INTERNATIONAL TELECOMMUNICATION UNION
TELECOMMUNICATION STANDARDIZATION SECTOR
STUDY PERIOD 2017-2020

SG2-C103
STUDY GROUP 2
Original: English

Question(s): Q7/2 Geneva, 4-13 July 2018

CONTRIBUTION

Source: Beijing University of Posts and Telecommunications

Title: The proposal to create a new work item on “Guidelines for defining REST-based managed objects and management interfaces”

Purpose: [Choose a purpose from the dropdown list]

Contact: WANG Zhili
BUPT
China
Tel: +86 10 61198090 ext. 8726
Fax: +86 10 62283412
Email: zlwang@bupt.edu.cn

Keywords: Distributed processing; (REST), managed objects, network management interfaces, JSON, JSON Schema.

Abstract: This document defines a set of guidelines for managed object modelling and a management interface for REST-based network management. It composes a framework for REST-based network management interfaces along with draft Q.rest. It specifies how REST-based management interfaces should be defined. It covers generic accessing methods of XML-based managed objects, information modelling in REST/HTTP and JSON schema. Some HTTP requests/responses and JSON schema are provided for defining some basic data types: generic managed object (MO) and generic MO accessing methods. This document and draft Q.rest together compose a framework for REST-based network management interfaces with a wide range of applications.

1. Background

REST technology is now broadly used in IT Industry. In some organization and fora, they have started the research work on how to apply REST technology in network management field as an alternative interface technology.

When using a REST technology in network management interfaces, some guidelines on how to use it to defined interface and managed entities, as well as some supporting services should be provided. These guidelines, supporting services and some common definitions of generic managed objects together can be called the framework for REST-based paradigm.

The purpose of this document is trying to provide some related information in order to establish the framework for defining REST based network management interface and supporting services, so that in the future, specific REST-based interface definition can follow those guidelines, and reuse some common services.

2. Short review of REST and HTTP

2.1 REST design principles

REST stands for **RE**presentational **S**tate **T**ransfer. It is an architectural style defined by the following principles:

(1) Client-server architecture

REST follows a client-server architecture. Client and server are linked by the uniform interface. The server is concerned with data storage. The client manipulates this data with create, read, update and delete (CRUD) operations. This architecture allows the client and server to evolve independently.

(2) Stateless servers

REST servers are stateless, meaning that no client context is stored on the server. It is the client holding the session state. Each request from a client contains all the information required to service the request.

(3) Cacheability

REST is cacheable. The client and any intermediary can cache responses, helping to improve system scalability and performance.

(4) Layered System

REST is a **layered system**. A client cannot know if it is interacting with the end server or an intermediate server on the way to the end server. Each component has only knowledge about the component it is interacting with. All components are independent and easily replaceable or extendable. This improves system scalability and enables load-balancing.

(5) Code on demand

Code on demand is an optional REST feature. It allows servers to transfer executable code to the client, thereby extending the functionality of the client.

(6)Uniform interface

The uniform interface is the most important aspect of REST. Client and server communicate via the uniform interface. It is characterized by the following

- *Resource identification*: The key concept is to abstract information into resources. These resources have a unique resource identification. Requests are directed towards resources.
- *Resource representation*: Each resource has one or multiple representations. Representations can be in e.g. XML, JSON or HTML. Resource representations are exchanged over the wire together with any representation metadata. The metadata provides information about the representation, such as its media type, the date of last modification, or even a checksum.
- *Self-descriptive messages*: Messages must be self-descriptive. All the information required to process the message is included in the message.
- *Hypermedia as the engine of application state* (HATEOAS): This refers to the capability of the server to send hyperlinks to the client allowing the client to traverse and dynamically discover resources without referring to external documentation.

2.2 HTTP methods

HTTP has several methods can be used, which are listed in Table 1:

HTTP methods	Explanations
HTTP GET	The HTTP GET method requests a representation of the resource

	specified by the URI. It is used to retrieve one or multiple resources from the server. The query component of the URI can be used for filtering purposes in case more than one resource is scoped by the path-abempty part of the URI. Only those resources passing the filtering criteria are returned.
HTTP HEAD	The HTTP HEAD method returns only the headers that are returned with a HTTP GET method together with the message body, except for the payload header fields. This method can be used to check if resources exist.
HTTP POST	The POST method sends data in the message body to the server. In contrast to HTTP PUT, replacing the resource representation, it requests the target resource to process the representation enclosed in the request according to the resource’s own specific semantics. With this method, it is possible to create a new resource. When a new resource is created, 201 (Created) is returned. The returned Location header carries the URI of the created resource. The URI of the new resource is created by the server. The response message body contains a representation of the created resource.
HTTP PUT	The HTTP PUT method requests that the resource representation of the target resource be created or replaced with the representation enclosed in the request message payload. This method replaces always the complete resource representation. Partial resource modifications are not possible. If a resource at the URI specified in the request does not exist yet, the server creates a new resource at this URI. Conditional requests (RFC 7232) using e.g. the entity tag (ETag) can be used to prevent accidentally overwriting modifications made to a resource by another client ("lost update problem").
HTTP DELETE	The DELETE method requests that the origin server deletes the resource identified by the Request-URI. This does not imply that the underlying information is deleted as well.
HTTP CONNECT	
HTTP OPTIONS	
HTTP TRACE	
HTTP PATCH	The HTTP PUT method only allows a complete resource replacement. For this reason, a new method, HTTP PATCH, has been defined by IETF in RFC 5789 for partial resource modifications. The set of changes to be applied is described in the request message body.

HTTP have already provided several methods to carry the interaction capabilities between managing and managed systems.

3. Benefits of introducing REST into Network management domain

REST provides a simplified mechanism to connect applications regardless of the technology or devices they use, or their location. They are based on industry standard protocols with universal vendor support that can leverage the internet for low cost communications, as well as other transport mechanisms. The loosely coupled messaging approach supports multiple connectivity and information sharing scenarios via services that are self describing and can be automatically discovered.

REST solutions uses HTTP as its operation protocols, which have been broadly used in the IT-industry for years, and it is mature and cost effective. The following features can be made use of when it is applied in the network management domain.

1) Good interoperability

REST solution has a good support for is universally interoperable, as far as the application support the globally used protocol HTTP, it can be connected to the REST environment.

2) Loosely coupled

Loosely coupled systems require a much simpler level of coordination and allow for more flexible reconfiguration, compared to tightly coupled systems.

REST solutions are self-describing software modules which encapsulates discrete functionality. REST-based services are accessible via standard Internet communication protocol HTTP directly. These services can be developed in any technologies (like C++, Java, .NET, PHP, Pearl etc.) and any application can access these services. So, the REST-based services are loosely coupled and can be used by applications developed in any technologies.

3) Broadly used

With the more rapid development of Internet-based technologies, REST-based services are now broadly used in IT services industry, for example e-business, business-to-business applications.

4) Low cost

REST APIs are open standards, and many tools, products, technologies is based on HTTP applications. This gives organizations a wide variety of choices, and they can select configurations that best meet their application requirements. Developers can enhance their productivity because with low cost, rather than having to develop their own solutions, they can choose from a ready market of off-the-shelf application components or third-party tools.

4. Proposal

It is proposed to start a new work item on “Guidelines for defining REST-based managed objects and management interfaces”.

The intended draft content for this work item may include the following aspects:

1) Principles for REST interface definitions

2) The containment relationship and naming rules for managed entities.

3) Definition of generic object accessing methods

Create, delete, getAttribute, setAttribute, etc.

4) The representation of object inheritance (limited to object attributes).

5) Guidelines for the definition of JSON-based information models

6) Conventions for REST interface definitions (Style)

7) Some common data types and exception definitions.

The outline text for the first draft can be found below :

Baseline text for “Guidelines for defining REST-based managed objects and management interfaces”

1 Scope

The network management architecture defined in [ITU-T M.3010] introduces the use of multiple management protocols. So far, the GDMO/CMIP, CORBA GIOP/IIOP, SMI/SNMP, 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 REST/SOAP is now an additional paradigm for network management.

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

– principles for REST interface definitions;

– containment relationship and naming rules for managed entities;

– generic accessing methods for managed objects;

– inheritance of managed objects and interfaces;

– information modelling guidelines for REST/JSON based interfaces;

– style conventions for REST/HTTP and JSON schema specifications;

– common data type and exception definitions.

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 (2011), *Management interface specification methodology*.
[RFC 4230]	IETF RFC 7230: *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*.
[RFC 7231]	IETF RFC 7231: *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*.
[RFC 3986]	IETF RFC 3986: *Uniform Resource Identifier (URI): Generic Syntax*.
[RFC 7232]	IETF RFC 7232, *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*.
[RFC 5789]	IETF RFC 5789, *PATCH Method for HTTP*.
[RFC 6902]	IETF RFC 6902, *JavaScript Object Notation (JSON) Patch*.
[RFC 6901]	IETF RFC 6901, *JavaScript Object Notation (JSON) Pointer*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

Editors’s Note: to be extended.

3.2 Terms defined in this Recommendation

This Recommendation does not define any new terms.

Editors’s Note: to be extended.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

HTTP	Hyper Text Transfer Protocol
JSON	JavaScript Object Notation
REST	REpresentational State Transfer
SOAP	Simple Object Access Protocol
TMN	Telecommunications Management Network
XML	extensible Markup Language
XSD	XML Schema Definition

Editors’s Note: to be extended.

5 Conventions

A few conventions are followed in this Recommendation to make the reader aware of the purpose of the text. While most of the Recommendation is normative, paragraphs succinctly stating mandatory requirements to be met by a management system (managing and/or managed) are preceded by a boldface "R" enclosed in parentheses, followed by a short name indicating the subject of the requirement, and a number. For example:

(R) EXAMPLE-1 An example mandatory requirement.

Requirements that may be optionally implemented by a management system are preceded by an "O" instead of an "R". For example:

(O) EXAMPLE-2 An example optional requirement.

The requirement statements are used to create compliance and conformance profiles.

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

```
{
    "title": "root",
    "items": {
        "title": "array item"
    }
}
```

6 Overview of a REST-based management framework

1) Managed object and interface definition guidelines:
    - definition of managed objects using JSON schema;
    - accessing methods for MOs;
    - inheritance of MOs and interface operations;
    - information modelling guidelines for REST-based interface operations;
    - style idioms for JSON schema specifications.
2) REST supporting services for network management:
    - definition of a REST-based notification services;
    - definition of a REST-based heartbeat service;
    - definition of a REST-based multiple object operations (MOO) service;
    - definition of a REST-based containment service.

This Recommendation mainly deals with the managed object and interface definition guidelines, and draft [ITU-T Q.rest] mainly deals with the REST-based supporting services for network management. The two Recommendations together form a REST-based management framework.

7 Principles for REST-based interface design

This clause identifies some interface design considerations that should be addressed by this framework through REST interfaces. It provides the modelling principles for REST-based managed objects and their accessing methods.

8 Definition of a generic managed object using XML schema

8.1 REST role in management interfaces

8.2 Definition of managed objects using JSON schema

8.2.1 Definition of a generic managed object class

8.2.2 Inheritance of managed objects

8.2.3 Common attributes and data types

9 Accessing methods for managed objects

10 Inheritance of managed objects and interface operations

10.1 Attributes inheritance of managed objects

10.2 Considerations for the inheritance of interface operation

11 Information modelling guidelines for REST-based interfaces

12 Style idioms for JSON schema specifications

12.1 Data model using JSON schema

12.2 JSON schema design considerations

12.3 Recommendations for schema developers

12.4 Guidelines for JSON schema extensions

13 Compliance and conformance

13.1 Standards document compliance

13.2 System conformance

13.3 Conformance statement guidelines

Annex A

Common REST-based JSON schema definitions

In this annex, the common definitions of REST interfaces as well as some common JSON schema based data types are defined.

A.1 JSON schema definitions for common data types and a generic managed object

Editors’s Note: to be extended.

A.1 justification for proposed draft new Recommendation X.xxx

Question:	7/2	**Proposed new ITU-T Recommendation**	July, 2018
Reference and title:	Recommendation ITU-T: "Guidelines for defining REST-based managed objects and management interfaces"		
Base text:		**Timing:**	2019
Editor(s):		**Approval process:**	AAP

Scope (defines the intent or object of the Recommendation and the aspects covered, thereby indicating the limits of its applicability):

The network management architecture defined in [ITU-T M.3010] introduces the use of multiple management protocols. So far, the GDMO/CMIP, CORBA GIOP/IIOP, SMI/SNMP, 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 REST/SOAP is now an additional paradigm for network management.

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

- principles for REST interface definitions;
- containment relationship and naming rules for managed entities;
- generic accessing methods for managed objects;
- inheritance of managed objects and interfaces;
- information modelling guidelines for REST/JSON based interfaces;
- style conventions for REST/HTTP and JSON schema specifications;
- common data type and exception definitions.

Summary (provides a brief overview of the purpose and contents of the Recommendation, thus permitting readers to judge its usefulness for their work):

This document defines a set of guidelines for managed object modelling and a management interface for REST-based network management. It composes a framework for REST-based network management interfaces along with draft [ITU-T Q.rest]. It specifies how REST-based management interfaces should be defined. It covers generic accessing methods of XML-based managed objects, information modelling in REST/HTTP and JSON schema. Some HTTP requests/responses and JSON schema are provided for defining some basic data types: generic managed object (MO) and generic MO accessing methods. This document and draft ITU-T Q.rest together compose a framework for REST-based network management interfaces with a wide range of applications.

Relations to ITU-T Recommendations or to other standards (approved or under development):

[ITU-T M.3020]: provides Management interface specification methodology. This draft Recommendation has this methodology as one of its basis.

Liaisons with other study groups or with other standards bodies:

3GPP SA5

Supporting members that are committing to contributing actively to the work item:

BUPT, China Telecom, CATTSoft, Telnor
