

Recommendation

ITU-T Q.4077 (04/2025)

SERIES Q: Switching and signalling, and associated measurements and tests

Testing specifications – Testing specifications for IMT-2020 and IoT

**Testbed as a service application program
interfaces descriptions and interoperability
requirements**

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
Testing specifications for next generation networks	Q.3900-Q.3999
Testing specifications for SIP-IMS	Q.4000-Q.4039
Testing specifications for Cloud computing	Q.4040-Q.4059
Testing specifications for IMT-2020 and IoT	Q.4060-Q.4099
PROTOCOLS AND SIGNALLING FOR PEER-TO-PEER COMMUNICATIONS	Q.4100-Q.4139
PROTOCOLS AND SIGNALLING FOR COMPUTING POWER NETWORKS	Q.4140-Q.4159
PROTOCOLS AND SIGNALLING FOR QUANTUM KEY DISTRIBUTION NETWORKS	Q.4160-Q.4179
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.4077

Testbed as a service application program interfaces descriptions and interoperability requirements

Summary

Recommendation ITU-T Q.4077 reports on the elaboration of testbed as a service (TaaS) application program interfaces (APIs) based on the requirements and reference model with properties of relevance for delivering testbed as a service (TaaS), to complement and extend Recommendation ITU-T Q.4068. Recommendation ITU-T Q.4077 is more particularly focused on the user interface, services, and requirements to address end-user needs when remotely accessing testbeds through APIs in order to deliver adequate user experience. From this perspective, the Recommendation elaborates the related terms and definitions, requirements, and reference model with properties of relevance for TaaS. Recommendation ITU-T Q.4077 also presents interoperability requirements for virtualizing and delivering modular and scalable TaaS on top of existing and future testbed infrastructures, including federated testbeds. TaaS is able to list the assets provided by the different testbeds and expose them through dedicated APIs based on Recommendation ITU-T Q.4068.

History*

Edition	Recommendation	Approval	Study Group	Unique ID
1.0	ITU-T Q.4077	2025-04-13	11	11.1002/1000/16293

Keywords

API, federated testbeds, interoperability, testbed as a service.

* To access the Recommendation, type the URL <https://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, and 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 <https://www.itu.int/ITU-T/ipr/>.

© ITU 2025

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

	Page
1 Scope.....	1
2 References.....	1
3 Definitions	1
3.1 Terms defined elsewhere	1
3.2 Terms defined in this Recommendation.....	1
4 Abbreviations and acronyms	2
5 Conventions	2
6 TaaS API and interoperability requirements	2
6.1 APIa.....	7
6.2 APIb.....	7
6.3 APIc.....	7
6.4 APId.....	8
6.5 APIe.....	8
6.6 APIf	8
6.7 APIg.....	8
6.8 APIh.....	8
6.9 APIi.....	9
6.10 APIj.....	9
6.11 APIk.....	9
6.12 APIl/GUI_l	9
6.13 APIm/GUI_m	10
6.14 APIn.....	11
6.15 APIo.....	11
6.16 APIp.....	12
6.17 APIq.....	12
6.18 APIr	12
6.19 APIs	12
6.20 APIt.....	13
6.21 APIu.....	13
6.22 APIv.....	13
6.23 APIw	13
6.24 APIx.....	14
6.25 APIy/GUI_y	14
6.26 APIz.....	15
Appendix I – Instantiation of generic APIs	16
I.1 TM Forum Business API.....	16
I.2 BSS/OSS APIs.....	16
I.3 Customer-facing APIs	16

	Page
I.4 [IEEE 2302]	16
I.5 Comparison between [ITU-T Q.4068] and [IEEE 2302] APIs	20
Bibliography.....	21

Recommendation ITU-T Q.4077

Testbed as a service application program interfaces descriptions and interoperability requirements

1 Scope

This Recommendation describes the testbed as a service application program interfaces (APIs) and interoperability requirements. The APIs specified in this Recommendation are dedicated exclusively to testbed as a service (TaaS). The integration, interoperability and extensibility of the TaaS are also studied in this Recommendation.

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 Q.4068] Recommendation ITU-T Q.4068 (2021), *Open application program interfaces (APIs) for interoperable testbed federations*.
- [ITU-T Q.4078] Recommendation ITU-T Q.4078 (2025), *User requirements and reference model for testbed as a service*.
- [IEEE 2302] IEEE 2302-2021, *IEEE Standard for Intercloud Interoperability and Federation (SIIF)*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

- 3.1.1 experiment** [b-ISO 3534-3]: Purposive investigation of a system through selective adjustment of controllable conditions and allocation of resources.
- 3.1.2 resource** [b-ETSI TS 102 812]: A well-defined capability or asset of a system entity, which can be used to contribute to the realization of a service. Examples: MPEG decoder, graphics system.
- 3.1.3 testbed** [ITU-T Q.4068]: Platform to realise scientific tests with new technologies on an environment fully controlled by experimenters.
- 3.1.4 testbed as a service (TaaS)** [b-ITU-T QSTR.FTT]: Refers to the offering of a testbed by its owner to users based on a defined policy framework. This framework may include service level agreements (SLAs), pricing models, usage time frames, and testbed availability. Users can access the testbed to run tests, analyse results, and utilize the findings while adhering to the established policies.

3.2 Terms defined in this Recommendation

None.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

API	Application Program Interface
BSS	Business Support Systems
CUT	Component Under Test
E2E	End-to-End
FHS	Fed Hosting Server
GUI	Graphical User Interface
HTTPS	Hypertext Transfer Protocol Secure
IEEE	Institute of Electrical and Electronics Engineers
MPEG	Moving Picture Experts Group
OSS	Operations Support Systems
SIIF	Standard for Intercloud Interoperability and Federation
SLA	Service Level Agreement
SUT	System Under Test
TaaS	Testbed as a Service

5 Conventions

None.

6 TaaS API and interoperability requirements

This Recommendation is dedicated to all the TaaS APIs which are based on the generic reference model defined in [ITU-T Q.4068] and [ITU-T Q.4078]. These APIs cover all aspects of the monetization of TaaS. Through the APIs specified in this Recommendation, it is possible for users, for example, to reserve a testbed slice, or to reuse an experiment based on a template of a testbed slice. Furthermore, if the user does not find a service through the TaaS APIs, a request to a specific endpoint of the TaaS APIs can be made by the user to execute the service on demand.

With reference to Figure 1 and Table 1 of [ITU-T Q.4068], the APIs needed for the TaaS are listed in Table 1.

Table 1 – Testbed as a service APIs

#	API name (identification tags)	API description
1	APIa	The API is used for providing descriptive information about the resource, its state and usage in real-time upon the invocation of the API by the testbed management system.
2	APIb	The API is used for enabling a test manager to interact with the resource, in cases where the resource is either a component under test (CUT) or system under test (SUT), or provides an interface that can be used by a test system to configure it, such that the test manager may configure the resource as may be required for some test scenario, and/or pull test results from the resource after a completion of a test.

Table 1 – Testbed as a service APIs

#	API name (identification tags)	API description
3	APIc	The API is used by Level-0 resources to dynamically provide information in real-time about their state in terms of usage and other information such as performance data and workload being sustained on the resource.
4	APId	The API is used by Level-1 resources to dynamically provide information in real-time about their state in terms of usage and other information such as performance data and workload being sustained on the resource.
5	APIe	The API is used by testbed management system to enable testbed admin to pull and view (visualize) information about the state of resources (Level-0 and Level-1) from the real-time state repository, especially when the testbed admin intends to view the state of certain resources before deciding to cause connectivity to be established among resources in the testbed and/or establishing connectivity to resources in other testbeds.
6	APIf	The API is used by test manager to pull information about Level-0 and Level-1 resources that may be required to participate in a certain test scenario a testbed user may want to execute. The API is also meant for use by a test manager to pull information about the state of resources (Level-0 and Level-1) from the real-time state repository, especially when state of a resource plays a role during the execution of certain test cases or when the test manager may require to use the state of certain resources in deciding to cause connectivity to be established among resources in the testbed and/or establishing connectivity to resources in other testbeds.
7	APIg	The API is used by a test manager to execute certain test cases that are meant to test the resource during a scenario in which the resource is a system under test (SUT) or a component under test (CUT). The API is also meant for use by a test manager to request the resource to execute a certain behaviour that is required by a test case(s) being executed by the test manager, including configuring some Level-0 resource(s) that can only be configured via the Level-1 resource.
8	APIh	The API is used by the testbed resource broker to gather information about the capabilities of the resource, its state and its availability to serve testbed services request(s) received from prospective testbed user(s). In case the resource is an orchestrator of resources (Level-1 and/or Level-0), then the same API is also used by the testbed resource broker to request the resource for orchestration of instance(s) of certain Level-1 resource(s) and/or Level-0 resource(s) as slices that are required to fulfil requirements of a testbed service request received from a prospective testbed user, when there is no existing instance (slice) of the required type of resource(s) that is already available to fulfil the requirements of the newly received request for a prospective testbed user. The same API is also used by testbed broker to obtain information about capabilities and state of resources

Table 1 – Testbed as a service APIs

#	API name (identification tags)	API description
		directly under the management and control responsibility of the Level-1 resource.
9	APIi	The API is used for providing descriptive information about the resource, its state and usage in real-time upon the invocation of the API by the testbed management system.
10	APIj	The API is used for providing updates to descriptive information about the resource, its state and usage in real-time upon the invocation of the API the Level-1 resource, whenever there are changes that have occurred on the resource.
11	APIk	The API is used for providing updates to descriptive information about the resource, its state and usage in real-time upon the invocation of the API the Level-0 resource, whenever there are changes that have occurred on the resource.
12	APIl (or GUI_1)	The GUI is to be used by the broker administrator (broker admin) for performing all necessary broker governance and management activities and operations on the broker. For example, the broker admin installs the policies that govern the operation of the broker in terms of how it registers the testbed domain to the inter-testbed E2E universal resource broker for testbeds federation. It also installs policies that govern the services offered to prospective users of testbeds (including the policies for testbed usage by prospective users). Via the GUI (API), the broker admin can manage the broker to prepare the broker in such a way that the broker can register with the inter-testbed E2E universal resource broker for testbeds federation. In this case, the broker is ready to provide its services to prospective testbed users.
13	APIm (or GUI_m)	The API is to be used by the testbed administrator (testbed admin) for performing all necessary testbed management activities and operations. Via the GUI (API), the testbed admin can manage the testbed to prepare the testbed in such a way that the testbed can provide testbed services to prospective users and can also participate in testbed federations with other testbeds and the inter-testbed E2E universal resource broker for testbeds federation. The GUI of the testbed management system is used by the testbed administrator in establishing connectivity of the testbed with other testbeds that should be interconnected with this testbed in order to provide federated capabilities and resources to prospective users of federated testbeds.
14	APIn	The API is used by the testbed resource broker to register itself into the testbed management system, provide descriptive state and change of state information in real-time to the testbed management system. The API is also used by the testbed resource broker to obtain descriptive information about all resources and other entities of the testbed domain and their capabilities descriptions (as the various resources and entities not only update the real-time state repository but the testbed management system as well, and the information is kept in sync and consistent between the testbed management system and the real-time state repository).

Table 1 – Testbed as a service APIs

#	API name (identification tags)	API description
		NOTE – The testbed resource broker may use an API provided by the real-time state repository for directly pulling out information about resources available in the testbed domain and their capabilities descriptions.
15	APIo	The API is used by a test manager to register itself into the testbed management system, provide descriptive state and change of state information in real-time to the testbed management system, because a test manager could be a considered as a resource itself.
16	APIp	The API is used for providing descriptive information about a test manager, its state and usage in real-time upon the invocation of the API by the testbed management system.
17	APIq	The API is used for providing test results to a test manager(s) that involved the resource in a test case as a component under test (CUT) or system under test (SUT). The same API is also used for communicating to a test manager some feedback (e.g., errors or failures during the execution) to some invocations triggered earlier on the resource by the test manager.
18	APIr	The API is to be used by a test suite/cases designer and test executer (upon the acceptance of its request for testbed service by the testbed resource broker) to connect to the test manager instance assigned to the testbed user to use the test manager to design, compile and run test cases, or to upload and compile some test cases designed offline and execute them. Through the API, the testbed user is able to upload some test cases or test suites if the testbed domain allows that and then compile and/or execute the test cases, or the user is only allowed to design, compile and execute test cases directly on the test manager without uploading test cases/suites from outside.
19	APIs	The API is used by the inter-testbed E2E universal resource broker for testbeds federation to connect to the test manager, e.g., to enable the E2E resource broker to access state information about the specific test manager, or for cases whereby some test results could be shared to the testbed user via the E2E resource broker if not possible that the test manager provides direct access to those kinds of test results directly to the user (test executor), though primarily the test executor should be able to access test results directly from the test manager(s). APIs is using the same uniform resource description model that APIx uses.
20	APIt	The API is used for providing test results to a test manager(s) that involved the resource in a test case as a component under test (CUT). The same API is also used for communicating to a test manager some feedback (e.g., errors or failures during the execution) to some invocations triggered earlier on the resource by the test manager.
21	APIu	The API is used by the testbed management system to keep synchronizing with the testbed resource broker on the state of the broker. The same API is also used by the testbed management system to provide updates to any changes in the descriptive information about all resources and other entities of the testbed

Table 1 – Testbed as a service APIs

#	API name (identification tags)	API description
		domain and their capabilities descriptions (Information that is kept in sync and consistent between the testbed management system and the real-time state repository).
22	APIv	The API is used by a Level-1 resource to push updated Information (updates) about state of resources under the management and control responsibility of the Level-1 resource and their capabilities descriptions, and any changes that may have occurred to the resources and capabilities. The same API is also used for synchronizations between the Level-1 resource and the testbed resource broker.
23	APIw	The API is used by the inter-testbed E2E universal resource broker for testbeds federation, after the testbed resource broker has registered itself with it via APIx, to then obtain (pull) descriptive information about all testbed resources available in the Testbed domain to serve testbed services requests that may come from the E2E resource broker and their capabilities descriptions. The same API is also used by the E2E resource broker to provide synchronization related descriptive state and change of state information in real-time to the test broker. APIw uses the same uniform resource description model that APIx uses.
24	APIx	The API is used by the testbed resource broker to push updated information (updates) about state of resources of the testbed domain and their capabilities descriptions, and any changes that may have occurred to the resources and capabilities. The same API is also used, complementarily to APIw, for synchronizations between the testbed resource broker and the inter-testbed E2E universal resource broker for testbeds federation. Complementarily to APIs, APIx is used to synchronize information between the test manager and the inter-testbed E2E universal resource broker for testbeds federation. APIx uses the same uniform resource description model that APIw uses.
25	APIy (or GUI_y)	The API is to be used by the broker administrator (broker admin) for performing all necessary broker governance and management activities and operations on the broker. For example, the broker admin installs the policies that govern the operation of the broker in terms of admitting (or not admitting) testbed domains in their attempts to discover and register with the broker, as well as policies that govern the services offered to prospective users of testbeds registered with broker (including the policies for testbeds user registrations). Via the GUI (API), the broker admin can manage the broker to prepare the broker in such a way that the broker can expose the APIz and any GUIs of the broker that can be made available to prospective testbeds users, such that the broker is ready to provide its services to prospective testbed users and to testbeds intending to register with it.
26	APIz	This API provides the entry point into the system of federated testbeds to prospective users (testbed users) of the system of federated testbeds. It provides 'search and query and find services' that enable the prospective user of testbed service(s), i.e., the test

Table 1 – Testbed as a service APIs

#	API name (identification tags)	API description
		suite/cases designer and test executor to find/discover testbeds that are available to accept new requests within the time of interest to the prospective testbed user as well as their capabilities topology information pertaining to their interconnection and federations with other testbeds. A prospective testbed user (test suite/cases designer and test executor) can query the broker for testbeds that fulfil certain capabilities and requirements such as end-to-end latency within the scope of the single testbed or across multiple testbeds, before the prospective user can then select testbeds and launch requests for testbed services. Then, the prospective user can contact the APIr of the different testbed domains to create new experiments, new test cases and test suites.

6.1 APIa

The APIa provides the description, the state and the usage of a given resource in real-time.

The calls related to the APIa are the following:

- getResourceDescription: an HTTPS GET message for receiving a response that contains the description of a given resource.
- getResourceState: an HTTPS GET message for retrieving the current state of a given resource.
- getResourceUsage: an HTTPS GET message for obtaining the usage in real-time of a given resource.

6.2 APIb

The APIb is used by a test manager to configure a resource.

The calls for the APIb are the following:

- interactWithComponent: a HTTPS POST message for triggering an action on a component under test.
- interactWithSystem: an HTTPS POST message for triggering an action on a system under test.
- getConfigurationInterface: an HTTPS GET message for obtaining information on an interface used to configure a component or a system under test.
- configureResource: an HTTPS POST message for configuring a given resource.
- getResult: an HTTPS GET message for retrieving the results of a test from a given resource.

6.3 APIc

The APIc retrieves in real-time the state, the usage, the performance and the workload of a resource of the Level-0 type.

The calls of the APIc are the following:

- getState: an HTTPS GET message for retrieving the state in real-time of a resource of the type Level-0.
- getUsage: an HTTPS GET message for obtaining the usage in real-time of a Level-0 resource.

- **getInformation:** an HTTPS GET message for retrieving all the information in real-time of a Level-0 resource. The information provided by a given Level-0 resource can be the performance, the workload and the energy consumption.

6.4 API_d

The API_d retrieves in real-time the state, the usage, the performance and the workload of a resource of the Level-1 type.

The calls for the API_d are the following:

- **getState:** an HTTPS GET message for retrieving the state in real-time of a resource of the type Level-1.
- **getUsage:** an HTTPS GET message for obtaining the usage in real-time of a Level-1 resource.
- **getInformation:** an HTTPS GET message for retrieving all the information in real-time of a Level-1 resource. The information provided by a given Level-1 resource can be the performance, the workload and the energy consumption.

6.5 API_e

The API_e permits to a testbed administrator to retrieve and see the current state of a resource.

The calls used by the API_e are the following:

- **getState:** an HTTPS GET message for obtaining the state in real-time of a set of resources from different types (Level-0 and Level-1).
- **viewState:** an HTTPS GET message for obtaining a view showing in real-time the state of a set of resources.

6.6 API_f

The API_f is used by a test manager to retrieve the current state of a resource.

The calls dedicated to the API_f are the following:

- **getInformation:** an HTTPS GET message for obtaining the information of a set of resources. This set can include resources from the Level-0 and Level-1 types.
- **getState:** an HTTPS GET message for obtaining the state of resources from the real-time state repository.

6.7 API_g

The API_g allows a test manager to execute a test on a given resource.

The calls for the API_g are the following:

- **executeSystemTest:** an HTTPS POST message for starting a test on a resource which is a SUT.
- **executeComponentTest:** an HTTPS POST message for starting a test on a resource which is a CUT.
- **configureLevel0Resource:** an HTTPS POST message for configuring a Level-0 resource. This message is sent to a Level-1 resource which effectively configures the Level-0 resource.
- **configureLevel1Resource:** an HTTPS POST message for configuring a Level-1 resource.

6.8 API_h

The API_h is used by the testbed resource broker to collect the capabilities, the state and the availability of a resource.

The calls of the APIh are the following:

- `getResourceCapabilities`: an HTTPS GET message for retrieving all the capabilities of a given resource.
- `getResourceState`: an HTTPS GET message for obtaining the state of a given resource.
- `getResourceAvailability`: an HTTPS GET message for retrieving the availability of a given resource.

6.9 APIi

The APIi provides the description, the state and the usage of a resource in real-time to the testbed management system.

The calls related to the APIi are the following:

- `getResourceDescription`: an HTTPS GET message for obtaining the description of a given resource.
- `getResourceState`: an HTTPS GET message for obtaining the current state of a given resource.
- `getResourceUsage`: an HTTPS GET message for obtaining the usage in real-time of a given resource.

6.10 APIj

The APIj provides the updated information on the description, the state and the usage of a Level-1 resource.

The calls related to the APIj are the following:

- `subscribeResourceDescription`: an HTTPS POST message for subscribing to any changes of the description of a given resource. The resource is from the Level-1 type.
- `subscribeResourceState`: an HTTPS POST message for subscribing to any changes of the state of a given resource of the Level-1 type.
- `subscribeResourceUsage`: an HTTPS POST message for subscribing to any changes of the usage of a Level-1 resource.

6.11 APIk

The APIk provides the updated information on the description, the state and the usage of a Level-0 resource.

The calls dedicated to the APIk are the following:

- `subscribeResourceDescription`: an HTTPS POST message for subscribing to any changes of the description of a given resource. The resource is from the Level-0 type.
- `subscribeResourceState`: an HTTPS POST message for subscribing to any changes of the state of a given resource of the Level-0 type.
- `subscribeResourceUsage`: an HTTPS POST message for subscribing to any changes of the usage of a Level-0 resource.

6.12 APIl/GUI_1

The APIl, also named GUI_1, permits to the broker administrator to access the testbed resource broker. It is composed by a graphical user interface (GUI) accessible by the broker administrator.

The different calls related to this API are the following:

- **brokerAdminLogin**: an HTTPS POST message which contains the username and the password of the broker administrator. This API call allows the authentication of the broker administrator on the GUI used to manage the broker.
- **installBrokerPolicy**: an HTTPS POST message composed by a policy to be applied on the broker. The API call permits the instantiation of a broker policy.
- **getBrokerPolicies**: an HTTPS GET message for obtaining all the policies applied on the broker.
- **getBrokerPolicy**: an HTTPS GET message composed by the identifier of a policy. This API call retrieves the policy based on its identifier.
- **updateBrokerPolicy**: an HTTPS PUT message for updating a broker policy. The message contains the policy to be updated in the broker.
- **deleteBrokerPolicy**: an HTTPS DELETE message for erasing a broker policy from the broker. The identifier of the broker policy is given in the message.
- **registerTestbed**: an HTTPS POST message for registering a new testbed in the universal resource broker. This message contains the information of the testbed to be registered.
- **getTestbeds**: an HTTPS GET message for obtaining the list of all the testbeds registered in the broker.
- **getTestbed**: an HTTPS GET message with the identifier of a particular testbed. This call retrieves the testbed registered in the broker, based on the given identifier.
- **updateTestbed**: an HTTPS PUT message for modifying in the broker the information of the testbed included in the message.
- **deleteTestbed**: an HTTPS DELETE message for erasing the testbed from the broker. The identifier of the testbed is available in the message.
- **installUserPolicy**: an HTTPS POST message composed by a policy to be applied on the broker. The API call permits the instantiation of a user policy.
- **getUserPolicies**: an HTTPS GET message for obtaining all the policies applied on the broker.
- **getUserPolicy**: an HTTPS GET message composed by the identifier of a user policy. This API call retrieves the policy based on its identifier.
- **updateUserPolicy**: an HTTPS PUT message for updating a user policy. The message contains the policy to be updated in the broker.
- **deleteUserPolicy**: an HTTPS DELETE message for erasing a user policy from the broker. The identifier of the user policy is given in the message.
- **publishService**: an HTTPS POST message for publishing a new service.
- **getServices**: an HTTPS GET message for obtaining all the services listed in the broker.
- **getService**: an HTTPS GET message for retrieving the service associated to the given identifier.
- **updateService**: an HTTPS PUT message for modifying an existing service registered in the broker. The service to be updated is given in the message.
- **deleteService**: an HTTPS DELETE message for erasing a service from the broker. The identifier of the service to be deleted is given in the message.

6.13 API_m/GUI_m

The API_m, as known as GUI_m, allows the testbed administrator to access the testbed management system. Through this GUI, the testbed administrator manages the testbed, including all the operations needed to make the testbed interoperable inside a testbeds federation.

The related API calls are the following:

- **testbedAdminLogin**: an HTTPS POST message for containing the username and the password of the testbed administrator. This API call allows the authentication of the testbed administrator on the GUI used to manage the testbed.
- **publishTestbedService**: an HTTPS POST message for publishing a new service. The message contains the new service to be published.
- **getTestbedServices**: an HTTPS GET message for obtaining all the services listed in the testbed.
- **getTestbedService**: an HTTPS GET message for retrieving the testbed service associated to the identifier given in the message.
- **updateTestbedService**: an HTTPS PUT message for modifying an existing service registered in the testbed. The service to be updated is given in the message.
- **deleteTestbedService**: an HTTPS DELETE message for erasing a service from the testbed. The identifier of the service to be deleted is in the message.
- **connectTestbedFederation**: an HTTPS POST message for connecting a testbed to a federation of testbeds. The message contains the identifier of the testbed, the identifier of the federation and other parameters needed for the connection.
- **disconnectTestbedFederation**: an HTTPS PUT message for disconnecting a testbed from a testbeds federation. The message contains the identifier of the testbed and the identifier of the federation.
- **getFederation**: an HTTPS GET message for obtaining the information about a particular testbeds federation. The identifier of the federation is given in the message.
- **getTestbed**: an HTTPS GET message for retrieving all the information on a given testbed. The message contains the identifier of the testbed.

6.14 API_{In}

The API_{In} is used by the testbed resource broker to register itself into the testbed management system.

The calls of the API_{In} are the following:

- **registerTestbedResourceBroker**: an HTTPS POST message for registering a testbed resource broker into the testbed management system.
- **sendTestbedResourceBrokerStateDescription**: an HTTPS POST message for sending the description of the state to the testbed management system.
- **sendTestbedResourceBrokerState**: an HTTPS POST message for sending the state in real-time to the testbed management system.
- **getResourceDescription**: an HTTPS GET message for obtaining the description of all the resources available in a testbed.
- **getResourceCapabilities**: an HTTPS GET message for retrieving the capabilities of all the resources of a given testbed.

6.15 API_{Io}

The API_{Io} is used by a test manager to register itself into the testbed management system.

The calls of the API_{Io} are the following:

- **registerTestManager**: an HTTPS POST message for registering a test manager into the testbed management system.
- **sendTestManagerStateDescription**: an HTTPS POST message for sending the description of the state of a test manager.

- sendTestManagerState: an HTTPS POST message for sending the state in real-time of a test manager to the testbed management system.

6.16 APIp

The APIp provides the description, the state and the usage of a test manager in real-time.

The calls of the APIp are the following:

- getTestManagerDescription: an HTTPS GET message for retrieving the description of a test manager.
- getTestManagerState: an HTTPS GET message for obtaining the state of a test manager.
- getTestManagerUsage: an HTTPS GET message for retrieving the usage in real-time of a test manager.

6.17 APIq

The APIq retrieves the results of a test and sends them to a test manager.

The calls of the APIq are the following:

- getResultsFromCUT: an HTTPS GET message for obtaining the results of a test involving a component under test (CUT).
- getResultsFromSUT: an HTTPS GET message for retrieving the results of a test for a system under test (SUT).
- subscribeErrors: an HTTPS POST message for subscribing to the errors encountered during the execution of the test.

6.18 APIr

The APIr lets access the test suite/cases designers and test executors to the test managers. The APIr gives the possibility to the test suite/cases designers to create new experiments, new test cases and new test suites. Then, test executors can run the different kinds of tests through the APIr. Stored test cases can be retrieved to be executed by the test executors.

The API calls are the following:

- testbedUserLogin: an HTTPS POST message containing the username and the password of the user who will designed and run a test. This API call allows the authentication of the testbed user to access the interface for the creation of tests.
- designTest: an HTTPS POST message for creating a new test. The test is included in the message.
- compileTest: an HTTPS POST message for launching the compilation of a test. The identifier of the test to be compiled is given in the message.
- runTest: an HTTPS POST message for executing the test given by the parameter named identifier.
- saveTest: an HTTPS POST message for saving a test. The message contains the test identifier and the location where to store the test.
- uploadTest: an HTTPS GET message for uploading a test to be executed.

6.19 APIs

The APIs is employed to connect the inter-testbed E2E universal resource broker for testbeds federation to the test manager.

The calls of the APIs are the following:

- `getTestManagerState`: an HTTPS GET message for retrieving the state of a test manager.
- `getTestResults`: an HTTPS GET message for obtaining the results of a given test.

6.20 API_t

The API_t provides the test results to a test manager.

The calls of the API_t are the following:

- `getResultsFromCUT`: an HTTPS GET message for obtaining the results of a test involving a component under test (CUT).
- `subscribeErrors`: an HTTPS POST message for subscribing to the errors encountered during the execution of the test.

6.21 API_u

The API_u synchronizes the testbed management system and the testbed resource broker.

The calls of the API_u are the following:

- `synchronize`: an HTTPS POST message for triggering the synchronization between the testbed resource broker and the testbed management system.
- `subscribeResourceDescription`: an HTTPS POST message for subscribing to any changes in the description of the resources.
- `subscribeResourceCapabilities`: an HTTPS POST message for subscribing to any changes in the capabilities of the resources.

6.22 API_v

The API_v is used by a Level-1 resource to push its state and its capabilities to the testbed resource broker.

The calls of the API_v are the following:

- `sendResourceState`: an HTTPS POST message for sending the state of a given resource of the Level-1 type.
- `sendResourceCapabilities`: an HTTPS POST message for sending the capabilities of a Level-1 resource.
- `synchronize`: an HTTPS POST message for doing the synchronization between a Level-1 resource and the testbed resource broker.

6.23 API_w

The API_w is employed by the inter-testbed E2E universal resource broker for testbeds federation to retrieve all the information related to all the available resources.

The calls dedicated to the API_w are the following:

- `getResourceDescription`: an HTTPS GET message for obtaining the description of all the resources available in the testbed.
- `getResourceCapabilities`: an HTTPS GET message for obtaining the capabilities of all the resources available in a testbed.
- `synchronize`: an HTTPS POST message for starting the synchronization between the inter-testbed E2E universal resource broker and the testbed resource broker.

6.24 APIx

The APIx allows the testbed resource broker to push updated information on the state and the capabilities of the resources.

The calls of the APIx are the following:

- **sendResourceState**: an HTTPS POST message for sending the state of the resources.
- **sendResourceCapabilities**: an HTTPS POST message generated by the testbed resource broker for sending the capabilities of the resources.
- **synchronize**: an HTTPS POST message for doing the synchronization between the testbed resource broker and the inter-testbed E2E universal resource broker.

6.25 APIy/GUI_y

The APIy, also named GUI_y, gives access to the inter-testbed E2E universal resource broker for the testbeds federation to the broker administrator. The activities done by a broker administrator through the APIy consist to apply the governance policies for the broker, the operations to discover and register the resources provided by the testbed through the broker. The APIy permits to expose the endpoints of the testbed made available by the broker.

The different calls for this API are the following:

- **brokerAdminLogin**: an HTTPS POST message containing the username and the password of the broker administrator. This API call allows the authentication of the broker administrator on the GUI used to manage the broker.
- **installBrokerPolicy**: an HTTPS POST message composed by a policy to be applied on the broker. The API call permits the instantiation of a broker policy.
- **getBrokerPolicies**: an HTTPS GET message for obtaining all the policies applied on the broker.
- **getBrokerPolicy**: an HTTPS GET message composed by the identifier of a policy. This API call retrieves the policy based on its identifier.
- **updateBrokerPolicy**: an HTTPS PUT message for updating a broker policy. The message contains the policy to be updated in the broker.
- **deleteBrokerPolicy**: an HTTPS DELETE message for erasing a broker policy from the broker. The identifier of the broker policy is given in the message.
- **admitTestbed**: an HTTPS POST message containing a testbed to be approved in the federation of testbeds.
- **refuseTestbed**: an HTTPS POST message for withdrawing a testbed from the testbeds federation. The identifiers of the testbed and the federation are included in the message.
- **installUserPolicy**: an HTTPS POST message composed by a policy to be applied on the broker. The API call permits the instantiation of a user policy.
- **getUserPolicies**: an HTTPS GET message for obtaining all the policies applied on the broker.
- **getUserPolicy**: an HTTPS GET message composed by the identifier of a user policy. This API call retrieves the policy based on its identifier.
- **updateUserPolicy**: an HTTPS PUT message for updating a user policy. The message contains the policy to be updated in the broker.
- **deleteUserPolicy**: an HTTPS DELETE message for erasing a user policy from the broker. The identifier of the user policy is included in the message.
- **publishService**: an HTTPS POST message for publishing a new service.
- **getServices**: an HTTPS GET message for obtaining all the services listed in the broker.

- getService: an HTTPS GET message for retrieving the service associated to the identifier included in the message.
- updateService: an HTTPS PUT message for modifying an existing service registered in the broker. The service to be updated is given in the message.
- deleteService: an HTTPS DELETE message for erasing a service from the broker. The identifier of the service to be deleted is given in the message.
- discoverResources: an HTTPS GET message for retrieving all the available resources.
- registerResource: an HTTPS POST message for registering a new resource.
- unregisterResource: an HTTPS DELETE message for removing an existing resource.

6.26 APIz

The APIz allows the test suite/cases designer and the test executer to reach out the inter-testbed E2E universal resource broker for testbeds federation. The APIz provides the necessary interface offered to the testbed users to search, query and find all the services of the Testbed as a Service. This APIz publishes all the information useful to the testbed users such as testbed capabilities, testbed topology and testbed features inside the testbeds federation.

The calls for this API are the following:

- searchTestbed: an HTTPS GET message for finding a testbed or several testbeds. The call returns the testbeds.
- queryTestbed: an HTTPS GET message for obtaining specific information on a given testbed. The identifier of the testbed is included in the message.
- findTestbedServices: an HTTPS GET message for obtaining all the services available in the given testbed.
- selectTestbeds: an HTTPS POST message for selecting the different testbeds for an experiment.

Appendix I

Instantiation of generic APIs

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

I.1 TM Forum Business API

The TM Forum Business API ecosystem [b-TMF-business-API] allows the monetization of the services provided through the testbed as a service. This API contains notably the licenses and the service level agreement (SLA) elements. It also provides an accountability service. This API permits the management of all the assets provided through the TaaS. The TM Forum Business API allows the creation of pricing plans such as one-time payment, pay-per-use payment and subscription. This API provides the necessary endpoints in function of the applied payment models. An instantiation of the TM Forum Business API can be used for the monetization of the services provided by the TaaS.

I.2 BSS/OSS APIs

Business support systems (BSS) and operations support systems (OSS) allow telecommunications network operators to activate and configure different resources for their customers. Furthermore, they enable a good management of their inventory and catalogue. For example, TM Forum publishes an OSS/BSS blueprint for the development of BSS/OSS solutions [b-TMF-OSS-BSS]. This toolkit offered by TM Forum is composed by TM Forum Open APIs, related data models and several guides to help the developers to build OSS/BSS solutions. An instantiation of the TM Forum OSS/BSS toolkit could be done in a testbeds federation for the management of all the resources and the related operations executed on these resources.

I.3 Customer-facing APIs

The customer-facing APIs are used by the users to register to the TaaS and then, to log in. They also allow the users to select which data should be shared, in particular personal data. The users can also choose the services provided by a federation of testbeds and configure them to their needs. This includes in particular the creation of the tests, their compilation, their execution and the retrieval of the test results. The customer-facing APIs enable the centralization and the display of the data provided by the components of the TaaS and the other components of a testbeds federation. Furthermore, the customer-facing APIs are instantiated on top of the TM Forum Business API to permit the monetization of the services. The customer-facing APIs are of course directly linked to the BSS/OSS APIs: indeed, they provide the information given by the users to the BSS/OSS components through the BSS/OSS APIs.

NOTE – In this context, the customers are the users of the federated testbeds.

I.4 [IEEE 2302]

IEEE 2302-2021 [IEEE 2302] is an IEEE standard for intercloud interoperability and federation (SIIF) [b-TBFG-I-028R1]. Several APIs related to testbeds federations are described and can be used in the context of the TaaS.

The first API is the FHS Operator API which permits management of a fed hosting server (FHS) and the communication between two fed hosting servers. Table I.1 presents the FHS Operator API.

Table I.1 – Fed Host Server (FHS) Operator API

HTTP request method	Endpoint	Description
POST	/FHSOperator/NewFedAdmin	Add a new administrator of the federations.
GET	/FHSOperator/FedAdmins	List all the administrators.
PUT	/FHSOperator/FedAdmin/{member_id}	Update the information of a given administrator.
DELETE	/FHSOperator/FedAdmin/{member_id}	Erase the given administrator.
GET	/FHSOperator/FedInstances	List all the federations.
PUT	/FHSOperator/FedInstances/{fed_id}	Update the information of a given federation.
DELETE	/FHSOperator/FedInstances/{fed_id}	Erase the given federation.
POST	/FHSOperator/AllowConnection	Allow the connection from another federation service.
POST	/FHSOperator/Connect	Connect to another federation service.
GET	/FHSOperator/Connections	List all the connections.
DELETE	/FHSOperator/Connection/{conn_id}	Erase a connection.

The second API, named Fed Hosting Server-Fed Hosting Server (FHS-FHS), allows the connection to the federation services, the management of members in a federation instance, the transmission of monitoring data, the monitoring and the management of a federation, the listing of available services in a federation and the management of the services. Table I.2 provides a summary of the different possible operations available through the FHS-FHS API.

Table I.2 – FHS-FHS API

HTTP request method	Endpoint	Description
POST	/Connect	Connect to another federation service.
DELETE	/Connect/{connection_id}	Disconnect from another federation service.
POST	/JoinFederation	Join a federation.
PUT	/UpdateFederation	Update the information of a federation.
GET	/ValidateMember	Validate a member from another federation service.
DELETE	/LeaveFederation/{fed_id}	Quit a federation.
POST	/MonitoringData/{fed_id}	Transmit monitoring data to another federation.
PUT	/MonitoringParams/{fed_id}	Set the monitoring parameters for the given federation.
GET	/MonitoringParams/{fed_id}	Get the current monitoring parameters for the given federation.

Table I.2 – FHS-FHS API

HTTP request method	Endpoint	Description
PUT	/MonitoringProxy/{fed_id}	Call the external monitoring system for the given federation.
GET	/Federation/Query	Get the list of federations which can be joined.
POST	/Federation/Join/{fed_id}	Request to join a federation.
GET	/Federation/JoinRequests	Get the list of all requests to join a federation.
POST	/Federation/JointGrant/{request_id}	Accept the request to join a federation.
POST	/Federation/JointDeny/{request_id}	Refuse the request to join a federation.
DELETE	/Federation/Leave/{fed_id}	Quit the given federation.
POST	/MemberLogin	Log into a federation.
DELETE	/MemberLogout/{login_session_id}	Log out from a federation.
GET	/Discovery/{fed_id}/{member_id}	Get the list of all services available in the given federation.
POST	/Federation	Create a federation.
GET	/Federation	List all the federations.
GET	/Federation/{fed_id}	Get the information of a given federation.
DELETE	/Federation/{fed_id}	Erase a federation.
POST	/Attribute/{fed_id}	Create an attribute in the given federation.
GET	/Attribute/{fed_id}	Get all the attributes of the given federation.
DELETE	/Attribute/{fed_id}/{attr_id}	Erase the given attribute from the given federation.
POST	/Membership/{fed_id}	Grant membership to the given federation.
GET	/Membership/{fed_id}	Get all the members of the given federation.
GET	/Membership/{fed_id}/{member_id}	Get the information of the given member of the specific federation.
DELETE	/Membership/{fed_id}/{member_id}	Erase the membership from the given federation.
PUT	/Authorization/{fed_id}/{member_id}/{attr_id}	Give the authorization on the given attribute to the specific member of the federation.

Table I.2 – FHS-FHS API

HTTP request method	Endpoint	Description
DELETE	/Authorization/{fed_id}/{member_id}/{attr_id}	Revoke the authorization on the given attribute from the specific member of the federation.
POST	/Services/{fed_id}	Register a new service in the given federation.
GET	/Services/{fed_id}/{svc_owner_id}	Get the list of all the services in the given federation, which are linked to a specific service owner.
GET	/Services/{fed_id}/{svc_owner_id}/{svc_id}	Get the information of a given service of a specific federation.
PUT	/Services/{fed_id}/{svc_owner_id}/{svc_id}	Update the information of a given service included into a specific federation.
DELETE	/Services/{fed_id}/{svc_owner_id}/{svc_id}	Remove the given service from a specific federation.
OPTIONS	/Invocation/{fed_id}/{svc_id}	Invoke the given service available in a specific federation.
HEAD	/Invocation/{fed_id}/{svc_id}	Invoke the given service available in a specific federation.
GET	/Invocation/{fed_id}/{svc_id}	Invoke the given service available in a specific federation.
POST	/Invocation/{fed_id}/{svc_id}	Invoke the given service available in a specific federation.
PUT	/Invocation/{fed_id}/{svc_id}	Invoke the given service available in a specific federation.
PATCH	/Invocation/{fed_id}/{svc_id}	Invoke the given service available in a specific federation.
DELETE	/Invocation/{fed_id}/{svc_id}	Invoke the given service available in a specific federation.

I.5 Comparison between [ITU-T Q.4068] and [IEEE 2302] APIs

Table I.3 compares the APIs defined in [ITU-T Q.4068] and [IEEE 2302].

Table I.3 – Mapping between APIs

API from [ITU-T Q.4068]	API from [IEEE 2302]
APII (or GUI_l)	FHS operator API with the endpoints corresponding to /FHSOperator/*
APIm (or GUI_m)	FHS-FHS API
APIr	Partially covered by FHS-FHS API. The creation, the compilation, the execution and the recording of tests are missing in the FHS-FHS API. A possible solution is to include these missing actions into a service or several services available in the testbeds federation through the FHS-FHS API.
APIy (or GUI_y)	FHS Operator API with the endpoints corresponding to /FHSOperator/*
APIz	FHS-FHS API

Bibliography

- [b-ITU-T QSTR.FTT] ITU-T Technical Report QSTR.FTT, *Federated testbeds taxonomy*.
- [b-ETSI TS 102 812] ETSI TS 102 812 V1.2.1 (2003), *Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1*.
- [b-ISO 3534-3] ISO 3534-3:2013, *Statistics – Vocabulary and symbols – Part 3: Design of experiments*. <https://www.iso.org/standard/44245.html>
- [b-TBFxG-I-028R1] IEEE, *Presentation that provides Information on IEEE Std 2302-2021 that could be considered and referenced for potential use in Testbed Federation APIs*.
<https://www.itu.int/en/ITU-T/focusgroups/tbfxg/Documents/Deliverables/FG-TBFxG-TS-D3.1.docx>
- [b-TMF-business-API] FIWARE TM Forum Business API Ecosystem.
<https://fiwaretmfbizecosystem.docs.apiary.io/#>
- [b-TMF-OSS-BSS] TM Forum OSS/BSS. <https://www.tmforum.org/resources/toolkit/agile-ossbss-toolkit/>

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