. In this framework, the generic notification management model is mapped to the JSON/HTTP definition.

Table 2 indicates the mapping from [ITU-T M.3702] to the REST-based notification service.

Table 2 – Notification operations mapping from [ITU-T M.3702] to this Recommendation

No.	[ITU-T M.3702]		[ITU-T Q.819]	
	Operation name		Operation name	
	Request/ response	Parameters	Request/ response	Parameters
1	subscribeNotification		HTTP POST	
	Input parameter	– managerId – notificationTypes – filteringCriteria – destination	Request	"POST" {notificationServiceURI}/"subscriptions" {HTTPVersion} request body schema: $ref: "notificationApiDataSchema.json#/definitions/ SubscribeNotificationRequest"
	Output parameter	– subscriptionId – status	Response	{HTTPVersion} {statusCode} {reasonPhrase} Location: {notificationServiceURI}/"subscriptions"/ {subscriptionId}{response body}
2	unsubscribeNotification		HTTP DELETE	
	Input parameter	– managerId – subscriptionId	Request	"DELETE" {notificationServiceURI}/"subscriptions"/ {subscriptionId} {HTTPVersion}
	Output parameter	– status	Response	{HTTPVersion} {statusCode} {reasonPhrase} {response body}
3	suspendSubscription		HTTP POST	
	Input parameter	– managerId – subscriptionId	Request	"POST" {notificationServiceURI}/"subscriptions"/ {subscriptionId}/"suspendSubscription" {HTTPVersion}
	Output parameter	– status	Response	{HTTPVersion} {statusCode} {reasonPhrase} {response body} Response body schema: $ref: "notificationApiDataSchema.json#/definitions/ NotificationInfo"

Table 2 – Notification operations mapping from [ITU-T M.3702] to this Recommendation

No.	[ITU-T M.3702]		[ITU-T Q.819]	
	Operation name		Operation name	
	Request/ response	Parameters	Request/ response	Parameters
4	resumeSubscription		HTTP POST	
	Input parameter	– managerId – subscriptionId	Request	"POST" {notificationServiceURI}/"subscriptions"/ {subscriptionId}/"resumeSubscriptions" {HTTPVersion}
	Output parameter	– status	Response	{HTTPVersion} {statusCode} {reasonPhrase} {response body}
5	getNotificationTypes		HTTP GET	
	Input parameter	– notificationIRPId	Request	"GET" {notificationServiceURI}/NotificationTypes {HTTPVersion}
	Output parameter	– status – notificationType List	Response	{HTTPVersion} {statusCode} {reasonPhrase} {response body} Response body schema: $ref:"notificationApiDataSchema.json# /definitions/GetNotificationTypesResponse"
6	querySubscription		HTTP GET	
	Input parameter	– subscriptionId	Request	"GET" {notificationServiceURI}/"subscriptions"/ {subscriptionId}{HTTPVersion}
	Output parameter	– subscribed NotificationType – subscriptionStatus – destination – filteringCriteria – status	Response	{HTTPVersion} {statusCode} {reasonPhrase} {response body} Response body schema: $ref: "notificationApiDataSchema.json# /definitions/QuerySubscriptionResponse"
7	modifySubscription		HTTP PATCH	
	Input parameter	– subscriptionId – filteringCriteria – destination – notificationTypes	Request	"PATCH" {notificationServiceURI}/"subscriptions"/ {subscriptionId} {HTTPVersion}{request body} Request body schema: $ref:"notificationApiDataSchema.json#/definitions/ModifySubscriptionRequest"
	Output parameter	– status	Response	{HTTPVersion} {statusCode} {reasonPhrase} {response body} Response body schema: $ref: "notificationApiDataSchema.json# /definitions/NotificationInfo"
8	listAllSubscriptionIds		HTTP GET	
	Input parameter	– managerId	Request	"GET" {notificationServiceURI}/"subscriptions"/ listAllSubscriptionIds {HTTPVersion} {request body} Request body schema: $ref: "notificationApiDataSchema.json# /definitions/ListAllSubscriptionsIdsRequest"
	Output parameter	– subscriptionIdSet – status	Response	{HTTPVersion} {statusCode} {reasonPhrase} {response body} Response body schema: $ref:"notificationApiDataSchema.json#/definitions/ListAllSubscriptionsIdsResponse"

Table 2 – Notification operations mapping from [ITU-T M.3702] to this Recommendation

No.	[ITU-T M.3702]		[ITU-T Q.819]	
	Operation name		Operation name	
	Request/ response	Parameters	Request/ response	Parameters
9	report notification:		HTTP POST (from provider to consumer)	
			Request	"POST" {destinationURI} {HTTPVersion} {request body} Request body schema: $ref: "notificationApiDataSchema.json# /definitions/NotifyRequest"
			Response	{HTTPVersion} {statusCode} {reasonPhrase} {response body}

A detailed interface definition using YAML can be found in clause A.1.

8.3 Notification format definitions

8.3.1 Notification format to be used in this Recommendation

This clause will provide a description for the notification format to be used in this Recommendation.

8.3.2 Common notification header definition

This clause provides the detailed common notification header described in [ITU-T M.3702].

Table 3 provides the definition of the parameters of the common notification header.

Table 3 – Common notification header definition

Parameter name	Qualifiers	JSON/YAML-schema type	Descriptions
objectClass	M	`string`	It is the class name of the MO which sends out this notification. See clause 7.3.5 of [ITU-T M.3702]
objectInstance	M	`uri`	It is the distinguished name (DN) of the MO which sends out this notification. See clause 7.3.5 of [ITU-T M.3702]
notificationId	M	`string`	It is the identifier of the notification for the purpose of notification correlation. See clause 7.3.5 of [ITU-T M.3702]
eventTime	M	`string`	It is the time when this event happened. See clause 7.3.5 of [ITU-T M.3702]
systemDN	M	`type:string` `format:uri`	It is the DN of the system which sends out this notification. See clause 7.3.5 of [ITU-T M.3702]

Table 3 – Common notification header definition

Parameter name	Qualifiers	JSON/YAML-schema type	Descriptions

notificationType	M	`NotificationType:`
		`  type: string`
		`  enum:`
		`  - objectCreation`
		`  - objectDeletion`
		`  - attributeValueChange`
		`  - stateChange`
		`  - communicationAlarm`
		`  - environmentalAlarm`
		`  - equipmentAlarm`
		`  - processingErrorAlarm`
		`  - qualityOfServiceAlarm`
		`  - integrityViolation`
		`  - operationalViolation`
		`  - physicalViolation`
		`  - securityViolation`
		`  - timeDomainViolation`
		`  - relationshipChange`
		`  - heartbeat`
			It indicates the type of notification. See clause 7.3.5 of [ITU-T M.3702].
			The possible common notification types defined in this Recommendation can be found in the list in the left column.
			This list can be extended to include new notification types when they are defined.

8.3.3 Common notification contents definition

1) Notification contents for the objectCreation and objectDeletion notifications are shown in Table 4.

Table 4 – Notification contents for objectCreation and objectDeletion

Notification parameter	Qualifiers	Data type	Descriptions

correlatedNotifications	O	`CorrelatedNotifications`
		`  type: array`
		`  items:`
		`  type: string`
			List of correlated notifications

additionalText	O	`string`	Text message
additionalInfo	O	`string`	Additional information not in text format. (The information will be provided in JSON strings. Below is the same.)

Table 4 – Notification contents for objectCreation and objectDeletion

Notification parameter	Qualifiers	Data type	Descriptions

sourceIndicator	O	SourceIndicator:
		  type: string
		  format: enum
		  - resourceOperation
		  - managementOperation
		  - unknown
			Cause of event

attributeList	O	NVPairList:
		  type: object
		  properties:
		  attributeList:
		  type: array
		  items:
		  $ref:
		'#/.../NVPair'
		NVPair:
		  type: object
		  properties:
		  name:
		  type: string
		  value:
		  type: string
		  type:
		  type: string
			Attribute values

2) Notification contents for the attributeValueChange notification are shown in Table 5.

Table 5 – Notification contents for attributeValueChange

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	CorrelatedNotifications	List of correlated notifications
additionalText	O	string	Text message
additionalInfo	O	string	Additional information not in text format
sourceIndicator	O	SourceIndicator	Cause of event
attributeChanges	M	NVPairList	Changed attributes

3) Notification contents for the stateChange notification are shown in Table 6.

Table 6 – Notification contents for stateChange

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	`CorrelatedNotifications`	List of correlated notifications
additionalText	O	`string`	Text message
additionalInfo	O	`string`	Additional information not in text format
sourceIndicator	O	`SourceIndicator`	Cause of event
stateChanges	M	`NVPairList`	Changed states

4) Notification contents for the communicationAlarm, environmentalAlarm, equipmentAlarm, processingErrorAlarm and qualityOfServiceAlarm notifications are shown in Table 7.

Table 7 – Notification contents for alarms

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	`CorrelatedNotifications`	List of correlated notifications
additionalText	O	`string`	Text message
additionalInfo	O	`string`	Additional information not in text format
probableCause	M	`string`	The probable cause of the alarm

specificProblems	O	`SpecificProblems:`
		`  type: array`
		`  items:`
		`  type: string`
			Non-standardized problems

perceivedSeverity	M	`AlarmSeverity:`
		`  type: string`
		`  format: enum`
		`  - indeterminate`
		`  - critical`
		`  - major`
		`  - minor`
		`  - warning`
		`  - cleared`
			It indicates the perceived severity of the alarm. See type definition for details.

backedUpStatus	O	`boolean`	"True" if backed up
backUpObject	O	`type: string` `format: uri`	It indicates the DN of the backup object if the backUpStatus is "false".

Table 7 – Notification contents for alarms

Notification parameter	Qualifiers	Data type	Descriptions

trendIndication	O	TrendIndication:
		  type: string
		  format: enum
		  - lessSevere
		  - noChange
		  - moreSevere
			See type for details

thresholdInfo	O	ThresholdInfo:
		  type: object
		  properties:
		  attributeID:
		  type: string
		  observedValue:
		  type: number
		  format: float
		  thresholdLevel:
		  $ref:
		'#/.../ThresholdLevelInd'
		  armTime:
		  type: string
		  format: date-time
		
		ThresholdLevelInd:
		  type: object
		  properties:
		  indication:
		  $ref:
		'#/.../ThresholdIndication'
		  low:
		  type: number
		  format: float
		  high:
		  type: number
		  format: float
		
		ThresholdIndication:
		  type: string
		  format: enum
		  - up
		  - down
			See type for details

stateChangeDefinition	O	NVPairList	It indicates the state changes in this alarm.
monitoredAttributes	O	type: array items: type: string	See type for details.
proposedRepairActions	O	type: array items: type: string	It indicates the proposed actions to repair this fault.

Table 7 – Notification contents for alarms

Notification parameter	Qualifiers	Data type	Descriptions
alarmEffectOnService	O	`boolean`	True if alarm is service effecting
alarmingResumed	O	`boolean`	True if alarming was just resumed, possibly resulting in delayed reporting

suspectObjectList	O	`type: array`
		`items:`
		`  type: string`
		`  format: uri`
			Objects possibly involved in failure

5) Notification contents for the integrityViolation, operationalViolation, physicalViolation, securityViolation and timeDomainViolation notifications are shown in Table 8.

Table 8 – Notification contents for violations

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	`CorrelatedNotifications`	List of correlated notifications
additionalText	O	`string`	Text message
additionalInfo	O	`string`	Additional information not in text format
securityAlarmCause	M	`string`	The cause of the security alarm
securityAlarmSeverity	M	`AlarmSeverity`	Clears allowed? [ITU-T X.721] appears to restrict the cleared value on this alarm, but clears should be allowed.

securityAlarmDetector	M	`SecurityAlarmDetector:`
		`  type: object`
		`  properties:`
		`  mechanism:`
		`  $ref: '#/.../UID'`
		`  obj:`
		`  type: string`
		`  format: uri`
		`UID:`
		`  type: object`
		`  properties:`
		`  uri:`
		`  type: string`
		`  format: uri`
		`  value:`
		`  type: integer`
		`  format: unsignedLong`
			See type for details

Table 8 – Notification contents for violations

Notification parameter	Qualifiers	Data type	Descriptions
serviceUser	M	`string`	The user of the service which is violated
serviceProvider	M	`string`	The service provider of the service which is violated

6) Notification contents for the relationshipChange notification are shown in Table 9.

Table 9 – Notification contents for relationshipChange

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	`CorrelatedNotifications`	List of correlated notifications
additionalText	O	`string`	Text message
additionalInfo	O	`string`	Additional information not in text format
sourceIndicator	O	`SourceIndicator`	Cause of event

relationshipChanges	M	`AttributeChangeSet:`
		`  type: array`
		`  items:`
		`  $ref: '#/.../AttributeChange'`
		`AttributeChange:`
		`  type: object`
		`  properties:`
		`  attributeName:`
		`  type: string`
		`  attributeType:`
		`  type: string`
		`  oldValue:`
		`  type: string`
		`  newValue:`
		`  type: string`
			Changed relationship attributes

The detailed data type definitions for notification contents can be found in clause A.1.

9 REST-based heartbeat service

The heartbeat service is used to verify the operation of the notification forwarding mechanism in a managed system as well as that of the communications network between the managed system and managing system.

It periodically sends a small notification to a managing system interested in receiving it and that notification identifies the system that emitted the heartbeat. After configuring this service, a managing system can ensure the notification service is functioning. Since these notifications flow through the same software and networks as notifications from other resources, they periodically verify the operation of these resources.

The heartbeat service contains the following operations:

(1) Heartbeat operation:

The "heartbeat" operation is used by the managed system to send a brief heartbeat notification to the management system. The heartbeat notification information type is HeartbeatMessageType. The operation is completed by the notify operation of the notification service defined. The corresponding REST interface definition is shown in Table 10.

Table 10 – Interface definition of heartbeat operation

Name	Schema
REQUEST	"POST" {destinationURI}/"notifications" {HTTPVersion} {RequestBody}

RequestBody	HeartbeatMessageType : : = {
	Allof: [
	{ NotificationBaseType },
	{
	systemLabel: string
	period: string
	timeStamp: string
	}]
	}

RESPONSE	{HTTPVersion} {StatusCode} {ReasonPhrase} Location: {destinationURI}/"notifications"/{notificationId} {ResponseBody}
ResponseBody	HeartbeatMessageType

(2) getHeartbeatAttributes operation:

The getHeartbeatAttributes operation is used to get the value of the systemLabel and period attributes in the heartbeat notification. The REST interface definition for this operation contains the following:

a) The HTTP method: the operation of the query attribute should be mapped to the "GET" request method.

b) The request URI: In the heartbeat service, the subscribed heartbeat notification is placed in the heartbeats collection resource of the heartbeat service for management, and is identified by the subscriptionId. The query parameter attributes are optional. If no query condition is added, it means that both systemLabel and period attribute values are obtained. If you want to query a specified attribute value, the attribute name to be queried is placed in the attributes query parameter of the request URI.

Based on the above analysis, the REST interface definition of the getHeartbeatAttributes operation can be found in Table 11.

Table 11 – Interface definition of getHeartbeatAttributes operation

Name	Schema
REQUEST	"GET" heartbeatServiceURI "/ heartbeats/"subscriptionId ["?attributes="("systemLabel" | "period")] HTTPVersion
RESPONSE	{HTTPVersion} {StatusCode} {ReasonPhrase} {ResponseBody}

ResponseBody	HeartBeatAttributes : : = {
	  systemLabel: string
	  period: string
	}

(3) setHeartbeatAttributes operation:

The setHeartbeatAttributes operation is used to set the value of the systemLabel and period attributes in the heartbeat notification. The REST interface definition for this operation contains the following:

a) The HTTP method: this operation is an operation to modify the attribute, and supports modification of some attributes, so it should be mapped to the 'PATCH' request method.

b) The content of the request body: the setHeartbeatAttributes operation needs to give the attribute name and attribute value to be modified in the request body. The manager can modify the systemLabel and period attribute values at the same time, or modify only one of the attribute values.

Based on the above analysis, the REST interface definition of the setHeartbeatAttributes operation can be found in Table 12.

Table 12 – Interface definition of setHeartBeatAttribute operation

Name	Schema
REQUEST	"PATCH" { heartbeatServiceURI }/" heartbeats "/{subscriptionId} {HTTPVersion} {RequestBody}

RequestBody	HeartbeatAttributes : : = {
	  systemLabel: string
	  period: string
	}

RESPONSE	{HTTPVersion} {StatusCode} {ReasonPhrase}

The detailed JSON/YAML Schema and REST interface definitions can be found in clause A.2.

10 REST-based containment service

In the network management field, a function is needed to be able to report which objects are contained by a superior object, to verify that a superior object exists before a subordinate is created, to make sure two objects with the same name are not created, etc. The containment service is defined in this framework to support this function.

The main functions to be supported by the containment service are to enable a managing system to query a managed system with the name of an MO, and receive back the names of the objects contained by this object. In addition, a means of getting names added to and removed from the service will be defined. These are not for use by managing systems but internally by MOs, factories and other parts of a managed system. They are provided to promote the development of reusable components, possibly by third parties, and are defined on an interface separate from that used by managing systems.

The containment service provides three operations to retrieve containment information. A short description of the operation semantics in the containment service is described as follows:

(1) exists operation:

The exists operation is used to check whether a specified MO is registered in the containment service or not. The REST interface definition for this operation contains the following:

a) The HTTP method: the operation of the query attribute should be mapped to the "GET" request method.

b) The request URI: In the containment service, the "ContainmentService/exists" is used as the collection resource for this operation, and the moInstance parameter in the path will provide the DN of the specified MO instance to be checked.

Based on the above analysis, the REST interface definition of the exists operation can be found in Table 13.

Table 13 – Interface definition of exists operation

Name	Schema

REQUEST	`"GET" {ContainmentServiceURI}`
	`"/ContainmentService/exists/" {moInstance}`
	`{HTTPVersion}`

RESPONSE	`{HTTPVersion} {StatusCode} {ReasonPhrase}` `{ResponseBody}`
ResponseBody	`boolean`

(2) getContained operation:

The "getContained" operation is used to retrieve the names of all the resources contained in the specified MO. It uses DN (URI[1]) to identify the target MO and then return the list of the URIs of the MOs that are contained under this MO instance. The REST interface definition for this operation contains the following:

a) The request URI: Since the operation cannot be mapped to the basic create, read, update and delete (CRUD) operations, it is represented by a Task Resource URI, and the actionName is used to indicate the operation to be performed after the resource object URI to be operated. The actionName of the operation is represented by 'getContained'.

b) The HTTP method: this operation is a query operation, so the 'GET' request method is used.

[1] URI is defined in [IETF RFC 3986]. Based on the descriptions provided in [IETF RFC 3986]: URL refers to the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location"). Within this Recommendation, only URI is used.

c) There are two input parameters for this operation. The first one is the baseMO, which indicates the root MO instance of a subtree to be queried. The other parameter is scope, which indicates the scope of the subtree to be queried for containment: BasicObjectOnly, WholeSubTree, IndividualLevel and BaseToLevel (the semantic explanation of the above four choices can be found in clause 6.3 of [b-ITU-T Q.818]); when the choices are IndividualLevel or BaseToLevel, an extra optional parameter *level* is provided, which is a positive integer.

d) Possible response codes and corresponding response body contents: The possible status codes of the method are 200, 400, 404, 405 and 500. When the status code is 200, the contained object instance URI list information in the scope is returned in the response body. The type is defined as MOList, which is an array type of string. Each element in the list is a string type in URI format.

Based on the above analysis, the REST interface definition of the getContainment operation can be found in Table 14.

Table 14 – Interface definition of getContained operation

Name	Schema
REQUEST	`"GET" {ContainmentServiceURI} "/ContainmentService/getContained/" {moInstance}/{scope} {HTTPVersion}`
actionName	`"getContained"`
RESPONSE	`{HTTPVersion} {StatusCode} {ReasonPhrase} {ResponseBody}`
ResponseBody	`list of string`

(3) getContainedByClass operation:

The "getContainedByClass" operation is used to retrieve the names of all the resources contained in the specified MO, whose MO class should be the same value as the input parameter. The semantic of this operation is very similar to the operation "getContained", the only difference is that the returned MO name list is filtered by the input parameter *class*. The REST interface definition for this operation contains the following:

a) The request URI: The operation is represented by a Task Resource URI and the actionName is 'getContainedByClass'.

b) The HTTP method: This operation is a query operation, so the 'GET' request method is used.

c) There are three input parameters, baseMO, scope and class. The first two have the same semantic as defined in the "getContained" operation. The last parameter class is a string, which indicates the MO class for filtering.

d) Possible response codes and corresponding response body contents: The possible status codes of the method are 200, 400, 404, 405 and 500. When the status code is 200, the contained object instance URI list information in the scope with the specified class is returned in the response body.

Based on the above analysis, the REST interface definition of the getContainmentByClass operation can be found in Table 15.

Table 15 – Interface definition of getContainedByClass operation

Name	Schema
REQUEST	"GET" {ContainmentServiceURI} "/ContainmentService/getContainedByClass/" {moInstance}/{scope}/{class} {HTTPVersion}
actionName	"getContainedByClass"
RESPONSE	{HTTPVersion} {StatusCode} {ReasonPhrase} {ResponseBody}
ResponseBody	list of string

The detailed JSON/YAML Schema and REST interface definitions can be found in Annex A.3.

[IETF RFC 7231] and [IETF RFC 6585] provide dozens of definitions for status code, which will likely be referenced in implementations of this Recommendation. The list of status codes can be found in Table A.2 of [ITU-T X.785].

11 Compliance and conformance

This clause defines the criteria that must be met by other standards documents claiming compliance to this framework and the functions that must be implemented by systems claiming conformance to this Recommendation.

11.1 System conformance points

This clause summarizes the individual functions described earlier in this Recommendation. These conformance points are then combined in profiles that must be supported by systems claiming conformance to this Recommendation.

1) An implementation claiming conformance to the notification service must:
   - support the notification service requirements and interface described in clause 8 and defined in the YAML/JSON schema in clause A.1;
   - support the notification format described in clause 8.3 and defined in the YAML/JSON schema in clause A.1.
2) An implementation claiming conformance to the heartbeat service must:
   - support the heartbeat service requirements and interface described in clause 9 and defined in the YAML/JSON schema in clause A.2.
3) An implementation claiming conformance to the containment service must:
   - support the containment service requirements and interface described in clause 10 and defined in the YAML/JSON schema in clause A.3.

11.2 Basic conformance profile

A system claiming conformance to this Recommendation's basic profile shall support:

1) the version of JSON/YAML schema and HTTP protocol as specified in clause 7.2;
2) the notification service (see conformance point 1);
3) the heartbeat service (see conformance point 2);
4) the containment service (see conformance point 3).

Annex A

YAML/JSON scheme definition of framework support services

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

A.1 REST-based notification service interface

A.1.1 JSON-schema definitions for REST-based notification service interface

The following provides the HTTP requests and responses for REST-based notification service interface.

1) subscribeNotification

REQUEST:

```
"POST" {notificationServiceURI}/"subscriptions" {HTTPVersion}
```

response body schema:

```
$ref: "notificationApiDataSchema.json#/definitions/SubscribeNotificationRequest"
```

RESPONSE:

```
{HTTPVersion} {statusCode} {reasonPhrase}
Location: { notificationServiceURI }/"subscriptions"/{subscriptionId}
{response body}
```

statusCode	reasonPhrase	response body schema
201	Created	`null or $ref:"notificationApiDataSchema.json#/definitions/NotificationInfo"`
400	Bad Request	`$ref:"commonApiDataSchema.json#/definitions/ErrorInformation"`
404	Not Found	`null`
409	Conflict	`null`
500	Internal Server Error	`null`

2) unsubscribeNotification

REQUEST:

```
"DELETE" {notificationServiceURI}/"subscriptions"/{subscriptionId} {HTTPVersion}
```

RESPONSE:

```
{HTTPVersion} {statusCode} {reasonPhrase}
{response body}
```

statusCode	reasonPhrase	response body schema
200	OK	`null`
404	Not Found	`null`
405	Method Not Allowed	`null`
500	Internal Server Error	`null`

3) suspendNotification

REQUEST:

```
"POST" { notificationServiceURI }/"subscriptions"/{subscriptionId}/
"suspendSubscription" {HTTPVersion}
```

RESPONSE:

```
{HTTPVersion} {statusCode} {reasonPhrase}

{response body}
```

statusCode	reasonPhrase	response body schema
200	OK	`null`
404	Not Found	`null`
405	Method Not Allowed	`null`
409	Conflict	`null (Already suspended)`
500	Internal Server Error	`null`

4) resumeSubscriptions

REQUEST:

```
"POST" { notificationServiceURI }/"subscriptions"/{subscriptionId}/
"resumeSubscriptions" {HTTPVersion}
```

RESPONSE:

```
{HTTPVersion} {statusCode} {reasonPhrase}

{response body}
```

statusCode	reasonPhrase	response body schema
200	OK	`null`
404	Not Found	`null`
405	Method Not Allowed	`null`
409	Conflict	`null (Already resumed)`
500	Internal Server Error	`null`

5) getNotificationTypes

REQUEST:

```
"GET" {notificationServiceURI}/"subscriptions"/{subscriptionId}/"getTypes"

{HTTPVersion}
```

RESPONSE:

```
{HTTPVersion} {statusCode} {reasonPhrase}

{response body}
```

statusCode	reasonPhrase	response body schema
200	OK	`$ref: "notificationApiDataSchema.json# /definitions/GetNotificationTypesResponse"`
404	Not Found	`null`
500	Internal Server Error	`null`

6) querySubscription

REQUEST:

```
"GET" {notificationServiceURI}/"subscriptions"/{subscriptionId} {HTTPVersion}
```

RESPONSE:

```
{HTTPVersion} {statusCode} {reasonPhrase}
{response body}
```

statusCode	reasonPhrase	response body schema
200	OK	`$ref: "notificationApiDataSchema.json#/definitions/QuerySubscriptionResponse"`
404	Not Found	`null`
500	Internal Server Error	`null`

7) modifySubscription

REQUEST:

```
"PATCH" {notificationServiceURI}/"subscriptions"/{subscriptionId}
{HTTPVersion}
{request body}
```

Request body schema:

```
$ref:"notificationApiDataSchema.json#/definitions/ModifySubscriptionRequest"
```

RESPONSE:

```
{HTTPVersion} {statusCode} {reasonPhrase}
{response body}
```

statusCode	reasonPhrase	response body schema
200	OK	`null or` `$ref: "notificationApiDataSchema.json#/definitions/NotificationInfo"`
400	Bad Request	`$ref: "commonApiDataSchema.json#/definitions/ErrorInformation"`
404	Not Found	`null`
405	Method Not Allowed	`null`
500	Internal Server Error	`null`

8) listAllSubscriptionIds

REQUEST:

```
"POST" {notificationServiceURI}/"subscriptions"/listAllSubscriptionIds
{HTTPVersion}
{request body}
```

Request body schema:

```
$ref:
"notificationApiDataSchema.json#/definitions/ListAllSubscriptionsIdsRequest"
```

RESPONSE:

```
{HTTPVersion} {statusCode} {reasonPhrase}
{response body}
```

statusCode	reasonPhrase	response body schema
200	OK	`$ref:"notificationApiDataSchema.json#/definitions/ListAllSubscriptionsIdsResponse"`
400	Bad Request	`$ref:"commonApiDataSchema.json#/definitions/ErrorInformation"`
404	Not Found	`null`
500	Internal Server Error	`null`

9) notify

REQUEST:

```
"POST" {destinationURI} {HTTPVersion}
{request body}
```

Request body schema:

```
$ref: "notificationApiDataSchema.json#/definitions/NotifyRequest"
```

RESPONSE:

```
{HTTPVersion} {statusCode} {reasonPhrase}
{response body}
response body schema:
```

statusCode	reasonPhrase	response body schema
200	OK	`null`
400	Bad Request	`$ref:"commonApiDataSchema.json#/definition/ErrorInformation"`
404	Not Found	`null`
405	Method Not Allowed	`null`
500	Internal Server Error	`null`

A.1.2 YAML-schema definitions for REST-based notification service

Figure A.1 provides the YAML-schema definitions for REST-based notification service interface.

NOTE – The "notify" operation in clause A.1.1 (9) is mapped to a callback in the YAML-schema interface definitions for the `subscribeNotification` operation of path /Subscription.

```
openapi: 3.0.0
info:
  title: ITU-T_NotificationService
  description: 'This is the formal definition of the ITU-T Notification
Service.'
  termsOfService: https://www.itu.int/en/ITU-T/about/Pages/default.aspx
  contact:
    email: zlwang@bupt.edu.cn
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
  version: 1.0.0
externalDocs:
  description: See Recommendation ITU-T Q.819
  url: https://www.itu.int/itu-t/recommendations/index.aspx?ser=X
servers:
- url: http://www.itu.int/sg2/rest/NotificationService/v1.0.0
```

```
- url: https://www.itu.int/sg2/rest/NotificationService/v1.0.0
tags:
- name: NotificationService
  description: All APIs related to the ITU-T Notification Service
  externalDocs:
    description: Find out more
    url: https://www.itu.int/itu-t/recommendations/index.aspx?ser=X

paths:
  /Subscription:
    post:
      tags:
      - NotificationService
      summary: "subscribe notification"
      description: "create an subscription in the notification services with the
specified parameters list"
      operationId: "subscribeNotification"
      requestBody:
        description:  "The  input  parameters  of  the  subscribeNotification
operation"
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SubscribeNotificationRequest'
        required: true
      responses:
        201:
          description:  "Subscription  is  successfully  created,  and  the  new
Subscription ID value will be returned."
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/subscriptionId'
        400:
          description:  "Parameter  Error  occurred  in  the  subscribeNotification
operation"
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorInfo'
        404:
          description: "The URI does not exist"
          content: {}
        409:
          description: "Subscription Conflict"
          content: {}
        500:
          description: "Internal Server Error"
          content: {}
      callbacks:
        notification:
          '{$request.query.requestBody.destination}':

#'{$request.query.requestBody.destination}?event={$request.query.requestBody.eve
ntTypes}':
            post:
              requestBody:
                content:
                  application/json:
                    schema:
```

```
                    $ref: '#/components/schemas/NotificationInfo'
                  #example:
                    #messageasdfasdf: Some event
              responses:
                '200':
                  description: OK

    get:
      tags:
      - NotificationService
      summary: "Get Subscription Information"
      description: "get the list of Identifiers of all the existing Notification
Subscriptions"
      operationId: "listAllSubscriptionIds"
      responses:
        200:
          description:  "The  attribute  value  list  of  the  specified  MO  is
retrieved successfully"
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: "#/components/schemas/subscriptionId"
        404:
          description: "Specified Subscription Not found"
          content: {}
        500:
          description: "Internal Server Error"
          content: {}

  /Subscription={subscriptionId}:
    parameters:
      - name: subscriptionId
        in: path
        description: "The unique Identifier of the notification Subscription"
        required: true
        schema:
          type: string

    get:
      tags:
      - NotificationService
      summary: "Query Subscription Information"
      description: "query the attributes information of a specific Notification
Subscription"
      operationId: querySubscription"
      responses:
        200:
          description:  "The  attribute  value  list  of  the  specified  MO  is
retrieved successfully"
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/SubscriptionInfo"
        404:
          description: "Specified Subscription Not found"
          content: {}
        500:
          description: "Internal Server Error"
```

```
          content: {}

    delete:
      tags:
      - NotificationService
      summary: "Unsubscribe Notification"
      description: "delete a specific Notification Subscription"
      operationId: "unsubscribeNotification"
      responses:
        200:
          description: "The specified Subscription is deleted successfully"
          content: {}
        404:
          description: "Specified Subscription does not exist"
          content: {}
        405:
          description: "Method Not Allowed, the specified Subscription cannot be
deleted"
          content: {}
        500:
          description: "Internal Server Error"
          content: {}

    patch:
      tags:
      - NotificationService
      summary: "Modify Subscription"
      description: "modify a specific Notification Subscription"
      operationId: "modifySubcription"
      requestBody:
        description: "The input parameters of the modifySubscription operation"
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ModifySubscriptionRequest'
        required: true
      responses:
        200:
          description: "The specified Subscription is modified successfully, and
the modified Subscription information is returned"
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/SubscriptionInfo'
        400:
          description:  "Parameter  Error  occurred  in  the  modifySubscription
operation"
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorInfo'
        404:
          description: "Specified Subscription does not exist"
          content: {}
        405:
          description: "Method Not Allowed, the specified Subscription cannot be
modified"
          content: {}
        500:
          description: "Internal Server Error"
```

```
        content: {}

  /Subscription={subscriptionId}/suspend:
    parameters:
      - name: subscriptionId
        in: path
        description: "The unique Identifier of the notification Subscription"
        required: true
        schema:
          type: string

    post:
      tags:
      - NotificationService
      summary: "Suspend Subscription "
      description: "Suspend the specific Notification Subscription"
      operationId: "suspendSubscription"
      responses:
        200:
          description: "The specified Subscription is suspended successfully"
          content: {}
        404:
          description: "Specified Subscription Not found"
          content: {}
        405:
          description: "Method Not Allowed"
          content: {}
        409:
          description: "Subscription already Suspended"
          content: {}
        500:
          description: "Internal Server Error"
          content: {}

  /Subscription={subscriptionId}/resume:
    parameters:
      - name: subscriptionId
        in: path
        description: "The unique Identifier of the notification Subscription"
        required: true
        schema:
          type: string

    post:
      tags:
      - NotificationService
      summary: "Resume Subscription "
      description: "Resume the specific suspended Notification Subscription"
      operationId: "resumeSubscription"
      responses:
        200:
          description: "The specified Subscription is resumed successfully"
          content: {}
        400:
          description: "Subscription already Resumed"
          content: {}
        404:
          description: "Specified Subscription Not found"
          content: {}
        405:
```

```
          description: "Method Not Allowed"
          content: {}
        500:
          description: "Internal Server Error"
          content: {}

  /NotificationTypes:
    get:
      tags:
      - NotificationService
      summary: "Get Notification Types"
      description: "get the supported Notification Types."
      operationId: "getNotificationTypes"
      responses:
        200:
          description:  "The  attribute  value  list  of  the  specified  MO  is
retrieved successfully"
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: "#/components/schemas/NotificationType"

        404:
          description: "Specified Subscription Not found"
          content: {}
        500:
          description: "Internal Server Error"
          content: {}

components:
  schemas:

    NVPair:
      type: object
      required:
      - name
      - value
      properties:
        name:
          type: string
        value:
          type: string
        type:
          type: string

    NVPairList:
      type: object
      properties:
        attributeList:
          type: array
          items:
            $ref: '#/components/schemas/NVPair'

    SubscribeNotificationRequest:
      type: object
      properties:
        managerId:
          type: string
```

```
        notificationTypeList:
        #empty list means subscribing all notification types.
          type: array
          items:
            $ref: '#/components/schemas/NotificationType'
        filteringCriteria:
        #optional
          type: string
        destination:
        #URI of the notification subscriber
          type: string
          format: uri
      required:
        - managerId
        - destination

    NotificationType:
      type: string
      enum:
        - objectCreation
        - objectDeletion
        - attributeValueChange
        - stateChange
        - communicationAlarm
        - environmentalAlarm
        - equipmentAlarm
        - processingErrorAlarm
        - qualityOfServiceAlarm
        - integrityViolation
        - operationalViolation
        - physicalViolation
        - securityViolation
        - timeDomainViolation
        - relationshipChange
        - heartbeat

    subscriptionId:
      type: string

    ErrorInfo:
      type: object
      properties:
        code:
          type: string
          enum:
            - invalidObjectInstance
            - noSuchObjectClass
            - noSuchAttribute
            - invalidAttributeValue
            - missingAttributeValue
            - notFound
            - cannotBeDeleted
        message:
          type: string
      required:
        - code

    SubscriptionInfo:
      type: object
      properties:
```

```
        subscriptionId:
          type: string
        managerId:
        #the identifier of manager of this notification subscription
          type: string
        notificationTypeList:
        #empty list means subscribing all notification types.
          type: array
          items:
            $ref: '#/components/schemas/NotificationType'
        filteringCriteria:
        #optional
          type: string
        destination:
        #URI of the notification subscriber
          type: string
          format: uri
        subscriptionStatus:
        #Whether the subscription is suspended or resumed
          type: string
          enum:
            - suspended
            - resumed
      required:
        - subscriptionId
        - managerId
        - destination
        - subscriptionState

    ModifySubscriptionRequest:
      type: object
      properties:
        notificationTypeList:
        #empty list means subscribing all notification types.
          type: array
          items:
            $ref: '#/components/schemas/NotificationType'
        filteringCriteria:
        #optional
          type: string
        destination:
        #URI of the notification subscriber
          type: string
          format: uri

    NotificationInfo:
      type: object
      properties:
        notificationHeader:
          $ref: '#/components/schemas/NotificationHeader'
        notificationBody:
          $ref: '#/components/schemas/NotificationBody'
      required:
      - notificationHeader
      - notificationBody

    NotificationHeader:
      type: object
      properties:
        objectClass:
```

```
          type: string
        objectInstance:
          type: string
          format: uri
        notificationId:
          type: string
        eventTime:
          type: string
          # to be confirmed.
          format: date-time
        systemDN:
          type: string
        notificationType:
          $ref: '#/components/schemas/NotificationType'
      required:
        - objectClass
        - objectInstance
        - notificationId
        - eventTime
        - systemDN
        - notificationType

    NotificationBody:
      type: object
      #One of the following will be used in a concreate notification
      properties:
        objectCreationBody:
          $ref: '#/components/schemas/ObjectCreationDeletionBody'
        objectDeletionBody:
          $ref: '#/components/schemas/ObjectCreationDeletionBody'
        atributeValueChangeBody:
          $ref: '#/components/schemas/AttributeValueChangeBody'
        stateChangeBody:
          $ref: '#/components/schemas/StateChangeBody'
        alarmNotificationBody:
          #for    communicationsAlarm,    environmentalAlarm,    equipmentAlarm,
processingErrorAlarm, qualityOfServiceAlarm
          $ref: '#/components/schemas/AlarmNotificationBody'
        violationsNotificationBody:
          $ref: '#/components/schemas/ViolationsNotificationBody'
        relationshipChangeNotificationBody:
          $ref: '#/components/schemas/RelationshipChangeNotificationBody'
        heartbeatNotificationBody:
          $ref: '#/components/schemas/HeartbeatNotificationBody'

    ObjectCreationDeletionBody:
      type: object
      properties:
        commonAttributes:
          $ref: '#/components/schemas/CMNotificationCommonAttrs'
        attributeList:
          $ref: '#/components/schemas/NVPairList'

    AttributeValueChangeBody:
      type: object
      properties:
        commonAttributes:
          $ref: '#/components/schemas/CMNotificationCommonAttrs'
        attributeChanges:
          $ref: '#/components/schemas/NVPairList'
```

```
    required:
      - attributeChanges

  StateChangeBody:
    type: object
    properties:
      commonAttributes:
        $ref: '#/components/schemas/CMNotificationCommonAttrs'
      stateChanges:
        $ref: '#/components/schemas/NVPairList'
    required:
      - stateChanges

  CMNotificationCommonAttrs:
    type: object
    properties:
      correlatedNotifications:
        type: array
        items:
          type: string
      additionalText:
        type: string
      additionalInfo:
        type: string
      sourceIndicator:
        $ref: '#/components/schemas/SourceIndicator'

  SourceIndicator:
    type: string
    format: enum
      - resourceOperation
      - managementOperation
      - unknown

  AlarmNotificationBody:
    type: object
    properties:
      correlatedNotifications:
        type: array
        items:
          type: string
      additionalText:
        type: string
      additionalInfo:
        type: string
      probableCause:
        type: string
      specificProblems:
        type: array
        items:
          type: string
      perceivedSeverity:
         $ref: '#/components/schemas/AlarmSeverity'
      backedUpStatus:
        type: boolean
      backUpObject:
        type: string
        format: uri
      trendIndication:
        $ref: '#/components/schemas/TrendIndication'
```

```
        thresholdInfo:
          $ref: '#/components/schemas/ThresholdInfo'
        stateChangeDefinition:
          $ref: '#/components/schemas/StateChangeDefinition'
        monitoredAttributes:
          type: array
          items:
            type: string
        proposedRepairActions:
          type: array
          items:
            type: string
        alarmEffectOnService:
          type: boolean
        alarmingResumed:
          type: boolean
        suspectObjectList:
          type: array
          items:
            type: string
            format: uri
      required:
        - probableCause
        - perceivedSeverity


    AlarmSeverity:
      type: string
      format: enum
        - cleared
        - warning
        - indetermined
        - minor
        - major
        - critical

    TrendIndication:
      type: string
      format: enum
        - lessSevere
        - noChange
        - moreSevere

    ThresholdInfo:
      type: object
      properties:
        attributeID:
          type: string
        observedValue:
          type: number
          format: float
        thresholdLevel:
          $ref: '#/components/schemas/ThresholdLevelInd'
        armTime:
          type: string
          format: date-time

    ThresholdLevelInd:
      type: object
      properties:
```

```
        indication:
          $ref: '#/components/schemas/ThresholdIndication'
        low:
          type: number
          format: float
        high:
          type: number
          format: float

    ThresholdIndication:
      type: string
      format: enum
        - up
        - down

    StateChangeDefinition:
      $ref: '#/components/schemas/NVPairList'

    ViolationsNotificationBody:
      type: object
      properties:
        correlatedNotifications:
          type: array
          items:
            type: string
        additionalText:
          type: string
        additionalInfo:
          type: string
        securityAlarmCause:
          type: string
        securityAlarmSeverity:
           $ref: '#/components/schemas/AlarmSeverity'
        securityAlarmDetector:
          $ref: '#/components/schemas/SecurityAlarmDetector'
        serviceUser:
          type: string
        serviceProvider:
          type: string
      required:
        - securityAlarmCause
        - securityAlarmSeverity
        - securityAlarmDetector
        - serviceUser
        - serviceProvider


    SecurityAlarmDetector:
      type: object
      properties:
        mechanism:
          $ref: '#/components/schemas/UID'
        obj:
          type: string
          format: uri

    UID:
      type: object
      properties:
        uri:
```

```
          type: string
          format: uri
        value:
          type: integer
          format: unsignedLong

    RelationshipChangeNotificationBody:
      type: object
      properties:
        commonAttributes:
          $ref: '#/components/schemas/CMNotificationCommonAttrs'
        relationshipChanges:
          $ref: '#/components/schemas/AttributeChangeSet'
      required:
        - relationshipChanges

    AttributeChangeSet:
      type: array
      items:
        $ref: '#/components/schemas/AttributeChange'

    AttributeChange:
      type: object
      properties:
        attribugteName:
          type: string
        attributeType:
          type: string
        oldValue:
          type: string
        newValue:
          type: string

    HeartbeatNotificationBody:
      type: object
      properties:
        systemLabel:
          type: string
          format: uri
        period:
          type: integer
          format: unitsInSeconds
        timeStamp:
          type: string
          format: date-time
```

Figure A.1 – YAML-schema definitions for REST-based notification service

A.2 REST-based heartbeat service interface

A.2.1 JSON-schema definitions for REST-based heartbeat service

The following provides the HTTP requests and responses for REST-based heartbeat service interface.

(1) Table A.1 shows the heartbeat operation.

Table A.1 – Interface definition of heartbeat operation

Name	Schema
REQUEST	"POST" { destinationURI }/" notifications " {HTTPVersion} {RequestBody}

RequestBody	HeartbeatMessageType : : = {
	Allof: [
	{ NotificationBaseType },
	{
	systemLabel: string
	period: string
	timeStamp: string
	}]
	}

RESPONSE	{HTTPVersion} {StatusCode} {ReasonPhrase} Location: { destinationURI }/" notifications "/{ notificationId } {ResponseBody}
ResponseBody	HeartbeatMessageType

(2) Table A.2 shows the getHeartbeatAttributes operation.

Table A.2 – Interface definition of getHeartbeatAttributes operation

Name	Schema
REQUEST	"GET" heartbeatServiceURI "/ heartbeats/"subscriptionId ["?attributes="("systemLabel" | "period")] HTTPVersion
RESPONSE	{HTTPVersion} {StatusCode} {ReasonPhrase} {ResponseBody}

ResponseBody	HeartBeatAttributes : : = {
	systemLabel: string
	period: string'
	}

(3) Table A.3 shows the setHeartbeatAttributes operation.

Table A.3 – Interface definition of setHeartbeatAttribute operation

Name	Schema
REQUEST	"PATCH" { heartbeatServiceURI }/" heartbeats "/{subscriptionId} {HTTPVersion} {RequestBody}

RequestBody	HeartbeatAttributes : : = {
	systemLabel: string
	period: string
	}

RESPONSE	{HTTPVersion} {StatusCode} {ReasonPhrase}

A.2.2 YAML-schema definitions for REST-based heartbeat service

Table A.4 provides the YAML-schema definitions for the REST-based heartbeat service interface.

Table A.4 – YAML-schema definitions for REST-based heartbeat service

```
openapi: 3.0.0
info:
  title: ITU-T_HeartbeatService
  description: 'This is the formal definition of the ITU-T Heartbeat Service.'
  termsOfService: https://www.itu.int/en/ITU-T/about/Pages/default.aspx
  contact:
    email: zlwang@bupt.edu.cn
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
  version: 1.0.0
externalDocs:
  description: See Recommendation ITU-T Q.819
  url: https://www.itu.int/itu-t/recommendations/index.aspx?ser=X
servers:
- url: https://www.itu.int/sg2/rest/HeartbeatService/v1.0.0
- url:  http://www.itu.int/sg2/rest/HeartbeatService/v1.0.0
tags:
- name: HeartbeatService
  description: All APIs related to the ITU-T Heartbeat Service
  externalDocs:
    description: Find out more
    url: https://www.itu.int/itu-t/recommendations/index.aspx?ser=X

paths:
  /HeartbeatService/{subscriptionId}:
    parameters:
      - name: subscriptionId
        in: path
        description: "The unique Identifier of the notification Subscription"
        required: true
        schema:
          type: string

    get:
      tags:
      - HeartbeatService
      summary: "Get Heartbeat Service Information"
      description:  "get  the  list  of  Identifiers  of  all  the  existing
Notification Subscriptions"
      operationId: "getHeartbeatAttributes"
      responses:
        200:
          description:  "The  attribute  value  list  of  the  specified  MO  is
retrieved successfully"
          content:
            application/json:
              schema:
                  $ref: "#/components/schemas/HeartbeatInfo"
        404:
          description: "Specified Subscription Not found"
          content: {}
        500:
          description: "Internal Server Error"
          content: {}

    patch:
      tags:
```

```
      - HeartbeatService
      summary: "Modify Heartbeat service attributes"
      description: "modify the attribute of a heartbeat service attributes."
      operationId: "modifyHeartbeatService"
      requestBody:
        description:  "The  input  parameters  of  the  modifySubscription
operation"
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ModifyHeartbeatServiceRequest'
        required: true
      responses:
        200:
          description: "The specified Subscription is modified successfully,
and the modified Subscription information is returned"
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/SubscriptionInfo'
        400:
          description: "Parameter Error occurred in the modifyHeartbeatService
operation"
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorInfo'
        404:
          description: "Specified Subscription does not exist"
          content: {}
        405:
          description: "Method Not Allowed, the specified Subscription cannot
be modified"
          content: {}
        500:
          description: "Internal Server Error"
          content: {}

components:
  schemas:
    NotificationType:
      type: string
      enum:
        - objectCreation
        - objectDeletion
        - attributeValueChange
        - stateChange
        - communicationAlarm
        - environmentalAlarm
        - equipmentAlarm
        - processingErrorAlarm
        - qualityOfServiceAlarm
        - integrityViolation
        - operationalViolation
        - physicalViolation
        - securityViolation
        - timeDomainViolation
        - relationshipChange
        - heartbeat
```

```
    ErrorInfo:
      type: object
      properties:
        code:
          type: string
          enum:
            - invalidObjectInstance
            - noSuchObjectClass
            - noSuchAttribute
            - invalidAttributeValue
            - missingAttributeValue
            - notFound
            - cannotBeDeleted
        message:
          type: string
      required:
        - code

    SubscriptionInfo:
      type: object
      properties:
        subscriptionId:
          type: string
        managerId:
        #the identifier of manager of this notification subscription
          type: string
        notificationTypeList:
        #empty list means subscribing all notification types.
          type: array
          items:
            $ref: '#/components/schemas/NotificationType'
        filteringCriteria:
        #optional
          type: string
        destination:
        #URI of the notification subscriber
          type: string
          format: uri
        subscriptionStatus:
        #Whether the subscription is suspended or resumed
          type: string
          enum:
            - suspended
            - resumed
      required:
        - subcriptionId
        - managerId
        - destination
        - subscriptionState

    HeartbeatInfo:
      type: object
      properties:
        systemLabel:
          type: string
          format: uri
        period:
          type: integer
          format: unitsInSeconds
      required:
```

```
      - systemLabel
      - period

  ModifyHeartbeatServiceRequest:
    type: object
    properties:
      systemLabel:
        type: string
        format: uri
      period:
        type: integer
        format: unitsInSeconds

  HeartbeatNotificationBody:
    type: object
    properties:
      systemLabel:
        type: string
        format: uri
      period:
        type: integer
        format: unitsInSeconds
      timeStamp:
        type: string
        format: date-time
```

A.3 REST-based containment service interface

A.3.1 JSON-schema definitions for REST-based containment service

The JSON schema definitions for the REST-base containment service interface include the following information, as show in Tables A.5, A.6 and A.7:

(1) Table A.5 shows the exists operation.

Table A.5 – Interface definition of exists operation

Name	Schema
REQUEST	"GET" {ContainmentServiceURI} "/ContainmentService/exists/" {moInstance} {HTTPVersion}
RESPONSE	{HTTPVersion} {StatusCode} {ReasonPhrase} {ResponseBody}
ResponseBody	boolean

(2) Table A.6 shows the getContained operation.

Table A.6 – Interface definition of getContained operation

Name	Schema
REQUEST	"GET" {ContainmentServiceURI} "/ContainmentService/getContained/" {moInstance}/{scope} {HTTPVersion}
actionName	"getContained"
RESPONSE	{HTTPVersion} {StatusCode} {ReasonPhrase} {ResponseBody}
ResponseBody	list of string

(3) Table A.7 shows the getContainedByClass operation.

Table A.7 – Interface definition of getContainedByClass operation

Name	Schema
REQUEST	"GET" {ContainmentServiceURI} "/ContainmentService/getContainedByClass/" {moInstance}/{scope}/{class} {HTTPVersion}
actionName	"getContainedByClass"
RESPONSE	{HTTPVersion} {StatusCode} {ReasonPhrase} {ResponseBody}
ResponseBody	list of string

A.3.2 YAML-schema definitions for REST-based containment service

Table A.8 provides the YAML-schema definitions for REST-based containment service interface.

Table A.8 – YAML-schema definitions for REST-based containment service

```
openapi: 3.0.0
info:
  title: ITU-T_ContainmentService
  description: 'This is the formal definition of the ITU-T Containment
Service.'
  termsOfService: https://www.itu.int/en/ITU-T/about/Pages/default.aspx
  contact:
    email: zlwang@bupt.edu.cn
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
  version: 1.0.0
externalDocs:
  description: See Recommendation ITU-T Q.819
  url: https://www.itu.int/itu-t/recommendations/index.aspx?ser=X
servers:
- url: http://www.itu.int/sg2/rest/ContainmentService/v1.0.0
- url: https://www.itu.int/sg2/rest/ContainmentService/v1.0.0
tags:
- name: ContainmentService
  description: All APIs related to the ITU-T Containment Service
  externalDocs:
    description: Find out more
    url: https://www.itu.int/itu-t/recommendations/index.aspx?ser=X
```

```
paths:
  /ContainmentService/exists/{moInstance}:
    parameters:
      - name: moInstance
        in: path
        description: "The DN(URI) of the specified MO"
        required: true
        schema:
          type: string
          format: uri

    get:
      tags:
      - ContainmentService
      summary: "check whether the specified MO instance is registered in the
containment service or not."
      description: "returns true if the specified MO instance exists in the
containment service, otherwise it returns false"
      operationId: "exists"
      responses:
        200:
          description:  "The  specified  MO  is  found  in  the  containment
services."
          content: {}
        404:
          description:  "The  specified  MO  is  not  found  in  the containment
services."
          content: {}
        500:
          description: "Internal Server Error"
          content: {}

  /ContainmentService/getContained/{baseMO}/{scope}:
    parameters:
      - name: baseMO
        in: path
        description: "The DN of the specified base MO which is the root of a
subtree"
        required: true
        schema:
          type: string
          format: uri
      - name: scope
        in: path
        description: "The scope of the containment information to be returned
associated with the root MO"
        required: true
        schema:
          $ref: '#/components/schemas/ScopeType'

    get:
      tags:
      - ContainmentService
      summary: "returns the MO instance list which are contained by the
specified base MO instance."
      description: "returns the list of MO instance that are contained by the
base  MO  instance.  the  scope  parameter  can  be  of  4  possible  values:
BaseObjectOnly, WholeSubTree, IndividualLevel, IndividualLevel"
      operationId: "getContained"
```

```
      responses:
        200:
          description: "The operation is executed successfully, and the list
of contained MO instances are returned"
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/MOList'
        404:
          description: "The specified MO is not found in the containment
services."
          content: {}
        500:
          description: "Internal Server Error"
          content: {}

  /ContainmentService/getContainedByClass/{baseMO}/{scope}/{class}:
    parameters:
      - name: baseMO
        in: path
        description: "The DN of the specified base MO which is the root of a
subtree"
        required: true
        schema:
          type: string
          format: uri
      - name: scope
        in: path
        description: "The scope of the containment information to be returned
associated with the root MO"
        required: true
        schema:
          $ref: '#/components/schemas/ScopeType'
      - name: class
        in: path
        description: "The class of the contained information to be returned
associated with the root MO"
        required: true
        schema:
          type: string

    get:
      tags:
      - ContainmentService
      summary: "returns the MO instance list which are contained by the
specified base MO instance."
      description: "returns the list of MO instance that are contained by the
base MO instance. the scope parameter can be of 4 possible values:
BaseObjectOnly, WholeSubTree, IndividualLevel, IndividualLevel, and the
returned MO list is filtered by the class specified in the input parameter"
      operationId: "getContainedByClass"
      responses:
        200:
          description: "The operation is executed successfully, and the list
of contained MO instances are returned"
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/MOList'
        404:
```

```
          description: "The specified MO is not found in the containment
services."
          content: {}
        500:
          description: "Internal Server Error"
          content: {}

components:
  schemas:

    ScopeEnumType:
      type: string
      format: enum
        - BasicObjectOnly
        - WholeSubtree
        - IndividualLevel
        - BaseToLevel

    ScopeType:
      type: object
      properties:
        scopeInd:
          $ref: '#/components/schemas/ScopeEnumType'
        level:
          type: integer

    MOList:
      type: array
      items:
        type: string
        format: uri
```

Bibliography

[b-ITU-T Q.818] Recommendation ITU-T Q.818 (2012), *Web service-based management services.*

[b-OAI-OAS3] OpenAPI Initiative (2017), *OpenAPI Specification Version 3.0.0*, http://spec.openapis.org/oas/v3.0.0.

SERIES OF ITU-T RECOMMENDATIONS

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

Printed in Switzerland
Geneva, 2022