SWIM Discovery Service (SDS) Implementation Specification, Version 1.0.0

U.S. Federal Aviation Administration (FAA) System Wide Information Management (SWIM)

July 2020

Foreword

Service discovery is a critical success factor for the overall usability of a service. If a potential service consumer is unable to find a service, then a shared information environment like SWIM fails to achieve its purpose. At its most general level, service discovery is a process of locating information about services that may meet service consumer business needs.

This specification describes the enabling technologies and practices that support federated service discovery among independently developed and autonomously managed discovery mechanisms.
It will incrementally evolve and mature to maintain timely and actionable guidance in addressing emerging business requirements.

Table of Contents

Table of Figures

1. Introduction

1.1. Scope

This document specifies a standard interface, key interaction patterns, schemas, and message exchange formats required to develop System Wide Information Management (SWIM) Discovery Services (SDS).

1.2. Purpose

The purpose of this specification is to establish guidelines and general technical principles for the development of SWIM Discovery Services (SDS). It presents the enabling technologies and practices that support federated service discovery among independently developed and autonomously managed SDS implementations.

This specification adheres to the following service design principles by requiring SDS to:

- conform to SOA architectural principles,
- be based on standard internet architecture and standards,
- use a formal standard language for information exchange,
- be composable but not coupled,
- be self-describing and self-advertising.

This document is designed for use by:

- Software developers and architects implementing SDS,
- Software developers and architects implementing agents and applications for SDS,
- Standards architects and analysts developing specifications for finding or exchanging information about SWIM services (service metadata).

1.3. References

[1] RFC 7231, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content; IETF; R. Fielding et al; June 2014
https://tools.ietf.org/html/rfc7231

[2] Architectural Styles and the Design of Network-based Software Architectures, Dissertation; Roy Thomas Fielding, 2000
https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[3] NIST SP 800-95, Open Grid Services Architecture Glossary of Terms; 25 January 2005
https://csrc.nist.gov/glossary/term/Service_Oriented-Architecture

[4] SWIM Controlled Vocabulary V. 1.0.0; FAA SWIM; 2019-03-25
https://semantics.aero/pages/swim-vocabulary.html

[5] OpenAPI Specification, Version 3.0.3; Swagger;
https://swagger.io/specification/

[6] Web Services Glossary; W3C Working Group Note; 11 February 2004
https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

[7] Concept of Operations for the SWIM Inter-Registry Framework (SIRF), Version 1.0.0; U.S. FAA SWIM, October 11, 2018
https://www.faa.gov/air_traffic/technology/swim/governance/international_collaboration/media/conops-sirf-1.0.0.pdf

[8] Service Description Conceptual Model (SDCM) 2.0, FAA/SESAR, June 3, 2016
http://swim.aero/sdcm/2.0.0/sdcm-2.0.0.html

[9] Web Services Architecture; W3C Working Group Note 11 February 2004
https://www.w3.org/TR/ws-arch/

[10] RFC 2119, Key words for Use in RFCs to Indicate Requirement Levels, Network Working Group, March 1997
http://www.rfc-editor.org/rfc/rfc2119.txt

[11] Guidance for Creating SWIM Service Identifiers; ICAO APAC SWIM-TF3; April 28, 2019
https://www.icao.int/APAC/Meetings/2019SWIMTF3/WP03_USA%20AI3d%20-%20Task%201-4_SWIM%20Service%20Identifier.pdf

[12] US Federal Chief Information Officers Council, “The HTTPS-Only Standard“
https://https.cio.gov/

[13] Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry, 2014-02-17
https://www.iana.org/assignments/http-authschemes/http-authschemes.xhtml

[14] The OAuth 2.0 Authorization Framework, RFC6749; IETF; October 2012
https://tools.ietf.org/html/rfc6749

1.4. Terms and Definitions

For the purpose of this specification, the following terms and definitions apply:

discovery service: A service that provides capabilities to a service consumer to obtain information about available services.

service registry: An application designed to support service discovery through a formal registration process for storing, cataloging, and managing metadata relevant to the service.

peer: A discovery service that interacts or may interact with other discovery services.

resource: An information object identified by a Uniform Resource Identifier (URI).

resource id: A URI by which the resource is uniquely referenced.

resource path: A relative path that represents a resource node within a hierarchical resource model.

Note: the *path* is appended to the service URL; no relative path resolution is assumed.

resource template: A *resource id* syntax that includes variables that must be substituted before the resource id’s resolution.

user: A person who deploys a *user agent* to initiate a request to and receives a response from a discovery service.

user agent: A software program, such as a browser, whose purpose is to mediate interactions with services on behalf of the user under the user's preferences.

1.4.1. Terminological Conventions

This specification discusses two kinds of services: discovery services and the services whose service descriptions are provided by discovery services, e.g., information services. To avoid ambiguity, the former will be referred to as *discovery services* and the latter as *services* throughout this document.

1.4.2. Key words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [14].

1.5. Overview

One of the chief architectural principles in realizing a SWIM paradigm is *service discovery*. Service discovery is the process of locating information about services that may meet service consumer business needs. This information, which normally includes functional and non-functional characteristics of a service, is conveyed through a uniformly interpretable set of service metadata and accessed through some form of retrieval mechanism by a user or by a software agent. Service discovery improves interoperability among services, encourages reuse of services, and lessens unnecessary costs of redundant service development.

Service discovery can be accomplished in a variety of ways. The two solutions commonly deployed by SWIM implementations are: *service registries* (the most prevalent) and *discovery services* (the subject of this specification).

- A generic definition of a service registry is "an authoritative, centrally controlled store of information" [6], [3]. A service registry is commonly realized as a database or data repository that allows a user to catalog and manage metadata relevant to services [4]. Because the concept of run-time service discovery has not lived up to its original expectations, a registry is used for design-time discovery more often than not.
- A discovery service is a service that enables agents to retrieve service-related resource descriptions. By virtue of being a *service* – as this notion is commonly defined in SOA literature – a discovery service is inherently loosely-coupled, interoperable, autonomous, reusable, discoverable, and self-describing. And consistent with SOA technological perspectives, a discovery service is implemented as a software component that is identified by a URI and whose interface and binding are capable of being defined, described, and discovered as machine-readable artifacts.

In a SWIM environment, both these concepts are usually realized in a mutually complementary way. A service registry typically supports the entry and management of service metadata by human users, after which it can be further disseminated by a discovery service. (It could be said that a discovery service uses a registry as a data source.) The discovery service in turn can augment the information stored in its associated registry with the information made available by other discovery services. A service registry may also serve as a client for accessing a discovery service. Figure 1 depicts a possible scenario wherein two SWIM implementations provide a service registry to their respective users and also use a discovery service to exchange service metadata.

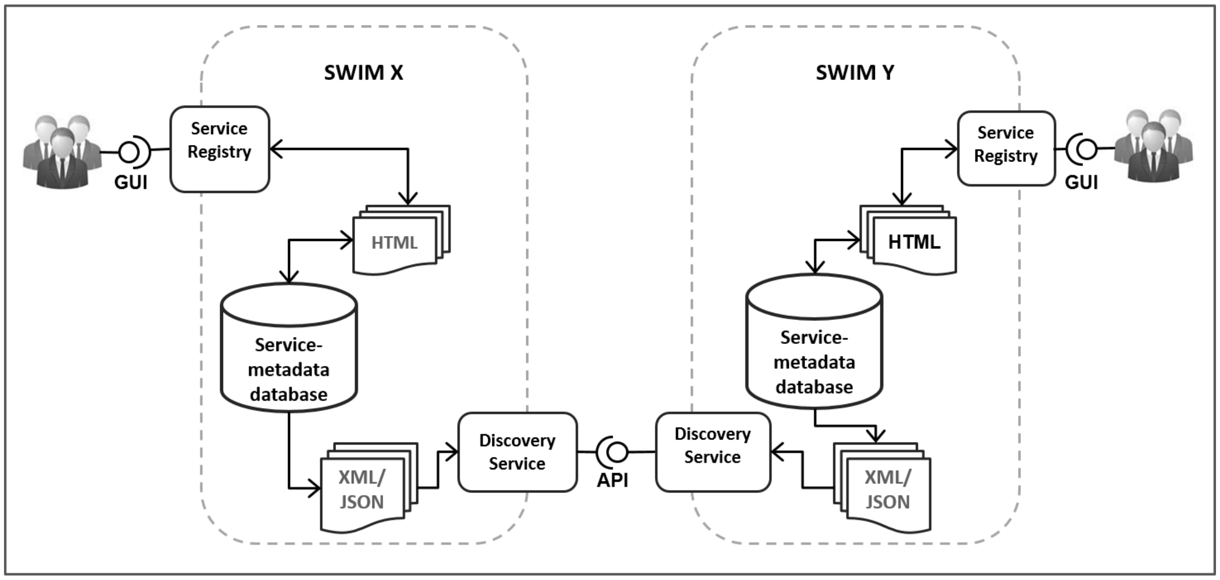

Figure 1 Example of SWIM environments providing service discovery capabilities

Although this specification focuses exclusively on the implementation of a discovery service, developers need to recognize potential coordination with SWIM service registries or other technology solutions supporting service discovery.

2. Architecture

The architecture of the SDS model draws upon well-established works in information technology. Among them are two architectural models that are foundational for the SDS concept: Representational State Transfer (REST) [2] and Peer-to-Peer (P2P) Discovery [9].

- REST defines an approach for implementing services (RESTful Web services) that provide and access a representation of information resources via a predefined, uniform set of operations. It is designed to deploy large-scale distributed hypermedia systems while leveraging existing ubiquitous standards (HTTP, XML, URI, MIME) with already pervasive necessary infrastructure [2].
- A P2P distributed application architecture defines a network where every node ("peer") is an equally privileged, equipotent participant with the same capabilities and responsibilities. In a service discovery realization of P2P, each peer is a discovery service that functions as both a "client" and a "server" to the other peers.

By uniting these two architectural paradigms, this document defines a vision of a network of discovery services collectively capable of supporting service discovery across diverse organizational domains. The envisioned assembly of RESTful Web services is innately scalable; it allows nodes to be added or removed without affecting the rest of the network. And the participating services do not need to rely on centralized registries; they describe themselves and provide references to other peers, and thus may be discovered dynamically.

2.1. Behavior Model

This section presents a collection of use cases, which together describe the discovery service behavior, i.e., how the service interacts with a user and other services. The SDS specification is designed to meet the use cases presented below.

Assumptions

a) A User (a person) always initiates a discovery request and is the ultimate recipient of the result. (The User may represent an organization, but this fact is irrelevant in the context of this model.)
b) The User has access to an affiliated discovery service (DS "X" in this model) and has the necessary security credentials to perform all operations offered by the service's interface.
c) The User always utilizes X to send a request to other discovery services and consolidate their responses.
d) All discovery services are assumed to be compliant with this specification; however, the extent of conformance and access control policy may vary.

2.1.1. USE CASE 01. Obtaining information about a discovery service (single Peer)

Precondition

The User is aware of the existence (i.e., knows the network address) of a discovery service Y.

Scenario

Before starting a federated search, the User needs to obtain information about the targeted discovery service "Y" (see Assumption d.). The User sends a request to Y. Y responds with self-describing information, which includes a list of the operations offered by Y via the API.

Effect

The User has all necessary information about Y (including security constraints for offered operations).

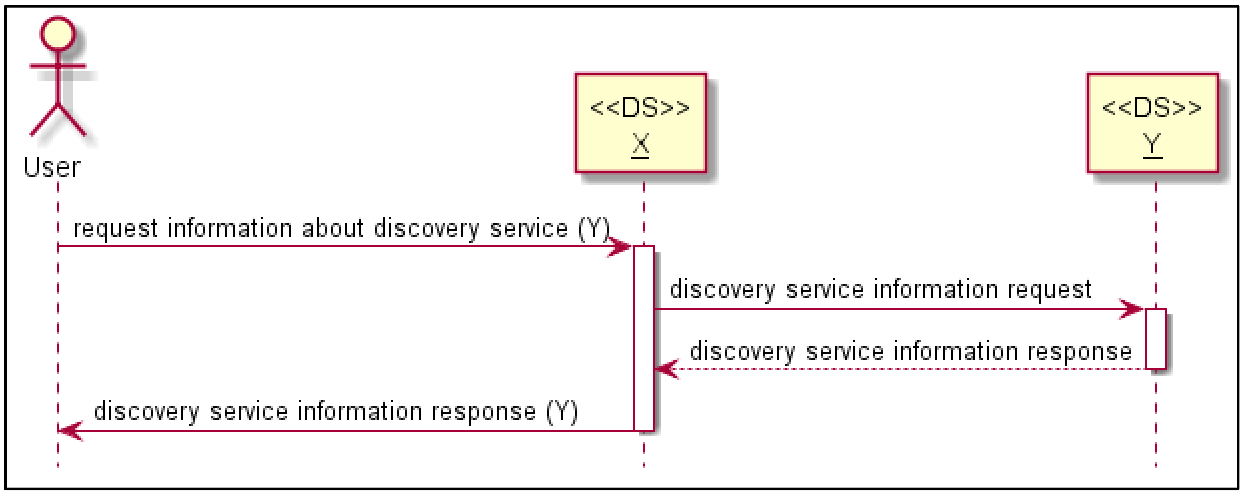
User
<<DS>>
X
<<DS>>
Y
request information about discovery service (Y)
discovery service information request
discovery service information response
discovery service information response (Y)

Figure 2 Use Case 01. Obtaining information about a discovery service

2.1.2. *USE CASE 02. Obtaining a list of services (single Peer)*

Precondition

The User is aware of the existence (i.e., knows the network address) of discovery service Y.
After conducting UC01, the User knows that Y is capable of supporting service discovery operations.

Scenario

The User wishes to obtain an indexed list of all services provided by both X and Y. Note: the User may also want to narrow the search by applying a set of criteria, such as type of service, availability status, etc. The User deploys X to create a request for the consolidated list of services. Upon receiving the request, X generates a list of services and passes the request to Y. Y generates a list of its services, and returns it to X. X aggregates both lists and returns the consolidated result to the User.

Effect

The User has references to – and subsequently may retrieve – descriptions of all services provided by both X and Y.

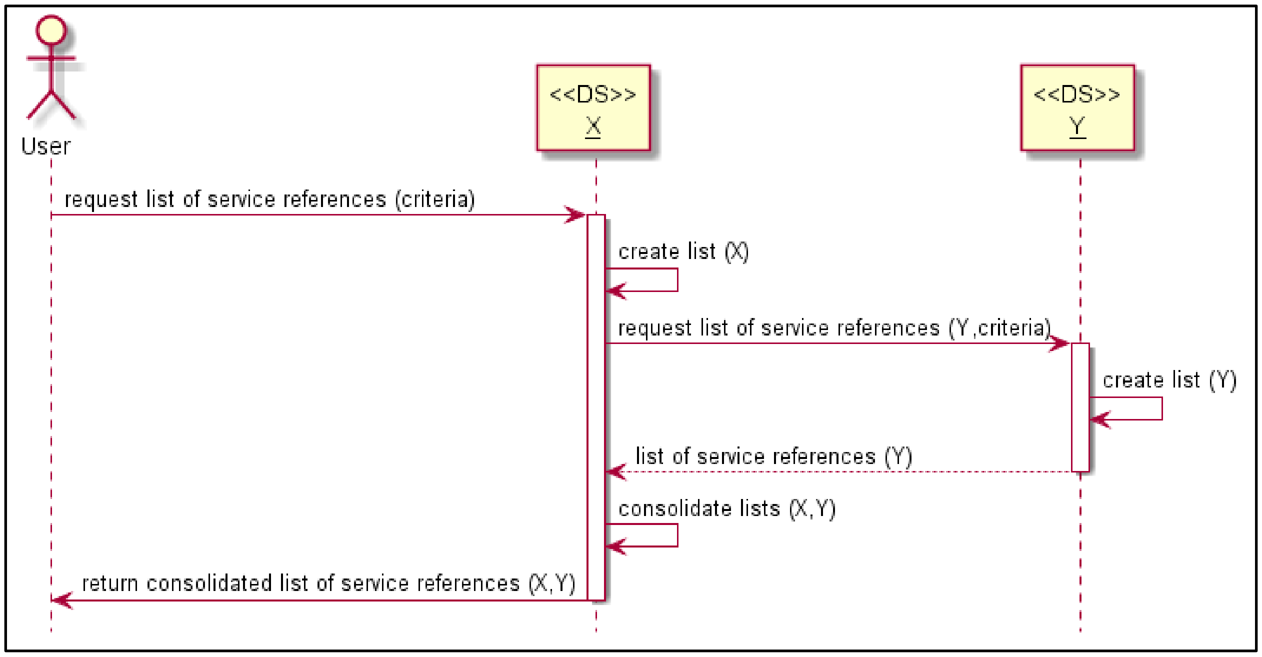
User
<<DS>>
X
<<DS>>
Y
request list of service references (criteria)
create list (X)
request list of service references (Y,criteria)
create list (Y)
list of service references (Y)
consolidate lists (X,Y)
return consolidated list of service references (X,Y)

Figure 3 USE CASE 02. Obtaining list of services

2.1.3. *USE CASE 03. Obtaining a description of a service (single Peer)*

Precondition

The User has a list of references to descriptions of all services provided by both X and Y.

Scenario

The User wishes to obtain a detailed description for one or more services. Using the list of references received as an outcome of UC02, the User sends the request to either X or Y to retrieve the desired descriptions.

Effect

The User has obtained descriptions of the services of interest.

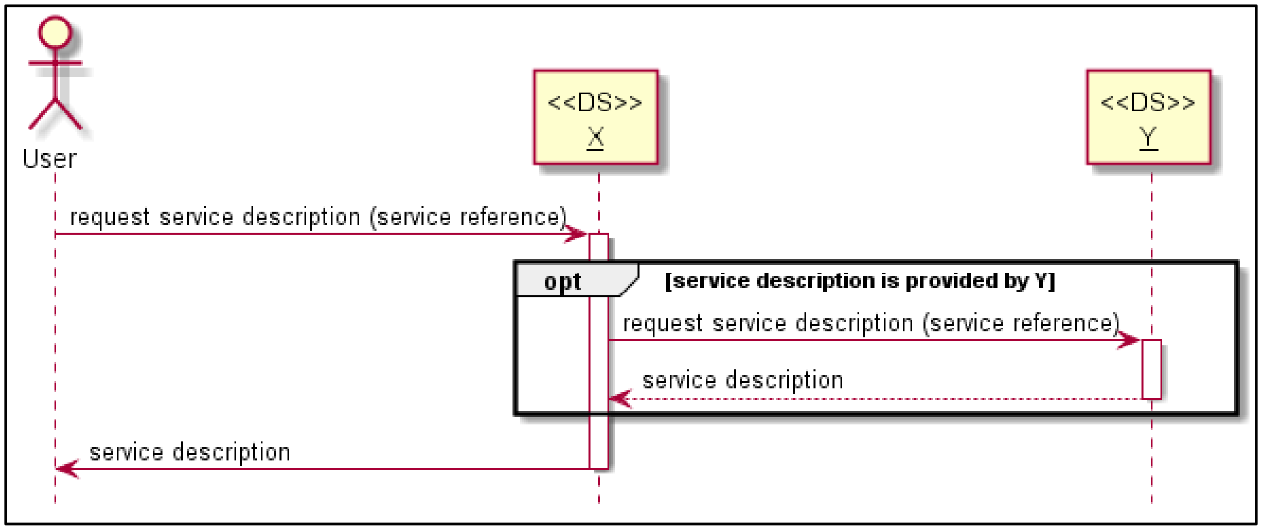
User
<<DS>>
X
<<DS>>
Y
request service description (service reference)
opt
[service description is provided by Y]
request service description (service reference)
service description
service description

Figure 4 USE CASE 03. Obtaining a description of a service

2.1.4. USE CASE 04: Discovering previously unknown peers (multiple peers)

Precondition

The User has access to X and is also aware of Y and its capabilities (the result of enacting UC01). Y is aware of the address of the discovery service Z not known to X.

Scenario

The User wishes to extend service discovery beyond the information provided by X and Y. The User sends an inquiry to Y about other discovery services that may be known to Y. Y responds with a list of zero or more addresses of discovery services, including a discovery service Z. The User then enacts UC01, UC02, and UC03 to obtain service information as provided by Z.
The User may then send a request to Z about the peers "known" to Z. The User then recursively continues to discover more peers capable of supporting the service discovery and includes them in a federated service discovery.

Effect

The User has obtained the addresses of previously unknown discovery services, which may also be deployed for federated service discovery.

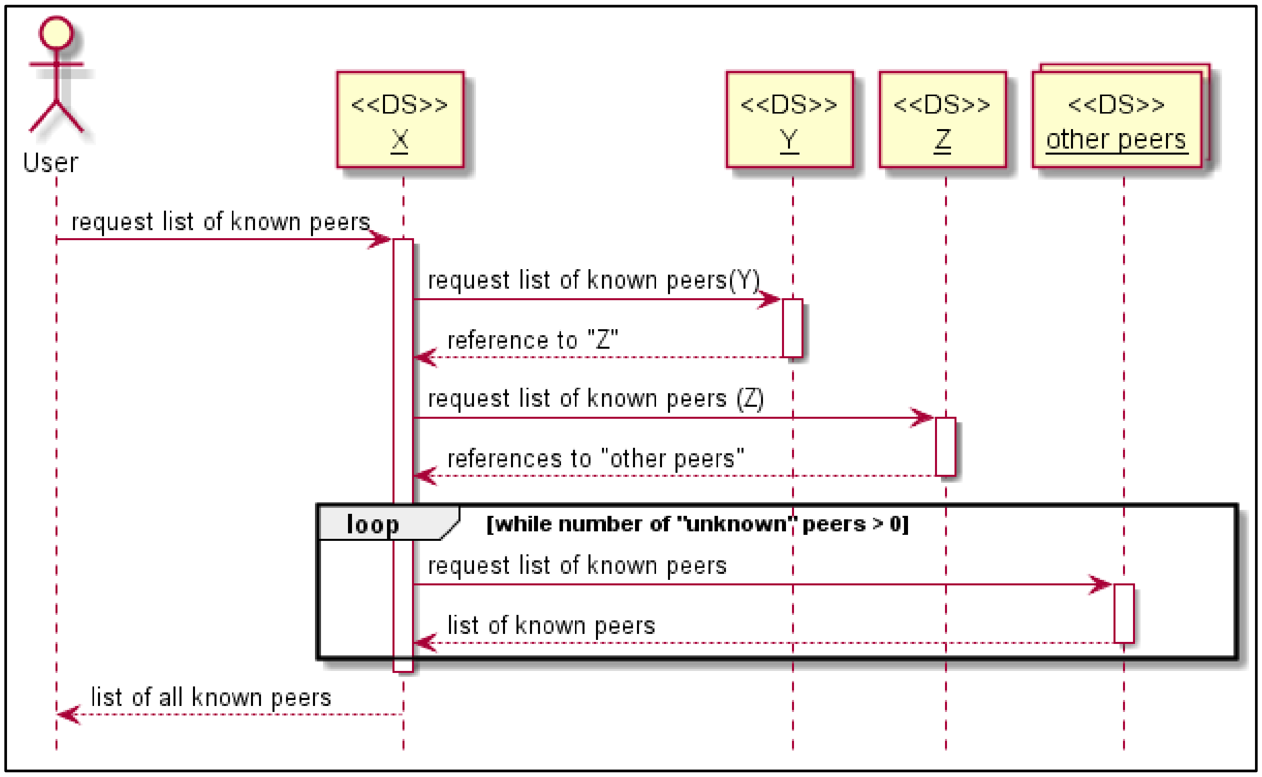
User
<<DS>> X
<<DS>> Y
<<DS>> Z
<<DS>> other peers
request list of known peers
request list of known peers(Y)
reference to "Z"
request list of known peers (Z)
references to "other peers"
loop
[while number of "unknown" peers > 0]
request list of known peers
list of known peers
list of all known peers

Figure 5 USE CASE 04: Discovering previously unknown Peers

2.2. Information Model

The Information Model presented in this section defines common structures for information exchanged among discovery services and/or a User. This information is divided into two major categories:

- Information that supports discovery services’ interactions, and which may include an identification of a discovery service, functionalities provided by the service, access policies, and references to other discovery services (peers),
- Information provided by a discovery service to support service discovery, such as lists of services or detailed description of these services.

2.2.1. Discovery Services Interaction Support Information

Interaction Support Information consists of information elements that allow a discovery service to describe itself to its peers in order to support possible invocations and interactions, thus adhering to the service design principle stated in section 1.2 of being self-describing and self-advertising.

2.2.1.1. Discovery Service Information

Discovery Service Information describes all defining characteristics, operations offered, and associated constraints of the SDS instance. Figure 6 depicts the structure of the Discovery Service Information concept.

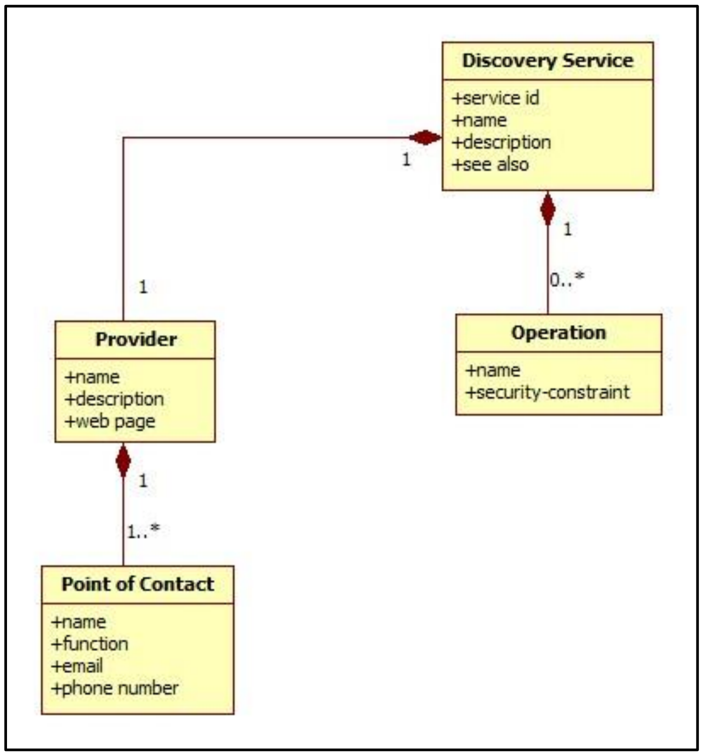

Figure 6 Discovery Service Information: logical model

Discovery Service Information includes:

- **a.** An identifier in the form of a URI by which the service is uniquely referenced and can be located
- **b.** The full name of the service and acronym by which service is commonly recognized
- **c.** The version of the service
- **d.** Information about the service provider, that is, the organization responsible for establishing and operating the instance of the discovery service
- **e.** The service provider organization information includes:
  - **1.** Provider organization name
  - **2.** Brief description of the organization
  - **3.** URI for the Web page that supplies information about the organization
- **f.** The service provider organization information also includes information about at least one point of contact (PoC), that is, a person or group within the provider organization, suitable for making a human contact for any purpose
- **g.** The PoC information includes:
  - **1.** The full name of the contact
  - **2.** The POC's job title or work functions
  - **3.** The POC's telephone number (optional)
  - **4.** The POC's e-mail address
- **h.** A reference to a location where additional information about the service can be found (e.g., an entry in a service registry, on-line documentation)

i. A list of operations offered by the service interface

j. Security constraints associated with each operation that users should comply with in order to invoke the operations

k. The security constraints information includes:

   1. The name of each security mechanism (e.g., “authentication”)

   2. A reference to a document (e.g. a protocol, a specification) that defines how the security mechanism is applied.

Figure 7 provides a formal representation of Discovery Service Information.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://sds.faa.gov/capability.json ",
  "title": "Discovery Service",
  "description": "Information that describes an instance of SWIM Discovery
Services (SDS).",
  "type": "object",
  "required": ["id","name"],
  "properties": {
    "id": {
      "description": "The identifier by which the service is uniquely
referenced.",
      "type": "string",
      "format": " uri-reference"
    },
    "name": {
      "description": "The full name and acronym of the service.",
      "type": "string"
    },
    "description": {
      "description": "The description of the service.",
      "type": "string"
    },
    "version": {
      "description": "The version of the service.",
      "type": "string"
    },
    "provider": {
      "minimum": 1,
      "maximum": 1,
      "$ref": "organization.json"
    },
    "operations": {
      "description": "A list of operations offered by the service.",
      "type": "array",
      "uniqueItems": true,
      "items": {
        "$ref": "#/definitions/operation"
      }
    },
    "see also": {
      "description": "A reference to an external resource (e.g., a registry)
for extended documentation. ",
      "type": "string",
      "format": "uri"
    }
  },
  "definitions": {
    "operation": {
      "description": "An operation offered by the service.",
      "type": "object",
      "properties": {
        "name": {
          "description": "The name of the operation.",
          "type": "string"
        },
        "security-constraint": {
          "type": "string"
        }
      }
    }
  }
}
```

Figure 7 Discovery Service information: schema

Note: The referenced class Organization is defined in Appendix A.

2.2.1.2. Discovery Service's Peers

Peers Information presents references to other discovery services (peers) that may be invoked to support federated searches. Figure 8 depicts the Peers information item.

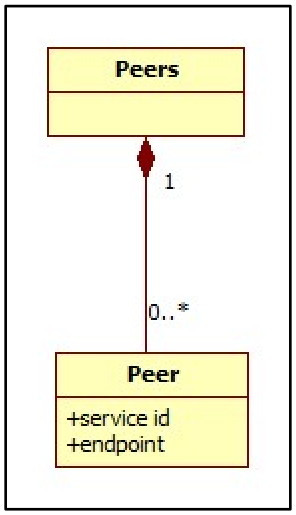

Figure 8 Peers' information: logical model

Peers Information for each peer includes:

a. The identifier by which the peer is uniquely referenced
b. The end point, that is, the network address, specified by a URI, by which the peer can be accessed.

Figure 9 provides a formal representation of Peers Information.

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   "$id": "http://sds.faa.gov/peers.json",
   "title": "Peers",
   "description": "Reference information about other discovery services,
which may or may not be known to an invoking agent.",
   "type": "object",
   "properties": {
         "peers": {
               "type": "array",
               "items": {"$ref": "#/definitions/peer"}
         }
   },
   "definitions": {
         "peer": {
               "type": "object",
               "properties": {
                     "service-id": {
                           "description": "Globally unique service
identifier.",
                           "type": "string",
                           "format": "uri"
                     },
                     "endpoint": {
                     "description": "A URI to the endpoint of the service
API,",
                     "type": "string",
                     "format":"uri"
                     }
               },
               "required":["service-id","endpoint"]
         }
   }
}
```

Figure 9 Peers' information: schema

2.2.2. Service Discovery Information

Service Discovery Information is the information provided by a discovery service in support of service discovery. This specification uses the Service Description Conceptual Model (SDCM) [8] to define a logical and subsequently a physical data model.

Service Discovery Information is represented as a combination of two elements:

a) Indexed list of references to all services made available by a discovery service,
b) Detailed description(s) of any service(s) presented in the list.

The former allows a user to obtain an indexed list of services (UC 02) and the latter allows the user to retrieve a description of the service (UC 03), that is, to discover the service.

2.2.2.1. Indexed List of Services

List of Services (hereinafter referred to simply as *Services*) Information presents a collection of references by which individual services can be identified and located. Information for each service in the list includes:

- **a.** The identifier by which the service is uniquely referenced
- **b.** The name of the service
- **c.** The version of the service
- **d.** A brief description of the service
- **e.** Zero or more values representing the category of the service
- **f.** Zero or one value representing the availability status of the service
- **g.** Zero or one value representing the interface type of the service

Figure 10 depicts the structure of the Services Information model.

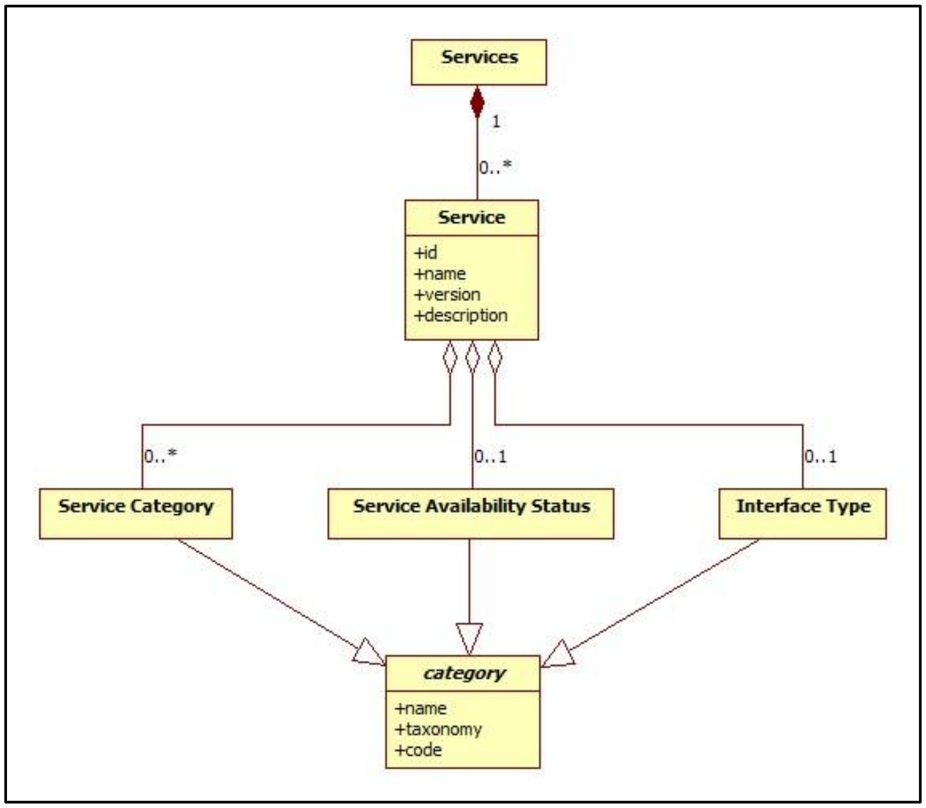

Figure 10 Services Information: logical model

Figure 11 provides a formal representation of Services Information.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://sds.faa.gov/services.json",
  "title": "Services",
  "description": "List of references to the services whose service
descriptions the discovery service provides.",
  "type": "object",
  "properties": {
    "services": {
      "type": "array",
      "uniqueItems": true,
      "items": {
        "$ref": "#/definitions/service"
      }
    }
  },
  "definitions": {
    "service": {
      "description": "Information comprising a single service reference.",
      "type": "object",
      "properties": {
        "id": {
          "description": "The identifier by which the service is uniquely
referenced.",
          "type": "string",
          "format": "uri"
        },
        "name": {
          "description": "The full name and acronym of the service.",
          "type": "string"
        },
        "description": {
          "description": "Brief description of the service.",
          "type": "string"
        },
        "service-category": {
          "type": "object",
          "properties": {
            "name": {
              "type": "string",
              "default": "SWIM Service Category"
            },
            "taxonomy": {
              "type": "string",
              "format": "uri",
              "default": "http://semantics.aero/service-category"
            },
            "code": {
              "type": "string",
              "format": "uri"
            }
          }
        },
        "service-availability-status": {
          "type": "object",
          "properties": {
            "name": {
              "type": "string",
              "default": "Service Availability Status"
            },
            "taxonomy": {
              "type": "string",
              "format": "uri",
              "default": "http://semantics.aero/availability-status"
            },
            "code": {
              "type": "string"
              "format": "uri",
            }
          }
        },
        "interface-type": {
          "type": "object",
          "properties": {
            "name": {
              "type": "string",
              "default": "Service Interface Type"
            },
            "taxonomy": {
              "type": "string",
              "default": "http://semantics.aero/interface-type"
            },
            "code": {
              "type": "string",
              "format": "uri"
            }
          }
        }
      },
      "required": ["id","name","description"]
    }
  }
}
```

Figure 11 Services information: schema

2.2.2.2. Service Information

Service Information is a description of a specific SWIM service. The type of SWIM service that is most commonly discovered is an "information service" (e.g., aeronautical, flight, or weather information services), although other types of services (e.g., mediation services, security services, as well as discovery services themselves) may also be discovered. A logical structure of Service Information is derived from the Service Description Conceptual Model (SDCM) [8] (available at http://swim.aero/sdcm/2.0.0). A representation of it in a formal language, *Service Description for JSON (SDM-J)*, is available at https://semantics.aero/service-description/sdm-j/sdm-j-1.0.0/. Figure 12 shows the root object in the collection of objects that together form a service description.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://sdm-j-1.0.0/service-description.schema.json",
  "title": "service description",
  "description": "This schema represents the top-level(root) object  of the model that
specifies a Service Description. It follows the SDCM v2.0
<http://swim.aero/sdcm/2.0.0> blueprint, in which the ServiceDescription class
aggregates three classes: Profile, Model and Grounding.",
  "type": "object",
  "properties": {
    "service-description": {
      "type": "object",
      "description": "The information needed in order to use, or consider using, a
service.",
      "properties": {
        "service-id": {
          "$ref": "#/definitions/service-id"
        },
        "profile": {
          "$ref": "https://semantics.aero/service-description/sdm-j/sdm-j-
1.0.0/profile.json#/profile"
        },
        "model": {
          "$ref": "https://semantics.aero/service-description/sdm-j/sdm-j-
1.0.0/model.json#/model"
        },
        "grounding": {
          "$ref": "https://semantics.aero/service-description/sdm-j/sdm-j-
1.0.0/grounding.json#/grounding"
        }
      },
      "required": [
        "service-id",
        "profile"
      ]
    }
  },
  "definitions": {
    "service-id": {
      "description": "The identifier by which the service is globally uniquely
referenced.",
      "type": "string",
      "format": "uri-reference"
    }
  }
}
```

FIGURE 12 Service information top-level schema

2.3. Resource Model

The SDS API follows the REST architecture and therefore defines a collection of interlinked resources. This set of resources in the REST API is referred to as a *resource model*. Architecturally, each resource is uniquely identified within the API by a combination of resource path and a standard HTTP Method that is used to access and/or manipulate the resource. Because this version of SDS only supports the return of information represented by a resource, only the GET HTTP method is used for all resources.

A definition for each resource may also include optional *parameters*. A parameter is an argument included in a request to alter the response. There are four types of parameters commonly included in a REST API: header parameters, path parameters, query string parameters, and request body parameters.

Each resource in this model has one or more *representations*. A resource representation is understood as a document that describes the resource in a formal language.

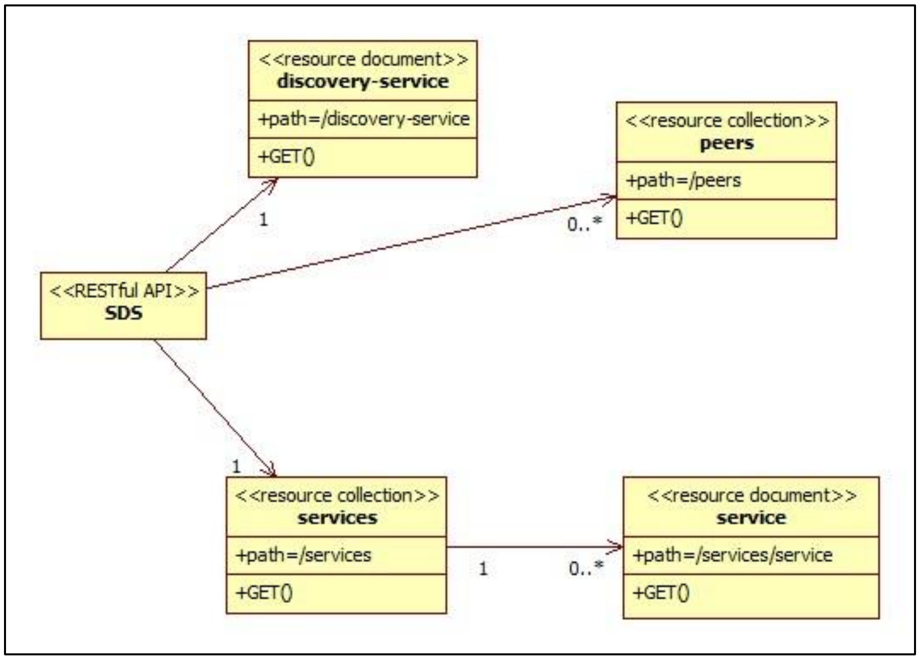

Figure 13 Resource Model

2.3.1. Discovery Service Resource

2.3.1.1. Description

name	discovery service
description	A resource that allows a requestor to retrieve a description of a discovery service.
path	/discovery-service
http method	GET
template	/discovery-service

2.3.1.2. Parameters

None.

2.3.1.3. Representation

The resource representation is rendered in JSON or XML.

2.3.2. Peers Resource

2.3.2.1. Description

name	peers

description	A resource that allows a requestor to retrieve a collection of references to other discovery services (peers).
path	/peers
http method	GET
template	/peers

2.3.2.2. Parameters

None.

2.3.2.3. Representation

The resource representation is rendered in JSON or XML.

2.3.3. Services Resource

2.3.3.1. Description

name	services
description	A resource that allows a requestor to retrieve a collection of references to services.
path	/services
http method	GET
template	/services

2.3.3.2. Parameters

name	service-category
permissible value	values defined in http://semantics.aero/service-category taxonomy
optional	true

name	availability-status
permissible value	values defined in http://semantics.aero/availability-status taxonomy
optional	true

name	interface-type
permissible value	values defined in http://semantics.aero/interface-type taxonomy
optional	true

2.3.3.3. Representation

The resource representation is rendered in JSON or XML.

2.3.4. Service Resource

2.3.4.1. Description

name	service
description	A resource that allows a requestor to retrieve information about a specific service.

path	/services/service
http method	GET
template	/ services/{service-id}

2.3.4.2. Parameters

A *path parameter* “service-id”. The parameter value is a valid id (always a URI) of a service as retrieved from the invocation of the resource “services” (see section 2.3.3).

2.3.4.3. Representation

The resource representation is rendered in JSON or XML.

3. Interface Requirements

A service interface, as it is prescribed by this specification, defines the operations supported by a discovery service, with each operation representing a simple interaction between the service or a user and the service.
As described in REST Web Service architecture, each operation executes one of the HTTP Methods defined in RFC 7231 [1], which specifies the actions to be taken in order to access a resource. It could be noted that each operation represents a unique combination of a resource path and an HTTP Method that is executed to access and manipulate the resource.
The operations defined by this specification implement only the HTTP GET method to retrieve a representation of the associated resource. All operations are inherently synchronous and idempotent.

a. All instances of SDS SHALL be valid with the SDS OpenAPI schema shown in Figure 14.

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "SWIM Discovery Service (SDS) API",
    "description": "OpenAPI-compliant description of the interface offered by
a SWIM Discovery Service (SDS)",
    "contact": {
      "name": "FAA SWIM Governance Lead",
      "email": "mark.kaplun@faa.gov"
    },
    "version": "1.0.0"
  },
  "paths": {
    "/discovery-service": {
      "get": {
        "summary": "A resource that allows a requestor to retrieve a
description of a discovery service.",
        "description": "A resource that allows a requestor to retrieve a
description of a discovery service.",
        "operationId": "GetDiscoveryService",
        "responses": {
          "200": {
            "description": "A representation of DiscoveryService object",
            "content": {
              "application/xml": {
                "schema": {
                  "$ref": "discovery-service.json"
                }
              }
            }
          }
        }
      }
    },
    "/peers": {
      "get": {
        "summary": "A resource that allows a requestor to retrieve a
collection of references to other discovery services (peers).",
        "description": "A resource that allows a requestor to retrieve a
collection of references to other discovery services (peers).",
        "operationId": "GetPeers",
        "responses": {
          "200": {
            "description": "A representation of Peers object",
            "content": {
              "application/xml": {
                "schema": {
                  "$ref": "peers.json"
                }
              }
            }
          }
        }
      }
    },
    "/services": {
      "get": {
        "summary": "A resource that allows a requestor to retrieve a
collection of references to services.",
        "description": "A resource that allows a requestor to retrieve a
collection of references to services.",
        "operationId": "GetServices",
        "parameters": [
          {
            "name": "availability-status",
            "in": "query",
            "description": "Values as defined in
http://semantics.aero/availability-status",
            "required": false,
            "style": "form",
            "explode": true,
            "schema": {
              "type": "uri-reference"
            }
          },
          {
            "name": "interface-type",
            "in": "query",
            "description": "Values as defined in
http://semantics.aero/interface-type",
            "required": false,
            "style": "form",
            "explode": true,
            "schema": {
              "type": "uri-reference"
            }
          },
          {
            "name": "service-category",
            "in": "query",
            "description": "Values as defined in
http://semantics.aero/service-category taxonomy",
            "required": false,
            "style": "form",
            "explode": true,
            "schema": {
              "type": "uri-reference"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "A representation of Services object",
            "content": {
              "application/xml": {
                "schema": {
                  "$ref": "services.json"
                }
              }
            }
          }
        }
      }
    },
    "services/service": {
      "get": {
        "summary": "Allows clients to retrieve one or more description of a
service.",
        "description": "Allows clients to retrieve one or more description of
a service the list of which was obtained in the GetServices operation.",
        "operationId": "GetService",
        "parameters": [
          {
            "name": "service-id",
            "in": "query",
            "description": "A reference to a service, always a value from the
list represented by a parent resource 'services'.",
            "required": true,
            "style": "form",
            "explode": true,
            "schema": {
              "type": "string",
              "format": "uri-reference"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "A representation of ServiceDescription object",
            "content": {
              "application/xml": {
                "schema": {
                  "$ref": "sdm-j.json"
                }
              }
            }
          },
          "404": {
            "description": "No service is identified by the given ID."
          }
        }
      }
    }
  }
}
```

Figure 14 SDS OpenAPI Schema

3.1. Operations

The following requirements apply to all operations performed by a discovery service. The operations-specific characteristics and requirements are shown in sub-sections 3.1.1 through 3.1.4.

- **a.** All operations SHALL support the GET HTTP Method as defined in RFC 7231 [1] section 4.3.1.
- **b.** No HTTP Methods other than GET SHALL be supported.
- **c.** Each operation SHALL be read-only, i.e., a requestor does not request, and does not expect, any state changes on the invoking service as a result of applying the operation to a target resource.
- **d.** Each operation SHALL be idempotent, i.e., the intended effect of multiple identical requests on the invoked service is the same as the effect for a single such request.
- **e.** Each operation MAY be cacheable, that is, the received response can be saved for a future use.

3.1.1. GetDiscoveryService

name	GetDiscoveryService
description	Allows a client to retrieve a *discovery service* resource as defined in Section 2.3.1.
path to associated resource	/discovery-service
http method	GET
obligation	required
input	HTTP request message for the “discovery service” resource
output	HTTP message containing the representation of the “discovery service” resource

3.1.1.1. *Example of the URI request*

`http://nsrr.faa.gov/smxs/discovery-service`

3.1.1.2. *Example of the implementation*

```
# Request
GET /discovery-service HTTP/1.1
Host: nsrr.faa.gov/smxs
Accept: application/json

# Response
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
      "id": "http://swim.faa.gov/smxs",
      "name": "SWIM Metadata Exchange Service (SMXS)",
      "description": "The SWIM Metadata Exchange Service (SMXS) is a service
discovery capability that allows consumers to find and retrieve information
about SWIM services.",
      "version": "1.0.0",
      "provider": {
            "name": "FAA SWIM Program",
            "description": "The SWIM program supports information sharing
among NAS stakeholders by providing governance as well as architectural and
technical solutions for identifying, developing, provisioning, and operating
a framework of shareable and reusable services.",
            "web page": "https://www.faa.gov/air_traffic/technology/swim/",
            "point of contact": {
                  "name": "John Doe",
                  "function": "Software Engineer",
                  "email": "john.doe@faa.gov"
            }
      },
      "operations": [
            {
                  "name": "GetDiscoveryService",
            },
            {
                  "name": "GetPeers",
            },
            {
                  "name": "GetServices",
                  "security-constraint": "HTTP Basic"
            }

      ],
      "see also": "https://nsrr.faa.gov/services/smxs"
```

3.1.2. GetPeers

name	GetPeers
description	Allows a client to retrieve a *peers* resource as defined in Section

	2.3.2.
path to associated resource	/peers
http method	GET
obligation	optional
input	HTTP request message for the “peers” resource
output	HTTP message containing the representation of the “peers” resource

3.1.2.1. Example of the URI request

`http://nsrr.faa.gov/smxs/peers`

3.1.2.2. Example of the implementation

```
# Request
GET /peers HTTP/1.1
Host: nsrr.faa.gov/smxs
Accept: application/json

# Response
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "peers": [
        {
            "service-id": "http://swim.example.org/ds",
            "endpoint": "https://swim.example.org:8443/development/ds"
        },
        {
            "service-id": "http://swim.faa.gov/services/smxs",
            "endpoint": "http://nsrr.swim.gov/smxs-1.1"
        }
    ]
}
```

3.1.3. GetServices

name	GetServices
description	Allows a client to retrieve the *services* resource as defined in Section 2.3.3.
path to associated resource	/services
http method	GET
obligation	required
input	HTTP request message for the “services” resource
output	HTTP message containing the representation of the “services” resource

3.1.3.1. Parameters

a. The GetServices operation SHOULD apply any or all parameters defined in section “2.3.3 Services Resource” of this specification.

b. The GetServices operation SHALL NOT override any of the parameters defined in section “2.3.3 Services Resource” of this specification.
c. The GetServices operation SHALL use the parameters only by appending their respective names and values to the operation path (so-called *query parameters*). See example in section 3.1.3.2.

3.1.3.2. Example of the URI request

```
http://nsrr.faa.gov/smxs/services?service-category=discovery&availability-status=prospective&interface-type=resource-oriented
```

3.1.3.3. Example of the implementation

Note: the following example assumes that the URI presented in section 3.1.3.2 has been used

```
# Request
GET /services HTTP/1.1
Host: nsrr.faa.gov/smxs
Accept: application/json
Authorization: Basic dXNlcjpwYXNzd29yZA==

# Response
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
  "services": [
    {
      "id": "http://nsrr.faa.gov/services/smxs",
      "name": "SWIM Metadata Exchange Service (SMXS)",
      "description": "The SWIM Metadata Exchange Service (SMXS) is a service discovery capability that lets consumers find and retrieve information (metadata) about SWIM services provided by the FAA and other SWIM agencies or states.",
      "service-category": {
        "taxonomy": "http://semantics.aero/service-category",
        "code": " http://semantics.aero/service-category/discovery"
      },
      "service-availability-status": {
        "taxonomy": "http://semantics.aero/availability-status",
        "code": " http://semantics.aero/service-category/prospective"
      },
      "interface-type": {
        "taxonomy": "http://semantics.aero/interface-type",
        "code": " http://semantics.aero/interface-type/resource-oriented"
      }
    }
  ]
```

3.1.4. GetService

name	GetService
description	Allows a client to retrieve a description of the service identified by the resource parameter value (see section 2.3.4).
path to associated resource	/service
http method	GET
obligation	required
input	HTTP request message for the “service” resource
output	HTTP message containing the representation of the “service” resource

3.1.4.1. Parameters

The GetService operation SHALL apply a parameter with the following attributes:

name	service-id
description	A reference to a service obtained from implementing GetServices operation (the attribute “service-id” in the received representation).
parameter type	path
template	/services/{service-id}
obligation	required

3.1.4.2. Example of the URI request

`http://swim.org/ds/services/”http:swim.org/fps”`

3.1.4.3. Example of the implementation

```
# Request
GET //services/”http:swim.org/fps” HTTP/1.1
Host: swim.org/fps
Accept: application/json
Authorization: Basic dXNlcjpwYXNzd29yZA==

# Response
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
  "service-description": {
    "service-id": "http://swim.faa.gov/services/fps",
    "profile": {
      "service-id": "http://swim.faa.gov/services/fps",
      "name": "Flight Plan Service (FPS)",
      "description": "A service for filing, updating, or canceling an IFR
(Instrument Flight Rules) flight plan.",
      "version": "1.0.0",
      "category": [
        {
          "category": "Service Category",
          "value": [
            "http://semantics.aero/service-category#flight"
          ],
          "taxonomy": "http://semantics.aero/service-category"
        },
        {
          "category": "Availability Status",
          "value": [
            "http://semantics.aero/availability-status#prospective"
          ],
          "taxonomy": "http://semantics.aero/availability-status"
        },
        {
          "category": "Interface Type",
          "value": [
            "http://semantics.aero/interface-type/method-oriented"
          ],
          "taxonomy": "http://semantics.aero/interface-type"
        }
      ],
      "function": [
        {
          "description": "File a flight plan.",
          "real-world-effect": "A flight plan has been filed and persists in
the FAA Web server for distribution to the FAA flight data processing
application within some parameter time of the estimated departure time."
        },
…
```

Note: this example is abbreviated and cannot be validated.

3.2. Messages

Following the architectural principles of the HTTP defined in RFC 7231 [1], each operation described in section 3.1 specifies the set of exactly two messages that the service sends or receives as part of the operation (request and response message, respectively).

- **a.** All messages SHALL comply with the syntax and semantics prescribed by RFC 7231 [1].
- **b.** All request messages, that is, the messages sent by a user agent or a discovery service, SHALL include a header field "Accept".
- **c.** The value of the field "Accept" SHOULD be "application/json".
- **d.** The value of the field "Accept" MAY be "application/xml".
- **e.** All response messages SHALL include a Content-Type header field to indicate the formal language used by the associated representation.
- **f.** The value "application/json" SHALL be a default value for the Content-Type header field.
- **g.** The value "application/xml" MAY be included in a request message header.
- **h.** All response messages SHALL include a status code as described in RFC 7231 [1] section 6.

4. Security

To protect its data from interception and alteration, a discovery service should follow best practices in Web service security, such as the HTTPS-only standard [12]. Accordingly, requests to a discovery service SHOULD be made over a communication channel secured by the Transport Security Layer/Secure Socket Layer (TLS/SSL) protocol.

A discovery service MAY restrict access to certain information it maintains. In these cases, a discovery service SHOULD require a client (a user or an application) to authenticate itself before access to one or more of its resources is granted. For example, the discovery service provider can choose to advertise services under development only to users within its own organization. Furthermore, it MAY restrict certain operations to a subset of authenticated users. For example, it can choose to allow anyone to retrieve a list of published services through the GetServices operation, while allowing registered users to access the detailed service descriptions by calling the GetService operation.

When authentication is required, the discovery service SHOULD use an authentication method that is compatible with the HTTP protocol, such as HTTP basic or digest authentication [12], or a token-based based authentication mechanism such as the OAuth 2.0 Authorization Framework [14]. A complete list of HTTP methods is maintained in the HTTP Authentication Scheme Registry. [13]