

Microservice Bus API Documentation

- Discover the Microservice Bus
- API Definition and Swagger-UI
 - Swagger
- Register service to the Microservice Bus
- Unregister service from the Microservice Bus
- Query service from the Microservice Bus
- MSB Client SDK

Discover the Microservice Bus

Microservice Bus provides a service registration/discovery and routing mechanism to simplify the communications between services, The service consumers only need to talk with Microservice Bus without any address information of individual service providers. However, all the service consumers /providers must know the IP and service port of Microservice Bus. MSB suggests the following approach to discover the Microservice Bus. Each service provider/consumer implementation can get the address of the Microservice Bus from a local config file via the key 'msb.address', the default value consists of a hostname and a service port. The hostname is 'msb.onap.org', which can be resolved by a DNS server in the local network, and the default port is 80. If a DNS server is not available in some deployments, an IP should be used instead.

Examples:

- Default config when a DNS server is available

msb.address: msb.onap.org:80

- An IP if a DNS server is not available

msb.address: 10.74.205.123:80

API Definition and Swagger-UI

ONAP intends to be a microservices-based architecture and every individual service is exposed as a restful API. Microservice Bus provide mechanisms to collect and display the API description(swagger UI) and metrics of services centrally. The services should provide the information to Microservice Bus by the below approaches:

Swagger

1. REST APIs should be described in a swagger.json file according to [Open API Initiative](#).
2. The swagger.json file should be put at the root path of the service url, such as: http(s)://[hostname][:port]/[ServiceType]/[ServiceName]/[Service Version]/Swagger.json
3. The swagger.json can be automatically generated by JAVA notation, more information can be find at [Swagger Annotations](#).

For example, the swagger description and UI of the services provided by Microservice Bus:

Service: /api/microservices/v1/

Swagger: /api/microservices/v1/swagger.json

[blocked URL](#)

Register service to the Microservice Bus

Operation	Register service to the Microservice Bus
URL	/api/microservices/v1/services
Verb	POST

Request							
	Parameter	Mandatory	Parameter type	Data Type	Default	example	Description
	Body	Y	Body	JSON String		<pre>{ "serviceName": "aa1", "version": "v8", "url": "/aa1/v8", "protocol": "REST", "visualRange": "1", "path": "/aa1/v8" "nodes": [{ "ip": "10.74.56.36", "port": "8988", "ttl": 0 }] }</pre>	Described in the below table
	createOrUpdate	N	Query	boolean	true		true: create new instances or replace the old instances if the instance with the same service name, ipandportexist false: create new instances and remove all the old instances with the same service name
	Parameter	Mandatory	Data Type	Default	Description		
	serviceName	Y	String		Service Name, must comply with RFC 1123 , only allowsthe ASCII letters 'a' through 'z', the digits '0' through '9', and the minus sign ('-').		
	version	Y	String		Service Version		
	url	Y if protocol is 'REST' or 'UI' or 'HTTP'	String		the actual URL of the service to be registered		
	protocol	Y	String		supported protocols: 'REST', 'UI', 'HTTP','TCP'		
	visualRange	N	String	1	Visibility of the service. External(can be accessed by external systems):0 Internal(can only be accessed by ONAP microservices):1		
path	N	String		Thecustomizedpublish path of this service. If path parameter is specified when registering the service, the service will be published to apigateway under this path. Otherwise, the service will be published to apigateway using a fixed format: api/{serviceName}{version}. The customized publish path should only be used for back-compatible.			
enable_ssl	N	boolean	False	True if the registered service is based on https. False if the registered service is based on http.			
nodes	Y	List		ip: the ipoftheservice instance node, it could also be a hostname like catalog.onap.org port: the port of the service instance node ttl: time to live, this parameter is reserved for later use			
Response	example	Description					
	<pre>{ "serviceName": "catalog", "version": "v1", "url": "/api/catalog/v1", "protocol": "REST", "visualRange": "1", "lb_policy": "ip_hash", "nodes": [{ "ip": "10.74.55.66", "port": "6666", "ttl": 0 }, { "ip": "10.74.56.37", "port": "8989", "ttl": 0 }, { "ip": "10.74.56.36", "port": "8988", "ttl": 0 }] }</pre>	serviceName: service name version: version url: url of the created service instance protocol: supported protocols: 'REST', 'UI', 'HTTP','TCP' nodes: the service instance nodes list to register lb_policy: Load balancing method, Currently two LB methods are supported, round-robin and ip-hash. ip: the ip of the service instance node, it could also be a hostname like catalog.onap.org port: the port of the service instance node ttl: time to live, this parameter is reserved for later use					
Success Code	201						
Error Code	422 Invalid Parameters 500 Internal Server Error						

Unregister service from the Microservice Bus

Operation	Unregister service from the Microservice Bus
URL	/api/microservices/v1/services/{serviceName}/version/{version}/nodes/{ip}/{port}
Verb	DELETE

Request	Parameter	Mandatory	Parameter type	Data Type	Default	example	Description
	serviceName	Y	Path	String			Service Name
	version	N	Path	String			Service Version
	ip	N	Path	String			the IP address of the service instance, it could also be a hostname like catalog.onap.org
	port	N	Path	String			the port of the service instance
Response							
Success Code 204							
Error Code 404 Can't find the service instance 422 Invalid Parameters 500 Internal Server Error							

Query service from the Microservice Bus

You can access service by two ways:

First is Point to Point, service consumer can query service instances address from MSB by the following API, and access the service provider directly; - Not recommended, we will lose centralized control capabilities if chose this approach, for example, service call logging, service load balancing policy, etc.

Second is API Gateway, service consumer do not care ip and port of the service provider, it just access the API-Gateway, and the API-Gateway will route the service call to the appropriate service provider; in this approach, service consumers only need to know the ip and port of API Gateway, it can be stored in a configuration file or an environment property of the service consumer implementation.

MSB support the following access methods:

if the service provider version property is provided when registration, you can use the version before or behind the service name.

/api/[servicename]/v1

/api/v1/[servicename]

if the service did not have version property, you can use the url without version.

/api/[servicename]/

Operation	Query service from the Microservice Bus						
URL	/api/microservices/v1/services/{serviceName}/version/{version}						
Verb	GET						
Request	Parameter	Mandatory	Parameter type	Data Type	Default	example	Description
	serviceName	N	Path	String			Service Name
	version	N	Path	String			Service Version
Response	example		Description				
	<pre>{ "serviceName": "catalog", "version": "v1", "url": "/api/catalog/v1", "protocol": "REST", "visualRange": "1", "nodes": [{ "ip": "10.74.55.66", "port": "6666", "ttl": 0 }, { "ip": "10.74.56.37", "port": "8989", "ttl": 0 }, { "ip": "10.74.56.36", "port": "8988", "ttl": 0 }] }</pre>		serviceName: service name version: version url: url of the created service instance protocol: supported protocols: 'REST', 'UI', 'HTTP', 'TCP' nodes: the service instance nodes list to register ip: the ip of the service instance node, it could also be a hostname like catalog.onap.org port: the port of the service instance node ttl: time to live, this parameter is reserved for later use				
Success Code	200						
Error Code	404 Can't find the service instance 422 Invalid Parameters 500 Internal Server Error						

MSB Client SDK

MSBServiceClient : JAVA SDK for MSB service discovery

RestServiceCreator: JAVA SDK for REST service invocation

MSB Client SDK Example

<https://gerrit.onap.org/r/gitweb?p=msb/java-sdk.git;a=tree;f=example;h=1c331f86cbcbdb8cc2935d8ac41169da1a523ec5;hb=refs/heads/master>

```
String MSB_IP="127.0.0.1";
int MSB_Port=10081;

MSBServiceClient msbClient = new MSBServiceClient(MSB_IP, MSB_Port);

RestServiceCreator restServiceCreator =
    new RestServiceCreator(msbClient);

AnimalServiceClient implProxy =
    restServiceCreator.createService(AnimalServiceClient.class);

Animal animal = implProxy.queryAnimal("panda").execute().body();
System.out.println("animal:" + animal);
```

Note: In order to test the example application, you need to run MSB following this guide [MSB Test Environment Setup](#)