ITUPublications

International Telecommunication Union
Standardization Sector

ITU Focus Group Technical Specification

(04/2024)

Focus Group on Testbeds Federations for IMT-2020 and beyond

(FG-TBFxG)

FG-TBFxG-TS-D2.2

Testbed as a Service APIs descriptions and interoperability requirements

Technical Specification ITU FG-TBFxG-TS-D2.2

Testbed as a Service APIs descriptions and interoperability requirements

Summary

This Technical Specification is reporting the elaboration of Testbed as a Service 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. It 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 point, the Technical Specification elaborates the related terms and definitions, requirements, reference model with properties of relevance for TaaS, and interoperability requirements for virtualizing and delivering modular and scalable TaaS on top of existing and future testbed infrastructures, including federated ones. The 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.

The experience and results gained by international research projects in this domain, such as F-Interop [b-F-Interop], Fed4FIRE+ [b-Fed4FIRE+], PAWR [b-PAWR] and SLICES [b-SLICES] are exploited in the Technical Specification.

Keywords

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

Note

This is an informative ITU-T publication. Mandatory provisions, such as those found in ITU-T Recommendations, are outside the scope of this publication. This publication should only be referenced bibliographically in ITU-T Recommendations.

Change Log

This document contains Version 1.0 of the ITU-T FG-TBFxG D2.2 Technical Specification "Testbed as a Service APIs descriptions and interoperability requirements" approved at FG-TBFxG eighth meeting held in Sophia Antipolis, France from 10 to 12 April 2024.

Acknowledgement

This Technical Specification was prepared under the leadership of Dr.-Ing. Giulio Maggiore (Telecom Italia, Italy) and Dr. Sébastien Ziegler (Mandat International, Switzerland), who served as the FG-TBFxG chair and FG-TBFxG vice-chair.

It is based on the contributions of various authors who participated in the Focus Group activities. FG-TBFxG appreciates Dr. Martial Michel (Data Machines Corp., US), Dr.-Ing. Ranganai Chaparadza (Capgemini Engineering, Germany), Dr.-Ing. Tayeb Ben Meriem (IPv6 Forum, France) and Dr. Robert Bohn (NIST, US) for presenting IEEE P2302-2021 Project and for their inputs to this Technical Specification.

Ing. Cédric Crettaz and Dr. Sébastien Ziegler (Mandat International, Switzerland) served as the main Editors of this Technical Specification.

Mr Denis Andreev (FG TBFxG Advisor) and Ms Emmanuelle Labare (FG-TBFxG Assistant) served as the FG-TBFxG Secretariat.

Editor:	Dr. Sébastien Ziegler (Mandat International, Switzerland)	Email: sziegler@mandint.org
Editor:	Ing. Cédric Crettaz (Mandat International, Switzerland)	Email: ccrettaz@mandint.org

Table of Contents

Technical Specification ITU FG-TBFxG-TS-D2.2

Testbed as a Service APIs descriptions and interoperability requirements

1 Scope

This Technical Specification describes the Testbed as a Service APIs and interoperability requirements. The APIs specified in this document are dedicated exclusively to TaaS. Integration, interoperability and extensibility of the TaaS are also studied in this Technical Specification.

2 References

[ITU-T Q.4068] Recommendation ITU-T Q.4068 (2021), *Open application program interfaces (APIs) for interoperable testbed federations.*

[b-ITU-T D2.1 FG-TBFxG] FG-TBFxG Technical Specification D2.1 (2025), *User requirements and reference model for Testbed as a Service*.

[IEEE 2302-2021] IEEE 2302-2021, *IEEE Standard for Intercloud Interoperability and Federation (SIIF)*.

3 Definitions

3.1 Terms defined elsewhere

This Technical Specification 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-ITU-R BT.1699]: A network data object or a service which is uniquely identified in a network. 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 [b-ITU-T D0.1 FG-TBFxG]: Service hosted on cloud providing access to distributed testbeds.

3.2 Terms defined in this Technical Specification

None.

4 Abbreviations and acronyms

This Technical Specification uses the following abbreviations and acronyms:

API Application Programming 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 Technical Specification is dedicated to all the TaaS APIs which are based on the generic reference model defined in [ITU-T Q.4068] and [b-ITU-T D2.1 FG-TBFxG]. These APIs are taking care of all the aspects to the monetization of the TaaS. Through the APIs specified in this Technical Specification, it is possible for the users, for example, to reserve a testbed slice, to reuse an experiment based on a template of a testbed slice. Furthermore, if the user doesn't find a service through the TaaS APIs, a request to a specific endpoint of the TaaS APIs is made by the user to execute the service on demand.

According to Figure 1 and Table 1 of [ITU-T Q.4068], the APIs needed for the TaaS are listed in the table below:

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.
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.

Table 1 – Testbed as a Service APIs

#	API name (identification tags)	API description
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 behavior 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 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_l)	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

Table 1 – Testbed as a Service APIs

#	API name (identification tags)	API description
		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). 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

Table 1 – Testbed as a Service APIs

#	API name (identification tags)	API description
		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 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 is using 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 is using 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

Table 1 – Testbed as a Service APIs

#	API name (identification tags)	API description
		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 suite/cases designer and test executer 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 executer) 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. And 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:

- 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.
- getResults: 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 APId

The APId 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 APId are the following ones:

- 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 APIe

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

The calls used by the APIe are:

- 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 APIf

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

The calls dedicated to the APIf 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 APIg

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

The calls for the APIg are composed by:

- 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 APIh

The APIh 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 ones:

- getResourceCapabilities: An HTTPS GET message for retrieving all the capabilities of a given resource.
- getResoureState: 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:

- 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_l

The APIl, also named GUI_l, 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 APIm/GUI_m

The APIm, 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 described below:

- 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 APIn

The APIn is used by the testbed resource broker to register itself into the testbed management system.

The calls of the APIn are the following ones:

- 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 APIo

The APIo is used by a test manager to register itself into the testbed management system.

The calls of the APIo are the following ones:

- 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 composed by:

- 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 executers 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 executers can run the different kinds of tests through the APIr. Stored test cases can be retrieved to be executed by the test executers.

The API calls are presented below:

- 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 ones:

- 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 APIt

The APIt provides the test results to a test manager.

The calls of the APIt are:

- 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 APIu

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

The calls of the APIu are:

- 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 APIv

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

The calls of the APIv 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 APIw

The APIw 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 APIw are listed below:

- 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:

- 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 consists 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 available below:

- 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 described as follows:

- 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

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 SLA (Service-Level Agreement) elements. It also provides an accountability service. This API permits the management of all the assets provided through the Testbed as a Service. 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 to the monetization of the services provided by the Testbed as a Service.

I.2 BSS/OSS APIs

The Business Support Systems (BSS) and the Operations Support Systems (OSS) allow the telecommunications network operators to activate and configure the 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 Testbed as a Service 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 Testbed as a Service 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 2302-2021

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

The first API is the FHS Operator API which permits to manage a Fed Hosting Server (FHS) and the communication between two Fed Hosting Servers. The following table 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. The following table 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.

Table I.2 – FHS-FHS API

HTTP request method	Endpoint	Description
GET	/MonitoringParams/{fed_id}	Get the current monitoring parameters for the given federation.
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.
DELETE	/Authorization/{fed_id}/{member_id}/{attr_id}	Revoke the authorization on the given attribute from the specific member of the federation.

Table I.2 – FHS-FHS API

HTTP request method	Endpoint	Description
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 Recommendation ITU-T Q.4068 and IEEE 2302-2021 APIs

The following table compared the APIs defined in [ITU-T Q.4068] and in [IEEE 2302-2021]:

Table I.3 – Mapping between APIs

API from [ITU-T Q.4068]	API from [IEEE 2302-2021]
APIl (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 D0.1 FG-TBFxG] FG-TBFxG Technical Specification D0.1 (2025), *Federated testbeds taxonomy*.

[b-ITU-T D2.1 FG-TBFxG] FG-TBFxG Technical Specification D2.1 (2025), *User requirements and reference model for Testbed as a Service*.

[b-ITU-R BT.1699] Recommendation ITU-R BT.1699 (2005), *Harmonization of declarative application formats for interactive TV*. https://www.itu.int/rec/R-REC-BT.1699/en

[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*.

[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-F-Interop] F-Interop project. https://cordis.europa.eu/project/id/687884

[b-Fed4FIRE+] Fed4FIRE+ project. https://www.fed4fire.eu/

[b-PAWR] PAWR project. https://advancedwireless.org/

[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/

[b-SLICES] SLICES project. https://slices-ri.eu/