Table of Contents

* [Introduction](#sec:introduction)
  + [Background](#background)
  + [Scope and Objectives of the Reference Architectures Subgroup](#X004b886bb90d787f1739ae86236aaa6892ce7db)
  + [Report Production](#sec:production)
  + [Report Structure](#report-structure)
* [NBDRA Interface Requirements](#sec:interface-requirements)
  + [High-Level Requirements of the Interface Approach](#X88b6bb88472646d064238626fb47a75c22e48cc)
    - [Technology- and Vendor-Agnostic](#technology--and-vendor-agnostic)
    - [Support of Plug-In Compute Infrastructure](#X03e9e806a86c5220ec886afd680a4d5ddd5bfb9)
    - [Orchestration of Infrastructure and Services](#X81cf4fafd86096c030a1f6976d39b6478c590eb)
    - [Orchestration of Big Data Applications and Experiments](#X9c019b5457ae5ae59b90e2db297ffc505326c93)
    - [Reusability](#reusability)
    - [Execution Workloads](#execution-workloads)
    - [Security and Privacy Fabric Requirements](#security-and-privacy-fabric-requirements)
  + [Component-Specific Interface Requirements](#X1e329653acf6a0cfcf796c9488266fb7bc11b34)
    - [System Orchestrator Interface Requirements](#sec:system-orchestrator-requirements)
    - [Data Provider Interface Requirements](#sec:data-provider-requirements)
    - [Data Consumer Interface Requirements](#sec:data-consumer-requirements)
    - [Big Data Application Interface Provider Requirements](#sec:data-application-requirements)
    - [Big Data Provider Framework Interface Requirements](#sec:provider-requirements)
    - [Big Data Application Provider to Big Data Framework Provider Interface](#sec:app-provider-requirements)
* [Specification Paradigm](#sec:spec-paradigm)
  + [Hybrid and Multiple Frameworks](#hybrid-and-multiple-frameworks)
  + [Design by Resource-Oriented Architecture](#design-by-resource-oriented-architecture)
  + [Design by Example](#design-by-example)
  + [Version Management](#version-management)
  + [Interface Compliancy](#interface-compliancy)
  + [Refernce implementations](#refernce-implementations)
* [Specification](#sec:specification)
  + [List of specifications](#sec:spec-table)
  + [Identity](#sec:spec-identity)
    - [Authentication](#authentication)
    - [Organization](#sec:spec-organization)
    - [User](#sec:spec-user)
    - [Public Key Store](#sec:spec-publickeystore)
  + [General Resources](#general-resources)
    - [Timestamp](#sec:spec-timestamp)
    - [Alias](#sec:spec-alias)
    - [Variables](#sec:spec-variables)
    - [Default](#sec:spec-default)
  + [Data Management](#data-management)
    - [Filestore](#sec:spec-filestore)
    - [Replica](#sec:spec-replica)
    - [Database](#sec:spec-database)
    - [Virtual Directory](#sec:spec-virtualdirectory)
  + [Compute Management - Virtual Clusters](#compute-management---virtual-clusters)
    - [Virtual Cluster](#sec:spec-virtualcluster)
    - [Scheduler](#sec:spec-scheduler)
  + [Compute Management - Virtual Machines](#compute-management---virtual-machines)
    - [Image](#sec:spec-image)
    - [Flavor](#sec:spec-flavor)
    - [Virtual Machine](#sec:spec-vm)
    - [Secgroup](#sec:spec-secgroup)
    - [Nic](#sec:spec-nic)
  + [Compute Management - Containers](#compute-management---containers)
    - [Containers](#sec:spec-containers)
  + [Compute Management - Functions](#compute-management---functions)
    - [Microservice](#sec:spec-microservice)
  + [Reservation](#reservation)
    - [Reservation](#sec:spec-reservation)
  + [Data Streams](#data-streams)
    - [Stream](#sec:spec-stream)
    - [Filter](#sec:spec-filter)
  + [Deployment](#deployment)
    - [Deployment](#sec:spec-deployment)
* [Status Codes and Error Responses](#status-codes-and-error-responses)
* [Acronyms and Terms](#acronyms-and-terms)
* [Bibliography](#bibliography)





**Abstract**

This document summarizes interfaces that are instrumental for the interaction with Clouds, Containers, and High Performance Computing (HPC) systems to manage virtual clusters to support the NIST Big Data Reference Architecture (NBDRA). The REpresentational State Transfer (REST) paradigm is used to define these interfaces, allowing easy integration and adoption by a wide variety of frameworks.

Big Data is a term used to describe extensive datasets, primarily in the characteristics of volume, variety, velocity, and/or variability. While opportunities exist with Big Data, the data characteristics can overwhelm traditional technical approaches, and the growth of data is outpacing scientific and technological advances in data analytics. To advance progress in Big Data, the NIST Big Data Public Working Group (NBD-PWG) is working to develop consensus on important fundamental concepts related to Big Data. The results are reported in the *NIST Big Data Interoperability Framework (NBDIF)* series of volumes. This volume, Volume 8, uses the work performed by the NBD-PWG to identify objects instrumental for the NIST Big Data Reference Architecture (NBDRA) which is introduced in the NBDIF: Volume 6, *Reference Architecture*.

**Keywords**

Adoption; barriers; implementation; interfaces; market maturity; organizational maturity; project maturity; system modernization.

**Acknowledgements**

This document reflects the contributions and discussions by the membership of the NBD-PWG, cochaired by Wo Chang (NIST ITL), Bob Marcus (ET-Strategies), and Chaitan Baru (San Diego Supercomputer Center; National Science Foundation). For all versions, the Subgroups were led by the following people: Nancy Grady (SAIC), Natasha Balac (SDSC), and Eugene Luster (R2AD) for the Definitions and Taxonomies Subgroup; Geoffrey Fox (Indiana University) and Tsegereda Beyene (Cisco Systems) for the Use Cases and Requirements Subgroup; Arnab Roy (Fujitsu), Mark Underwood (Krypton Brothers; Synchrony Financial), and Akhil Manchanda (GE) for the Security and Privacy Subgroup; David Boyd (InCadence Strategic Solutions), Orit Levin (Microsoft), Don Krapohl (Augmented Intelligence), and James Ketner (AT&T) for the Reference Architecture Subgroup; and Russell Reinsch (Center for Government Interoperability), David Boyd (InCadence Strategic Solutions), Carl Buffington (Vistronix), and Dan McClary (Oracle), for the Standards Roadmap Subgroup, Gregor von Laszewski (Indiana University) for the Interface Subgroup.

The following milestone releases exist:

* **Version 2.1**: A previous volume used the definitions of the schema based on examples only. It was easier to read but only included the definition of the resources and not the interaction with the resources. This volume was in place until June 2018.
* **Version 2.2**: This version was significantly changed and used OpenAPI 2.0 to specify the interfaces between the various services and components.
* **Version 3.1.1**: The version includes significant improvements of the object specifications but are still using OpenAPI 2.0.
* **Version 3.2.0**: All specifications have been updated to OpenAPI 3.0.2. Significant updates have been done to a number of specifications.

The editors for these documents are:

* Gregor von Laszewski (Indiana University)
* Wo Chang (NIST).

Laurie Aldape (Energetics Incorporated) and Elizabeth Lennon (NIST) provided editorial assistance across all NBDIF volumes.

NIST SP 1500-9, Draft NIST Big Data Interoperability Framework: Volume 8, Reference Architecture Interfaces, Version 2 has been collaboratively authored by the NBD-PWG. As of the date of publication, there are over six hundred NBD-PWG participants from industry, academia, and government. Federal agency participants include the National Archives and Records Administration (NARA), National Aeronautics and Space Administration (NASA), National Science Foundation (NSF), and the U.S. Departments of Agriculture, Commerce, Defense, Energy, Census, Health and Human Services, Homeland Security, Transportation, Treasury, and Veterans Affairs.

NIST would like to acknowledge the specific contributions to this volume, during Version 1 and/or Version 2 activities. *Contributors* are members of the NIST Big Data Public Working Group who dedicated great effort to prepare and gave substantial time on a regular basis to research and development in support of this document. This includes the following NBD-PWG members:

* Gregor von Laszewski, Indiana University
* Wo Chang, National Institute of Standard and Technology,
* Fugang Wang, Indiana University
* Geoffrey C. Fox, Indiana University
* Shirish Joshi, Indiana University
* Badi Abdul-Wahid, formerly Indiana Univresity
* Alicia Zuniga-Alvarado, Consultant
* Robert C. Whetsel, DISA/NBIS
* Pratik Thakkar, Philips

**Executive Summary**

The *NIST Big Data Interoperability Framework (NBDIF): Volume 8, Reference Architecture Interfaces* document was prepared by the NIST Big Data Public Working Group (NBD-PWG) Reference Architecture Subgroup to identify interfaces in support of the NIST Big Data Reference Architecture (NBDRA). The interface define resources that are part of the NBDRA. These resources are formulated in OpenAPI 3.0.2 format and can be easily integrated into a REpresentational State Transfer (REST) framework or an object-based framework.

The resources were categorized in groups that are identified by the NBDRA set forward in the *NBDIF: Volume 6, Reference Architecture* document. While the *NBDIF: Volume 3, Use Cases and General Requirements* document provides *application-*oriented high-level use cases, the use cases defined in this document are subsets of them and focus on *interface* use cases. The interface use cases are not meant to be complete examples, but showcase why the resource has been defined. Hence, the interfaces use cases are only representative, and do not encompass the entire spectrum of Big Data usage. All the interfaces were openly discussed in the working group (NIST, n.d.). Additions to the interfaces are welcome and the NBD-PWG is open to discuss any contributions.

The *NIST Big Data Interoperability Framework (NBDIF)* was released in three versions, which correspond to the three stages of the NBD-PWG work. Version 3 (current version) of the NBDIF volumes resulted from Stage 3 work with major emphasis on the validation of the NBDRA Interfaces and content enhancement. Stage 3 work built upon the foundation created during Stage 2 and Stage 1. The current effort documented in this volume reflects concepts developed within the rapidly evolving field of Big Data. The three stages (in reverse order) aim to achieve the following with respect to the NIST Big Data Reference Architecture (NBDRA):

* Stage 1: Identify the high-level Big Data reference architecture key components, which are technology-, infrastructure-, and vendor-agnostic.
* Stage 2: Define general interfaces between the NBDRA components; and
* Stage 3: Validate the NBDRA by building Big Data general applications through the general interfaces;

The NBDIF consists of nine volumes, each of which addresses a specific key topic, resulting from the work of the NBD-PWG. The nine volumes are as follows:

* Volume 1, Definitions (W. L. Chang (Co-Chair), N. Grady (Subgroup Co-chair), and NIST Big Data Public Working Group 2019a)
* Volume 2, Taxonomies (W. L. Chang (Co-Chair), N. Grady (Subgroup Co-chair), and NIST Big Data Public Working Group 2019b)
* Volume 3, Use Cases and General Requirements (W. L. Chang (Co-Chair), G. Fox (Subgroup Co-chair), and NIST Big Data Public Working Group 2019)
* Volume 4, Security and Privacy (W. L. Chang (Co-Chair), A. Roy (Subgroup Co-chair), M. Underwood (Subgroup Co-chair), and NIST Big Data Public Working Group 2019)
* Volume 5, Architectures White Paper Survey (W. L. Chang (Co-Chair), S. Mishra (Editor), and NIST Big Data Public Working Group 2019)
* Volume 6, Reference Architecture (W. L. Chang (Co-Chair), D. Boyd (Subgroup Co-chair), O. Levin (Version 1 Subgroup Co-Chair), and NIST Big Data Public Working Group 2019)
* Volume 7, Standards Roadmap (W. L. Chang (Co-Chair), R. Reinsch (Subgroup Co-chair), D. Boyd (Version 1 Subgroup Co-chair), C. Buffington (Version 1 Subgroup Co-chair), and NIST Big Data Public Working Group 2019)
* Volume 8, Reference Architecture Interfaces (this volume) (W. L. Chang (Co-Chair), G. von Laszewski (Editor), and NIST Big Data Public Working Group 2019)
* Volume 9, Adoption and Modernization (W. L. Chang (Co-Chair), R. Reinsch (Subgroup Co-chair), C. Austin (Editor), and NIST Big Data Public Working Group 2019)

During Stage 1, Volumes 1 through 7 were conceptualized, organized and written. The finalized Version 1 documents can be downloaded from the V1.0 Final Version page of the NBD-PWG website (NIST 2015b).

During Stage 2, the NBD-PWG developed Version 2 of the NBDIF Version 1 volumes, with the exception of Volume 5, which contained the completed architecture survey work that was used to inform Stage 1 work of the NBD-PWG. The goals of Version 2 were to enhance the Version 1 content, define general interfaces between the NBDRA components by aggregating low-level interactions into high-level general interfaces, and demonstrate how the NBDRA can be used. As a result of the Stage 2 work, the need for NBDIF Volume 8 and NBDIF Volume 9 were identified and the two new volumes were created. Version 2 of the NBDIF volumes, resulting from Stage 2 work, can be downloaded from the V2.0 Final Version page of the NBD-PWG website (NIST 2015a).

This document is the result of Stage 3 work of the NBD-PWG. Coordination of the group is conducted on the NBD-PWG web page (NIST, n.d.).

# Introduction

## Background

There is broad agreement among commercial, academic, and government leaders about the potential of Big Data to spark innovation, fuel commerce, and drive progress. Big Data is the common term used to describe the deluge of data in today’s networked, digitized, sensor-laden, and information-driven world. The availability of vast data resources carries the potential to answer questions previously out of reach, including the following:

* How can a potential pandemic reliably be detected early enough to intervene?
* Can new materials with advanced properties be predicted before these materials have ever been synthesized?
* How can the current advantage of the attacker over the defender in guarding against cybersecurity threats be reversed?

There is also broad agreement on the ability of Big Data to overwhelm traditional approaches. The growth rates for data volumes, speeds, and complexity are outpacing scientific and technological advances in data analytics, management, transport, and data user spheres.

Despite widespread agreement on the inherent opportunities and current limitations of Big Data, a lack of consensus on some important fundamental questions continues to confuse potential users and stymie progress. These questions include the following:

* How is Big Data defined?
* What attributes define Big Data solutions?
* What is new in Big Data?
* What is the difference between Big Data and *bigger data* that has been collected for years?
* How is Big Data different from traditional data environments and related applications?
* What are the essential characteristics of Big Data environments?
* How do these environments integrate with currently deployed architectures?
* What are the central scientific, technological, and standardization challenges that need to be addressed to accelerate the deployment of robust, secure Big Data solutions?

Within this context, on March 29, 2012, the White House announced the Big Data Research and Development Initiative (The White House Office of Science and Technology Policy, “Big Data is a Big Deal,” *OSTP Blog*, accessed February 21, 2014 (The White House Office of Science and Technology Policy 2014). The initiative’s goals include helping to accelerate the pace of discovery in science and engineering, strengthening national security, and transforming teaching and learning by improving analysts’ ability to extract knowledge and insights from large and complex collections of digital data.

Six federal departments and their agencies announced more than $200 million in commitments spread across more than 80 projects, which aim to significantly improve the tools and techniques needed to access, organize, and draw conclusions from huge volumes of digital data. The initiative also challenged industry, research universities, and nonprofits to join with the federal government to make the most of the opportunities created by Big Data.

Motivated by the White House initiative and public suggestions, the National Institute of Standards and Technology (NIST) accepted the challenge to stimulate collaboration among industry professionals to further the secure and effective adoption of Big Data. As a result of NIST’s Cloud and Big Data Forum held on January 15–17, 2013, there was strong encouragement for NIST to create a public working group for the development of a Big Data Standards Roadmap. Forum participants noted that this roadmap should define and prioritize Big Data requirements, including interoperability, portability, reusability, extensibility, data usage, analytics, and technology infrastructure. In doing so, the roadmap would accelerate the adoption of the most secure and effective Big Data techniques and technology.

On June 19, 2013, the NIST Big Data Public Working Group (NBD-PWG) was launched with extensive participation by industry, academia, and government from across the nation. The scope of the NBD-PWG involves forming a community of interests from all sectors—including industry, academia, and government—with the goal of developing consensus on definitions, taxonomies, secure reference architectures, security and privacy, and, from these, a standards roadmap. Such a consensus would create a vendor-neutral, technology- and infrastructure-independent framework that would enable Big Data stakeholders to identify and use the best analytics tools for their processing and visualization requirements on the most suitable computing platform and cluster, while also allowing added value from Big Data service providers.

The *NIST Big Data Interoperability Framework (NBDIF)* was released in three versions, which correspond to the three stages of the NBD-PWG work. Version 3 (current version) of the NBDIF volumes resulted from Stage 3 work with major emphasis on the validation of the NBDRA Interfaces and content enhancement. Stage 3 work built upon the foundation created during Stage 2 and Stage 1. The current effort documented in this volume reflects concepts developed within the rapidly evolving field of Big Data. The three stages (in reverse order) aim to achieve the following with respect to the NIST Big Data Reference Architecture (NBDRA).

Stage 3: Validate the NBDRA by building Big Data general applications through the general interfaces; Stage 2: Define general interfaces between the NBDRA components; and Stage 1: Identify the high-level Big Data reference architecture key components, which are technology-, infrastructure-, and vendor-agnostic.

The NBDIF consists of nine volumes, each of which addresses a specific key topic, resulting from the work of the NBD-PWG. The nine volumes are as follows:

* Volume 1, Definitions (W. L. Chang (Co-Chair), N. Grady (Subgroup Co-chair), and NIST Big Data Public Working Group 2019a)
* Volume 2, Taxonomies (W. L. Chang (Co-Chair), N. Grady (Subgroup Co-chair), and NIST Big Data Public Working Group 2019b)
* Volume 3, Use Cases and General Requirements (W. L. Chang (Co-Chair), G. Fox (Subgroup Co-chair), and NIST Big Data Public Working Group 2019)
* Volume 4, Security and Privacy (W. L. Chang (Co-Chair), A. Roy (Subgroup Co-chair), M. Underwood (Subgroup Co-chair), and NIST Big Data Public Working Group 2019)
* Volume 5, Architectures White Paper Survey (W. L. Chang (Co-Chair), S. Mishra (Editor), and NIST Big Data Public Working Group 2019)
* Volume 6, Reference Architecture (W. L. Chang (Co-Chair), D. Boyd (Subgroup Co-chair), O. Levin (Version 1 Subgroup Co-Chair), and NIST Big Data Public Working Group 2019)
* Volume 7, Standards Roadmap (W. L. Chang (Co-Chair), R. Reinsch (Subgroup Co-chair), D. Boyd (Version 1 Subgroup Co-chair), C. Buffington (Version 1 Subgroup Co-chair), and NIST Big Data Public Working Group 2019)
* Volume 8, Reference Architecture Interfaces (this volume) (W. L. Chang (Co-Chair), G. von Laszewski (Editor), and NIST Big Data Public Working Group 2019)
* Volume 9, Adoption and Modernization (W. L. Chang (Co-Chair), R. Reinsch (Subgroup Co-chair), C. Austin (Editor), and NIST Big Data Public Working Group 2019)

During Stage 1, Volumes 1 through 7 were conceptualized, organized and written. The finalized Version 1 documents can be downloaded from the V1.0 Final Version page of the NBD-PWG website (NIST 2015b).

During Stage 2, the NBD-PWG developed Version 2 of the NBDIF Version 1 volumes, with the exception of Volume 5, which contained the completed architecture survey work that was used to inform Stage 1 work of the NBD-PWG. The goals of Version 2 were to enhance the Version 1 content, define general interfaces between the NBDRA components by aggregating low-level interactions into high-level general interfaces, and demonstrate how the NBDRA can be used. As a result of the Stage 2 work, the need for NBDIF Volume 8 and NBDIF Volume 9 were identified and the two new volumes were created. Version 2 of the NBDIF volumes, resulting from Stage 2 work, can be downloaded from the V2.0 Final Version page of the NBD-PWG website (NIST 2015a).

## Scope and Objectives of the Reference Architectures Subgroup

Reference architectures provide “an authoritative source of information about a specific subject area that guides and constrains the instantiations of multiple architectures and solutions” (Department of Defense 2010). Reference architectures generally serve as a foundation for solution architectures and may also be used for comparison and alignment of instantiations of architectures and solutions.

The goal of the NBD-PWG Reference Architecture Subgroup is to develop an open reference architecture for Big Data that achieves the following objectives:

* Provides a common language for the various stakeholders;
* Encourages adherence to common standards, specifications, and patterns;
* Provides consistent methods for implementation of technology to solve similar problem sets;
* Illustrates and improves understanding of the various Big Data components, processes, and systems, in the context of a vendor- and technology-agnostic Big Data conceptual model;
* Provides a technical reference for U.S. government departments, agencies, and other consumers to understand, discuss, categorize, and compare Big Data solutions; and
* Facilitates analysis of candidate standards for interoperability, portability, reusability, and extendibility.

The NBDRA is a high-level conceptual model crafted to serve as a tool to facilitate open discussion of the requirements, design structures, and operations inherent in Big Data. The NBDRA is intended to facilitate the understanding of the operational intricacies in Big Data. It does not represent the system architecture of a specific Big Data system, but rather is a tool for describing, discussing, and developing system-specific architectures using a common framework of reference. The model is not tied to any specific vendor products, services, or reference implementation, nor does it define prescriptive solutions that inhibit innovation.

The NBDRA does not address the following:

* Detailed specifications for any organization’s operational systems;
* Detailed specifications of information exchanges or services; and
* Recommendations or standards for integration of infrastructure products.

The goals of the Subgroup were realized throughout the three planned phases of the NBD-PWG work, as outlined in sec. 1.3.

## Report Production

The *NBDIF: Volume 8,* *References Architecture Interfaces* is one of nine volumes, whose overall aims are to define and prioritize Big Data requirements, including interoperability, portability, reusability, extensibility, data usage, analytic techniques, and technology infrastructure to support secure and effective adoption of Big Data. The overall goals of this volume are to define and specify interfaces to implement the Big Data Reference Architecture. This volume arose from discussions during the weekly NBD-PWG conference calls. Topics included in this volume began to take form in Phase 2 of the NBD-PWG work. During the discussions, the NBD-PWG identified the need to specify a variety of interfaces.

Phase 3 work, which built upon the groundwork developed during Phase 2, included an early specification based on resource object specifications that provided a simplified version of an API interface design.

## Report Structure

To enable interoperability between the NBDRA components, a list of well-defined NBDRA interfaces is needed. These interfaces are documented in this volume. To introduce them, the NBDRA structure will be followed, focusing on interfaces that allow bootstrapping of the NBDRA. The document begins with a summary of requirements that will be integrated into our specifications. Subsequently, each section will introduce a number of objects that build the core of the interface addressing a specific aspect of the NBDRA. A selected number of *interface use cases* will be showcased to outline how the specific interface can be used in a reference implementation of the NBDRA. Validation of this approach can be achieved while applying it to the application use cases that have been gathered in the *NBDIF: Volume 3, Use Cases and Requirements* document. These application use cases have considerably contributed towards the design of the NBDRA. Hence the expectation is that: (1) the interfaces can be used to help implement a Big Data architecture for a specific use case; and (2) the proper implementation. This approach can facilitate subsequent analysis and comparison of the use cases.

The organization of this document roughly corresponds to the process used by the NBD-PWG to develop the interfaces. Following the introductory material presented in sec. 1, the remainder of this document is organized as follows:

* sec. 2 presents the interface requirements;
* sec. 3 presents the specification paradigm that is used;
* sec. 4 presents several objects grouped by functional use while providing a summary table of selected proposed objects in sec. 4.1.

While each NBDIF volume was created with a specific focus within Big Data, all volumes are interconnected. During creation, the volumes gave and/or received input from other volumes. Broad topics (e.g., definition, architecture) may be discussed in several volumes with the discussion circumscribed by the volume’s particular focus. Arrows shown in fig. 1 indicate the main flow of input/output. Volumes 2, 3, and 5 (blue circles) are essentially standalone documents that provide output to other volumes (e.g., to Volume 6). These volumes contain the initial situational awareness research. Volumes 4, 7, 8, and 9 (green circles) primarily received input from other volumes during the creation of the particular volume. Volumes 1 and 6 (red circles) were developed using the initial situational awareness research and continued to be modified based on work in other volumes. These volumes also provided input to the green circle volumes.



Figure 1: Figure 1: NBDIF Documents Navigation Diagram Provides Content Flow Between Volumes

# NBDRA Interface Requirements

The development of a Big Data reference architecture requires a thorough understanding of current techniques, issues, and concerns. To this end, the NBD-PWG collected use cases to gain an understanding of current applications of Big Data, conducted a survey of reference architectures to understand commonalities within Big Data architectures in use, developed a taxonomy to understand and organize the information collected, and reviewed existing technologies and trends relevant to Big Data. The results of these NBD-PWG activities were used in the development of the NBDRA (fig. 2) and the interfaces presented herein. Detailed descriptions of these activities can be found in the other volumes of the *NBDIF*.



Figure 2: Figure 2: NIST Big Data Reference Architecture (NBDRA)

This vendor-neutral, technology- and infrastructure-agnostic conceptual model, the NBDRA, is shown in fig. 2 and represents a Big Data system composed of five logical functional components connected by interoperability interfaces (i.e., services). Two fabrics envelop the components, representing the interwoven nature of management and security and privacy with all five of the components. These two fabrics provide services and functionality to the five main roles in the areas specific to Big Data and are crucial to any Big Data solution. Note: None of the terminology or diagrams in these documents is intended to be normative or to imply any business or deployment model. The terms *provider* and *consumer* as used are descriptive of general roles and are meant to be informative in nature.

The NBDRA is organized around five major roles and multiple sub-roles aligned along two axes representing the two Big Data value chains: the Information Value (horizontal axis) and the Information Technology (IT; vertical axis). Along the Information Value axis, the value is created by data collection, integration, analysis, and applying the results following the value chain. Along the IT axis, the value is created by providing networking, infrastructure, platforms, application tools, and other IT services for hosting of and operating the Big Data in support of required data applications. At the intersection of both axes is the Big Data Application Provider role, indicating that data analytics and its implementation provide the value to Big Data stakeholders in both value chains. The term *provider* as part of the Big Data Application Provider and Big Data Framework Provider is there to indicate that those roles provide or implement specific activities and functions within the system. It does not designate a service model or business entity.

The DATA arrows in fig. 2 show the flow of data between the system’s main roles. Data flows between the roles either physically (i.e., by value) or by providing its location and the means to access it (i.e., by reference). The SW arrows show transfer of software tools for processing of Big Data *in situ*. The Service Use arrows represent software programmable interfaces. While the main focus of the NBDRA is to represent the run-time environment, all three types of communications or transactions can happen in the configuration phase as well. Manual agreements (e.g., service-level agreements) and human interactions that may exist throughout the system are not shown in the NBDRA.

Detailed information on the NBDRA conceptual model is presented in the *NBDIF: Volume 6, Reference Architecture* document.

Prior to outlining the specific interfaces, general requirements are introduced and the interfaces are defined.

## High-Level Requirements of the Interface Approach

This section focuses on the high-level requirements of the interface approach that are needed to implement the reference architecture depicted in fig. 2.

### Technology- and Vendor-Agnostic

Due to the many different tools, services, and infrastructures available in the general area of Big Data, an interface ought to be as vendor-independent as possible, while, at the same time, be able to leverage best practices. Hence, a methodology is needed that allows extension of interfaces to adapt and leverage existing approaches, but also allows the interfaces to provide merit in easy specifications that assist the formulation and definition of the NBDRA.

### Support of Plug-In Compute Infrastructure

As Big Data is not just about hosting data, but about analyzing data, the interfaces provided herein must encapsulate a rich infrastructure environment that is used by data scientists. This includes the ability to integrate (or plug-in) various compute resources and services to provide the necessary compute power to analyze the data. These resources and services include the following:

* Access to hierarchy of compute resources from the laptop/desktop, servers, data clusters, and clouds;
* The ability to integrate special-purpose hardware such as graphics processing units (GPUs) and field-programmable gate arrays (FPGAs) that are used in accelerated analysis of data; and
* The integration of services including microservices that allow the analysis of the data by delegating them to hosted or dynamically deployed services on the infrastructure of choice.

### Orchestration of Infrastructure and Services

From review of the use case collection, presented in the *NBDIF: Volume 3, Use Cases and General Requirements* document (W. L. Chang (Co-Chair), G. Fox (Subgroup Co-chair), and NIST Big Data Public Working Group 2019), the need arose to address the mechanism of preparing suitable infrastructures for various use cases. As not every infrastructure is suited for every use case, a custom infrastructure may be needed. As such, this document is not attempting to deliver a single deployed NBDRA, but allow the setup of an infrastructure that satisfies the particular use case. To achieve this task, it is necessary to provision software stacks and services while orchestrating their deployment and leveraging infrastructures. It is not the focus of this document to replace existing orchestration software and services, but provide an interface to them to leverage them as part of defining and creating the infrastructure. Various orchestration frameworks and services could therefore be leveraged, even as part of the same framework, and work in orchestrated fashion to achieve the goal of preparing an infrastructure suitable for one or more applications.

### Orchestration of Big Data Applications and Experiments

The creation of the infrastructure suitable for Big Data applications provides the basic computing environment. However, Big Data applications may require the creation of sophisticated applications as part of interactive experiments to analyze and probe the data. For this purpose, the applications must be able to orchestrate and interact with experiments conducted on the data while assuring reproducibility and correctness of the data. For this purpose, a *System Orchestrator* (either the data scientists or a service acting on behalf of the data scientist) is used as the command center to interact on behalf of the Big Data Application Provider to orchestrate dataflow from Data Provider, carry out the Big Data application life cycle with the help of the Big Data Framework Provider, and enable the Data Consumer to consume Big Data processing results. An interface is needed to describe these interactions and to allow leveraging of experiment management frameworks in scripted fashion. A customization of parameters is needed on several levels. On the highest level, application-motivated parameters are needed to drive the orchestration of the experiment. On lower levels, these high-level parameters may drive and create service-level agreements, augmented specifications, and parameters that could even lead to the orchestration of infrastructure and services to satisfy experiment needs.

### Reusability

The interfaces provided must encourage reusability of the infrastructure, services, and experiments described by them. This includes (1) reusability of available analytics packages and services for adoption; (2) deployment of customizable analytics tools and services; and (3) operational adjustments that allow the services and infrastructure to be adapted while at the same time allowing for reproducible experiment execution.

### Execution Workloads

One of the important aspects of distributed Big Data services can be that the data served is simply too big to be moved to a different location. Instead, an interface could allow the description and packaging of analytics algorithms, and potentially also tools, as a payload to a data service. This can be best achieved, not by sending the detailed execution, but by sending an interface description that describes how such an algorithm or tool can be created on the server and be executed under security considerations (integrated with authentication and authorization in mind).

### Security and Privacy Fabric Requirements

Although the focus of this document is not security and privacy, which are documented in the *NBDIF: Volume 4, Security and Privacy* (W. L. Chang (Co-Chair), A. Roy (Subgroup Co-chair), M. Underwood (Subgroup Co-chair), and NIST Big Data Public Working Group 2019), the interfaces defined herein must be capable of integration into a secure reference architecture that supports secure execution, secure data transfer, and privacy. Consequently, the interfaces defined herein can be augmented with frameworks and solutions that provide such mechanisms. Thus, diverse requirement needs stemming from different use cases addressing security need to be distinguished. To contrast that the security requirements between applications can vary drastically, the following example is provided. Although many of the interfaces and their objects to support Big Data applications in physics are similar to those in healthcare, they differ in the integration of security interfaces and policies. While in physics the protection of data is less of an issue, it is a stringent requirement in healthcare. Thus, deriving architectural frameworks for both may use largely similar components, but addressing security issues will be very different. The security of interfaces may be addressed in other documents. In this document, they are considered an advanced use case showcasing that the validity of the specifications introduced here is preserved, even if security and privacy requirements differ vastly among application use cases.

## Component-Specific Interface Requirements

This section summarizes the requirements for the interfaces of the NBDRA components. The five components are listed in fig. 2 and addressed in sec. 2.2.1 (System Orchestrator Interface Requirements) and sec. 2.2.4 (Big Data Application Provider to Big Data Framework Provider Interface) of this document. The five main functional components of the NBDRA represent the different technical roles within a Big Data system and are the following:

* System Orchestrator: Defines and integrates the required data application activities into an operational vertical system (see sec. 2.2.1);
* Data Provider: Introduces new data or information feeds into the Big Data system (see sec. 2.2.2);
* Data Consumer: Includes end users or other systems that use the results of the Big Data Application Provider (see sec. 2.2.3);
* Big Data Application Provider: Executes a data life cycle to meet security and privacy requirements as well as System Orchestrator-defined requirements (see sec. 2.2.4);
* Big Data Framework Provider: Establishes a computing framework in which to execute certain transformation applications while protecting the privacy and integrity of data (see sec. 2.2.5); and
* Big Data Application Provider to Framework Provider Interface: Defines an interface between the application specification and the provider (see sec. 2.2.6).

### System Orchestrator Interface Requirements

The System Orchestrator role includes defining and integrating the required data application activities into an operational vertical system. Typically, the System Orchestrator involves a collection of more specific roles, performed by one or more actors, which manage and orchestrate the operation of the Big Data system. These actors may be human components, software components, or some combination of the two. The function of the System Orchestrator is to configure and manage the other components of the Big Data architecture to implement one or more workloads that the architecture is designed to execute. The workloads managed by the System Orchestrator may be assigning/provisioning framework components to individual physical or virtual nodes at the lower level, or providing a graphical user interface that supports the specification of workflows linking together multiple applications and components at the higher level. The System Orchestrator may also, through the Management Fabric, monitor the workloads and system to confirm that specific quality of service requirements is met for each workload, and may elastically assign and provision additional physical or virtual resources to meet workload requirements resulting from changes/surges in the data or number of users/transactions. The interface to the System Orchestrator must be capable of specifying the task of orchestration the deployment, configuration, and the execution of applications within the NBDRA. A simple vendor-neutral specification to coordinate the various parts either as simple parallel language tasks or as a workflow specification is needed to facilitate the overall coordination. Integration of existing tools and services into the System Orchestrator as extensible interfaces is desirable.

### Data Provider Interface Requirements

The Data Provider role introduces new data or information feeds into the Big Data system for discovery, access, and transformation by the Big Data system. New data feeds are distinct from the data already in use by the system and residing in the various system repositories. Similar technologies can be used to access both new data feeds and existing data. The Data Provider actors can be anything from a sensor, to a human inputting data manually, to another Big Data system. Interfaces for data providers must be able to specify a data provider so it can be located by a data consumer. It also must include enough details to identify the services offered so they can be pragmatically reused by consumers. Interfaces to describe pipes and filters must be addressed.

### Data Consumer Interface Requirements

Like the Data Provider, the role of Data Consumer within the NBDRA can be an actual end user or another system. In many ways, this role is the mirror image of the Data Provider, with the entire Big Data framework appearing like a Data Provider to the Data Consumer. The activities associated with the Data Consumer role include the following:

* Search and Retrieve,
* Download,
* Analyze Locally,
* Reporting,
* Visualization, and
* Data to Use for Their Own Processes.

The interface for the data consumer must be able to describe the consuming services and how they retrieve information or leverage data consumers.

### Big Data Application Interface Provider Requirements

The Big Data Application Provider role executes a specific set of operations along the data life cycle to meet the requirements established by the System Orchestrator, as well as meeting security and privacy requirements. The Big Data Application Provider is the architecture component that encapsulates the business logic and functionality to be executed by the architecture. The interfaces to describe Big Data applications include interfaces for the various subcomponents including collections, preparation/curation, analytics, visualization, and access. Some of the interfaces used in these subcomponents can be reused from other interfaces, which are introduced in other sections of this document. Where appropriate, application-specific interfaces will be identified and examples provided with a focus on use cases as identified in the *NBDIF: Volume 3, Use Cases and General Requirements*.

#### Collection

In general, the collection activity of the Big Data Application Provider handles the interface with the Data Provider. This may be a general service, such as a file server or web server configured by the System Orchestrator to accept or perform specific collections of data, or it may be an application-specific service designed to pull data or receive pushes of data from the Data Provider. Since this activity is receiving data at a minimum, it must store/buffer the received data until it is persisted through the Big Data Framework Provider. This persistence need not be to physical media but may simply be to an in-memory queue or other service provided by the processing frameworks of the Big Data Framework Provider. The collection activity is likely where the extraction portion of the Extract, Transform, Load (ETL)/Extract, Load, Transform (ELT) cycle is performed. At the initial collection stage, sets of data (e.g., data records) of similar structure are collected (and combined), resulting in uniform security, policy, and other considerations. Initial metadata is created (e.g., subjects with keys are identified) to facilitate subsequent aggregation or look-up methods.

#### Preparation

The preparation activity is where the transformation portion of the ETL/ELT cycle is likely performed, although analytics activity will also likely perform advanced parts of the transformation. Tasks performed by this activity could include data validation (e.g., checksums/hashes, format checks), cleaning (e.g., eliminating bad records/fields), outlier removal, standardization, reformatting, or encapsulating. This activity is also where source data will frequently be persisted to archive storage in the Big Data Framework Provider and provenance data will be verified or attached/associated. Verification or attachment may include optimization of data through manipulations (e.g., deduplication) and indexing to optimize the analytics process. This activity may also aggregate data from different Data Providers, leveraging metadata keys to create an expanded and enhanced data set.

#### Analytics

The analytics activity of the Big Data Application Provider includes the encoding of the low-level business logic of the Big Data system (with higher-level business process logic being encoded by the System Orchestrator). The activity implements the techniques to extract knowledge from the data based on the requirements of the vertical application. The requirements specify the data processing algorithms to produce new insights that will address the technical goal. The analytics activity will leverage the processing frameworks to implement the associated logic. This typically involves the activity providing software that implements the analytic logic to the batch and/or streaming elements of the processing framework for execution. The messaging/communication framework of the Big Data Framework Provider may be used to pass data or control functions to the application logic running in the processing frameworks. The analytic logic may be broken up into multiple modules to be executed by the processing frameworks which communicate, through the messaging/communication framework, with each other and other functions instantiated by the Big Data Application Provider.

#### Visualization

The visualization activity of the Big Data Application Provider prepares elements of the processed data and the output of the analytic activity for presentation to the Data Consumer. The objective of this activity is to format and present data in such a way as to optimally communicate meaning and knowledge. The visualization preparation may involve producing a text-based report or rendering the analytic results as some form of graphic. The resulting output may be a static visualization and may simply be stored through the Big Data Framework Provider for later access. However, the visualization activity frequently interacts with the access activity, the analytics activity, and the Big Data Framework Provider (processing and platform) to provide interactive visualization of the data to the Data Consumer based on parameters provided to the access activity by the Data Consumer. The visualization activity may be completely application-implemented, leverage one or more application libraries, or may use specialized visualization processing frameworks within the Big Data Framework Provider.

#### Access

The access activity within the Big Data Application Provider is focused on the communication/interaction with the Data Consumer. Like the collection activity, the access activity may be a generic service such as a web server or application server that is configured by the System Orchestrator to handle specific requests from the Data Consumer. This activity would interface with the visualization and analytic activities to respond to requests from the Data Consumer (who may be a person) and uses the processing and platform frameworks to retrieve data to respond to Data Consumer requests. In addition, the access activity confirms that descriptive and administrative metadata and metadata schemes are captured and maintained for access by the Data Consumer and as data is transferred to the Data Consumer. The interface with the Data Consumer may be synchronous or asynchronous in nature and may use a pull or push paradigm for data transfer.

### Big Data Provider Framework Interface Requirements

Data for Big Data applications are delivered through data providers. They can be either local providers, data contributed by a user, or distributed data providers, data on the Internet. This interface must be able to provide the following functionality:

* Interfaces to files,
* Interfaces to virtual data directories,
* Interfaces to data streams, and
* Interfaces to data filters.

#### Infrastructures Interface Requirements

This Big Data Framework Provider element provides all the resources necessary to host/run the activities of the other components of the Big Data system. Typically, these resources consist of some combination of physical resources, which may host/support similar virtual resources. The NBDRA needs interfaces that can be used to deal with the underlying infrastructure to address networking, computing, and storage.

#### Platforms Interface Requirements

As part of the NBDRA platforms, interfaces are needed that can address platform needs and services for data organization, data distribution, indexed storage, and file systems.

#### Processing Interface Requirements

The processing frameworks for Big Data provide the necessary infrastructure software to support implementation of applications that can deal with the volume, velocity, variety, and variability of data. Processing frameworks define how the computation and processing of the data is organized. Big Data applications rely on various platforms and technologies to meet the challenges of scalable data analytics and operation. A requirement is the ability to interface easily with computing services that offer specific analytics services, batch processing capabilities, interactive analysis, and data streaming.

#### Crosscutting Interface Requirements

Several crosscutting interface requirements within the Big Data Framework Provider include messaging, communication, and resource management. Often these services may be hidden from explicit interface use as they are part of larger systems that expose higher-level functionality through their interfaces. However, such interfaces may also be exposed on a lower level in case finer-grained control is needed. The need for such crosscutting interface requirements will be extracted from the *NBDIF: Volume 3, Use Cases and General Requirements* document.

#### Messaging/Communications Frameworks

Messaging and communications frameworks have their roots in the High Performance Computing environments long popular in the scientific and research communities. Messaging/Communications Frameworks were developed to provide application programming interfaces (APIs) for the reliable queuing, transmission, and receipt of data.

#### Resource Management Framework

As Big Data systems have evolved and become more complex, and as businesses work to leverage limited computation and storage resources to address a broader range of applications and business challenges, the requirement to effectively manage those resources has grown significantly. While tools for resource management and *elastic computing* have expanded and matured in response to the needs of cloud providers and virtualization technologies, Big Data introduces unique requirements for these tools. However, Big Data frameworks tend to fall more into a distributed computing paradigm, which presents additional challenges.

### Big Data Application Provider to Big Data Framework Provider Interface

The Big Data Framework Provider typically consists of one or more hierarchically organized instances of the components in the NBDRA IT value chain (fig. 2). There is no requirement that all instances at a given level in the hierarchy be of the same technology. In fact, most Big Data implementations are hybrids that combine multiple technology approaches to provide flexibility or meet the complete range of requirements, which are driven from the Big Data Application Provider.

# Specification Paradigm

This section summarizes the elementary specification paradigm.

## Hybrid and Multiple Frameworks

To avoid vendor lock-in, Big Data systems must be able to deal with hybrid and multiple frameworks. This is not only true for Clouds, containers, DevOps, but also for components of the NBDRA.

## Design by Resource-Oriented Architecture

A resource-oriented architecture represents a software architecture and programming paradigm for designing and developing software in the form of resources. It is often associated with *REpresentational State Transfer (REST)* interfaces. The resources are software components which can be reused in concrete reference implementations. The service specification is conducted with OpenAPI, allowing use to provide it in a very general form that is independent of the framework or computer language in which the services can be specified. Note that OpenAPI defines services in REST The previous version only specified the resource objects.

## Design by Example

To accelerate discussion among the NBD-PWG members, contributors to this document are encouraged to also provide the NBD-PWG with examples.

## Version Management

Previous work that shaped the current version of this volumes and are documented In GitHub (Laszewski 2019b) with prior versions of Volume 8 (Laszewski 2019d)(Laszewski et al. 2015) and Cloudmesh (Laszewski et al. 2017) in support of the NIST Big Data Architecture Framework (NIST, n.d.).

During the design phase and development period of each version of this document, enhancements are managed through GitHub and community contributions are managed via GitHub issues. This allows preservation of the history of this document. When a new version is ready, the version will be tagged in GitHub. Older versions will, through this process, also be available as historical documents. Discussions about objects in written form are communicated as GitHub issues.

## Interface Compliancy

Due to the easy extensibility of the resource objects specified in this document and their interfaces, it is important to introduce a terminology that allows the definition of interface compliancy. We define three levels of interface compliance as follows:

* **Full Compliance:** These are reference implementations that provide full compliance to the objects defined in this document. A version number is added to assure that the snapshot in time of the objects is associated with the version. A full complient framework implements all objects.
* **Partial Compliance:** These are reference implementations that provide partial compliance to the objects defined in this document. A version number will is added to assure that the snapshot in time of the objects is associated with the version. This reference implementation implements a partial list of the objects and interfaces. A document is to be added that specifies the differences to a full complient implementation.
* **Extended Compliance:** In addition to full and partial compliance additional resources can be identified while documenting additional resource objects and interfaces that are not included in the current specification. The extended complience document can lead to additional improvements of the current specification.

## Refernce implementations

Documents generated during a reference implementation can be forwarded to the Reference Architecture Subgroup for further discussion and for possible future modifications based on additional practical user feedback.

# Specification

The specifications in this section are provided through an automated document creation process using the actual OpenAPI specifications yaml files as the source. All OpenAPI specifications located in the cloudmesh/cloudmesh-nist/spec/ directory in GitHub (Laszewski 2019c).

Limitations of the current implementation are as follows. It is a demonstration that showcases the generation of a fully functioning REST service based on the specifications provided in this document. However, it is expected that scalability, distribution of services, and other advanced options need to be addressed based on application requirements.

## List of specifications

The following table lists the current set of resource objects that are defined in this draft. Additional objects are also available in GitHub (Laszewski 2019c).

tbl. 1 shows the list of currently included specification in this version of the document.

Table 1: Table 1: Specifications

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | Version | Date | Reference | Section |  |
| Alias | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/alias.yaml) | sec. 4.3.2 |  |
| Database | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/database.yaml) | sec. 4.4.3 |  |
| Default | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/default.yaml) | sec. 4.3.4 |  |
| File | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/filestore.yaml) | sec. 4.4.1 |  |
| Flavor | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/flavor.yaml) | sec. 4.6.2 |  |
| Image | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/image.yaml) | sec. 4.6.1 |  |
| Nic | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/nic.yaml) | sec. 4.6.5 |  |
| Organization | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/organization.yaml) | sec. 4.2.2 |  |
| Public Key Store | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/publickeystore.yaml) | sec. 4.2.4 |  |
| Replica | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/replica.yaml) | sec. 4.4.2 |  |
| Reservation | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/reservation.yaml) | sec. 4.9.1 |  |
| Scheduler | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/scheduler.yaml) | sec. 4.5.2 |  |
| Secgroup | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/secgroup.yaml) | sec. 4.6.4 |  |
| Timestamp | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/timestamp.yaml) | sec. 4.3.1 |  |
| User | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/user.yaml) | sec. 4.2.3 |  |
| Variables | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/variables.yaml) | sec. 4.3.3 |  |
| Virtual Cluster | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/virtualcluster.yaml) | sec. 4.5.1 |  |
| Virtual Directory | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/virtualdirectory.yaml) | sec. 4.4.4 |  |
| Virtual Machine | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/vm.yaml) | sec. 4.6.3 |  |
| Containers | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/containers.yaml) | sec. 4.7.1 |  |
| Deployment | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/deployment.yaml) | sec. 4.11.1 |  |
| Filter | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/filter.yaml) | sec. 4.10.2 |  |
| Microservice | 3.2.0 | 14-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/microservice.yaml) | sec. 4.8.1 |  |
| Stream | 3.2.0 | 08-06-2019 | [☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/stream.yaml) | sec. 4.10.1 |  |

fig. 3 shows the provider view of the specification resources.



Figure 3: Figure 3: Provider view

fig. 4 shows the resources view of the specification resources.



Figure 4: Figure 4: Resource view

## Identity

As part of services an identity often needs to be specified. In addition, such persons ( Internet2 Middleware Architecture Committee for Education 2016) are often part of groups. Thus, three important terms related to the identity are distinguished as follows:

* Organization: The information representing an Organization that manages a Big Data Service (sec. 4.2.2)
* Group: A group that a person may belong to that is important to define access to services (included in sec. 4.2.2)
* User: The information identifying the profile of a person (sec. 4.2.3)

### Authentication

Mechanisms are not included in this specification to manage authentication to external services. However, the working group has shown multiple solutions to this as part of cloudmesh. This includes the posibility of a

* *Local configuration file:* A configuration file is managed locally to allow access to the clouds. It is the designer’s responsibility not to expose such credentials.
* *Session based authentication:* No passwords are stored in the configuration file and access is granted on a per session basis where the password needs to be entered.
* *Service based authentication:* The authentication is delegated to an external process. One example here is Auth. The service that acts in behalf of the user needs to have access to the appropriate cloud provider credentials

An example for a configuration file is provided at (Laszewski 2019a).

### Organization

An important concept in many services is the management of a group of users in an organization. Within an organization we distinguish different groups of users and within a Group. Groups can be used to charachterize differnt roles users can fulfill. Users can belong to multiple groups. Such groups can also be used to specify access rights to services.

#### Schema Organization

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/organization.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the organization |
| users | array[User] | List of users |

#### Schema Group

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/group.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | The name of the group |
| description | string | The description of the group |
| users | array[string] | The user names that are members of the group |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /organization | Returns a list of organizations |
| put | /organization | Uploads an organization to the list of organizations |
| get | /organization/{name} | Returns the named organization |
| delete | /organization/{name} | Deletes the named organization |
| get | /organization/{name}/user | Returns all users of the organization |
| get | /organization/{name}/user/{username} | Returns the specific user of that organization |
| put | /organization/{name}/user/{username} | Updates or adds a user in the organization |
| delete | /organization/{name}/user/{username} | Delete an user in the organization |
| get | /organization/{name}/group/ | Returns all group names |
| get | /organization/{name}/group/{groupname} | Returns the specific group of that organization |
| put | /organization/{name}/group/{groupname} | Updates or adds a group in the organization |
| delete | /organization/{name}/group/{groupname} | Delete a group in the organization |
| put | /organization/{name}/group/{groupname}/{username} | Updates or adds a username to the group |

##### /organization

###### GET /organization

Returns a list of all organizations

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of organizations | array[[Organization](#organization)] |
| 400 | No Organizations found | Empty |
| 401 | Not authorized | Empty |

###### PUT /organization

Uploads an organization to the list of organizations

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Organization updated | Empty |
| 400 | Error updating organization. | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The organization to be uploaded | True | [Organization](#organization) |  |

##### /organization/{name}

###### GET /organization/{name}

Returns an organization by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Retruning the information of the organization | [Organization](#organization) |
| 400 | No organization found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named organization could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the organization | True | String |

###### DELETE /organization/{name}

Deletes an organization by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No organization found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named organization could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the organization | True | String |

##### /organization/{name}/user

###### GET /organization/{name}/user

Returns all users of the organization

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The organization | [Organization](#organization) |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the organization | True | String |

##### /organization/{name}/user/{username}

###### GET /organization/{name}/user/{username}

Returns the specific user of that organization

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The user | [User](#user) |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the organization | True | String |
| username | path | The username | True | String |

###### PUT /organization/{name}/user/{username}

Updates or adds a user in the organization

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | User added sucessfully | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the organization | True | String |
| username | path | The username | True | String |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The user to be uploaded | True | [User](#user) |  |

###### DELETE /organization/{name}/user/{username}

Delete an user in the organization

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the organization | True | String |
| username | path | The username | True | String |

##### /organization/{name}/group/

###### GET /organization/{name}/group/

Returns all group names

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the group | array[String] |
| 400 | No group found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named group could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the organization | True | String |

##### /organization/{name}/group/{groupname}

###### GET /organization/{name}/group/{groupname}

Returns the specific group of that organization

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The group | [Group](#group) |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the organization | True | String |
| groupname | path | The groupname | True | String |

###### PUT /organization/{name}/group/{groupname}

Updates or adds a group in the organization

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Group added sucessfully | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the group | True | String |
| groupname | path | The group name | True | String |

###### DELETE /organization/{name}/group/{groupname}

Delete a group in the organization

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the organization | True | String |
| groupname | path | The group name | True | String |

##### /organization/{name}/group/{groupname}/{username}

###### PUT /organization/{name}/group/{groupname}/{username}

Updates or adds a username to the group

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Group added sucessfully | Empty |
| 401 | Not authorized | Empty |
| 404 | The organization or group could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the group | True | String |
| groupname | path | The group name | True | String |
| username | path | The username | True | String |

#### organization.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: Organization  
 description: |-  
  
 An important concept in many services is the management of a group  
 of users in an organization. Within an organization we distinguish  
 different groups of users and within a Group. Groups can be used to  
 charachterize differnt roles users can fulfill. Users can belong to  
 multiple groups. Such groups can also be used to specify access rights to  
 services.  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /organization:  
 get:  
 tags:  
 - Organization  
 summary: Returns a list of organizations  
 description: Returns a list of all organizations  
 operationId: cloudmesh.organization.list  
 responses:  
 '200':  
 description: The list of organizations  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Organization'  
 '400':  
 description: No Organizations found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Organization  
 summary: Uploads an organization to the list of organizations  
 description: Uploads an organization to the list of organizations  
 operationId: cloudmesh.organization.add  
 requestBody:  
 description: The organization to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Organization'  
 responses:  
 '200':  
 description: Organization updated  
 '400':  
 description: Error updating organization.  
 '401':  
 description: Not authorized  
 /organization/{name}:  
 get:  
 tags:  
 - Organization  
 summary: Returns the named organization  
 description: Returns an organization by name  
 operationId: cloudmesh.organization.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the organization  
 responses:  
 '200':  
 description: Retruning the information of the organization  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Organization'  
 '400':  
 description: No organization found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named organization could not be found  
 delete:  
 tags:  
 - Organization  
 summary: Deletes the named organization  
 description: Deletes an organization by name  
 operationId: cloudmesh.organization.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the organization  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No organization found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named organization could not be found  
 /organization/{name}/user:  
 get:  
 tags:  
 - Organization  
 summary: Returns all users of the organization  
 description: Returns all users of the organization  
 operationId: cloudmesh.organization.user.list  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the organization  
 responses:  
 '200':  
 description: The organization  
 content:  
 application/json:  
 schema:  
 $ref: "#/components/schemas/Organization"  
 '401':  
 description: Not authorized  
 /organization/{name}/user/{username}:  
 get:  
 tags:  
 - Organization  
 summary: Returns the specific user of that organization  
 description: Returns the specific user of that organization  
 operationId: cloudmesh.organization.user.get\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the organization  
 - name: username  
 description: The username  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: The user  
 content:  
 application/json:  
 schema:  
 $ref: "user.yaml#/components/schemas/User"  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Organization  
 summary: Updates or adds a user in the organization  
 description: Updates or adds a user in the organization  
 operationId: cloudmesh.organization.user.add  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the organization  
 - name: username  
 description: The username  
 in: path  
 required: true  
 schema:  
 type: string  
 requestBody:  
 description: The user to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: 'user.yaml#/components/schemas/User'  
 responses:  
 '200':  
 description: User added sucessfully  
 '401':  
 description: Not authorized  
 delete:  
 tags:  
 - Organization  
 summary: Delete an user in the organization  
 description: Delete an user in the organization  
 operationId: cloudmesh.organization.user.delete  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the organization  
 - name: username  
 description: The username  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Deletion successful  
 '401':  
 description: Not authorized  
 /organization/{name}/group/:  
 get:  
 tags:  
 - Organization  
 summary: Returns all group names  
 description: Returns all group names  
 operationId: cloudmesh.organization.group.list  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the organization  
 responses:  
 '200':  
 description: Returning the information of the group  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 type: string  
 '400':  
 description: No group found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named group could not be found  
 /organization/{name}/group/{groupname}:  
 get:  
 tags:  
 - Organization  
 summary: Returns the specific group of that organization  
 description: Returns the specific group of that organization  
 operationId: cloudmesh.organization.group.get\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the organization  
 - name: groupname  
 description: The groupname  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: The group  
 content:  
 application/json:  
 schema:  
 $ref: "#/components/schemas/Group"  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Organization  
 summary: Updates or adds a group in the organization  
 description: Updates or adds a group in the organization  
 operationId: cloudmesh.organization.group.add  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the group  
 - name: groupname  
 description: The group name  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Group added sucessfully  
 '401':  
 description: Not authorized  
 delete:  
 tags:  
 - Organization  
 summary: Delete a group in the organization  
 description: Delete a group in the organization  
 operationId: cloudmesh.organization.greop.delete  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the organization  
 - name: groupname  
 description: The group name  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Deletion successful  
 '401':  
 description: Not authorized  
 /organization/{name}/group/{groupname}/{username}:  
 put:  
 tags:  
 - Organization  
 summary: Updates or adds a username to the group  
 description: Updates or adds a username to the group  
 operationId: cloudmesh.organization.group.user.add  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the group  
 - name: groupname  
 description: The group name  
 in: path  
 required: true  
 schema:  
 type: string  
 - name: username  
 description: The username  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Group added sucessfully  
 '401':  
 description: Not authorized  
 '404':  
 description: The organization or group could not be found  
components:  
 schemas:  
 Organization:  
 type: object  
 properties:  
 name:  
 description: Name of the organization  
 type: string  
 users:  
 description: List of users  
 type: array  
 items:  
 $ref: "user.yaml#/components/schemas/User"  
 Group:  
 type: object  
 description: The groups  
 properties:  
 name:  
 type: string  
 description: The name of the group  
 description:  
 type: string  
 description: The description of the group  
 users:  
 description: The user names that are members of the group  
 type: array  
 items:  
 type: string

### User

Services need to specify which users have access to them. User information can be reused in other services and organized in a virtual organization. A user can be added to a named list of users within this organization. A group associated with the user can be used to augment users to be part of one or more groups.

#### Schema User

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/user.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| username | string | The unique username associated with the user |
| firstname | string | The firstname of the user |
| lastname | string | The lastname of the user |
| email | string | The email of the user |
| comment | string | A comment regarding the user |
| publickey | string | The public key of the user |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /user | Returns a list of users |
| put | /user | Uploads a user to the list of users |
| get | /user/{name} | Returns the named user |
| delete | /user/{name} | Deletes the named user |

##### /user

###### GET /user

Returns a list of all users

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of users | array[[User](#user)] |
| 400 | No users found | Empty |
| 401 | Not authorized | Empty |

###### PUT /user

Uploads a user to the list of users

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | User updated | Empty |
| 400 | Error updating user. | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The user to be uploaded | True | [User](#user) |  |

##### /user/{name}

###### GET /user/{name}

Returns an user by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the user | [User](#user) |
| 400 | No user found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named user could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the user | True | String |

###### DELETE /user/{name}

Deletes an user by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No user found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named user could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the user | True | String |

#### user.yaml

openapi: "3.0.2"  
info:  
 version: "3.2.0"  
 x-date: 14-06-2019  
 x-status: defined  
 title: User  
 description: |-  
  
 Services need to specify which users have access to them. User  
 information can be reused in other services and organized in a virtual  
 organization. A user can be added to a named list of users within this  
 organization. A group associated with the user can be used to augment  
 users to be part of one or more groups.  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: Cloudmesh User  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /user:  
 get:  
 tags:  
 - User  
 summary: Returns a list of users  
 description: Returns a list of all users  
 operationId: cloudmesh.user.list  
 responses:  
 '200':  
 description: The list of users  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/User'  
 '400':  
 description: No users found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - User  
 summary: Uploads a user to the list of users  
 description: Uploads a user to the list of users  
 operationId: cloudmesh.user.add  
 requestBody:  
 description: The user to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/User'  
 responses:  
 '200':  
 description: User updated  
 '400':  
 description: Error updating user.  
 '401':  
 description: Not authorized  
 /user/{name}:  
 get:  
 tags:  
 - User  
 summary: Returns the named user  
 description: Returns an user by name  
 operationId: cloudmesh.user.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the user  
 responses:  
 '200':  
 description: Returning the information of the user  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/User'  
 '400':  
 description: No user found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named user could not be found  
 delete:  
 tags:  
 - User  
 summary: Deletes the named user  
 description: Deletes an user by name  
 operationId: cloudmesh.user.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the user  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No user found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named user could not be found  
components:  
 schemas:  
 User:  
 type: object  
 properties:  
 username:  
 type: string  
 description: The unique username associated with the user  
 firstname:  
 type: string  
 description: The firstname of the user  
 lastname:  
 type: string  
 description: The lastname of the user  
 email:  
 type: string  
 description: The email of the user  
 comment:  
 type: string  
 description: A comment regarding the user  
 publickey:  
 type: string  
 description: The public key of the user

### Public Key Store

Many services and frameworks use Secure Shell (SSH) keys to authenticate. This service allows the convenient storage of the public keys.

#### Schema Key

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/key.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | The name of the public key |
| value | string | The value of the public key |
| kind | string | The key kind such as rsa, dsa |
| group | string | An optional group name allowing to group keys to create custom key groups within the public key store |
| comment | string | A comment for the public key |
| uri | string | The uri of the public key if any |
| fingerprint | string | The fingerprint of the public key |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /key | Returns a list of keys |
| put | /key | Set a key |
| get | /key/{name} | Returns the named key |
| delete | /key/{name} | Deletes the named key |

##### /key

###### GET /key

Returns a list of all keys

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of keys | array[[Key](#key)] |
| 400 | No key found | Empty |
| 401 | Not authorized | Empty |

###### PUT /key

Sets the named key

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Key updated | Empty |
| 400 | Error updating key. | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The new key to create | True | [Key](#key) |  |

##### /key/{name}

###### GET /key/{name}

Returns a key by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the key | [Key](#key) |
| 400 | No key found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named key could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the key | True | String |

###### DELETE /key/{name}

Deletes a key by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No key found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named key could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the key | True | String |

#### publickeystore.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: Public Key Store  
 description: |-  
   
 Many services and frameworks use Secure Shell (SSH) keys to  
 authenticate. This service allows the convenient storage of the  
 public keys.  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /key:  
 get:  
 tags:  
 - Key  
 summary: Returns a list of keys  
 description: Returns a list of all keys  
 operationId: cloudmesh.key.list  
 responses:  
 '200':  
 description: The list of keys  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Key'  
 '400':  
 description: No key found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Key  
 summary: Set a key  
 description: Sets the named key  
 operationId: cloudmesh.key.add  
 requestBody:  
 description: The new key to create  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Key'  
 responses:  
 '200':  
 description: Key updated  
 '400':  
 description: Error updating key.  
 '401':  
 description: Not authorized  
 /key/{name}:  
 get:  
 tags:  
 - Key  
 summary: Returns the named key  
 description: Returns a key by name  
 operationId: cloudmesh.key.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the key  
 responses:  
 '200':  
 description: Returning the information of the key  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Key'  
 '400':  
 description: No key found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named key could not be found  
 delete:  
 tags:  
 - Key  
 summary: Deletes the named key  
 description: Deletes a key by name  
 operationId: cloudmesh.key.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the key  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No key found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named key could not be found  
components:  
 schemas:  
 Key:  
 type: object  
 description: the public key  
 properties:  
 name:  
 type: string  
 description: The name of the public key  
 value:  
 type: string  
 description: The value of the public key  
 kind:  
 type: string  
 description: The key kind such as rsa, dsa  
 group:  
 type: string  
 description: An optional group name allowing to group keys to create  
 custom key groups within the public key store  
 comment:  
 type: string  
 description: A comment for the public key  
 uri:  
 type: string  
 description: The uri of the public key if any  
 fingerprint:  
 type: string  
 description: The fingerprint of the public key

## General Resources

### Timestamp

Data often needs to be time stamped to indicate when it has been accessed, created, or modified. All objects defined in this document will have, in their final version, a timestamp. The date-time string is defined in [RFC3339](https://xml2rfc.ietf.org/public/rfc/html/rfc3339.html#anchor14).

#### Schema Timestamp

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/timestamp.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| accessed | string | The time stamp when the object was last accessed |
| created | string | The time stamp when the object was created |
| modified | string | The time stamp when the object was modified |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /timestamp | Returns the time |

##### /timestamp

###### GET /timestamp

Returns the time to be used in a timestamp

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The current time and date | string |

#### timestamp.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: Timestamp  
 description: |-  
   
 Data often needs to be time stamped to indicate when it has been  
 accessed, created, or modified. All objects defined in this  
 document will have, in their final version, a timestamp.  
 The date-time string is defined in  
 [RFC3339](https://xml2rfc.ietf.org/public/rfc/html/rfc3339.html#anchor14).  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /timestamp:  
 get:  
 summary: Returns the time  
 description: Returns the time to be used in a timestamp  
 responses:  
 '200':  
 description: The current time and date  
 content:  
 application/json:  
 schema:  
 type: string  
 example: 1985-04-12T23:20:50.52Z  
components:  
 schemas:  
 Timestamp:  
 type: object  
 description: the timestamp  
 properties:  
 accessed:  
 type: string  
 format: date-time  
 description: The time stamp when the object was last accessed  
 example: 1985-04-12T23:20:50.52Z  
 created:  
 type: string  
 format: date-time  
 description: The time stamp when the object was created  
 example: 1985-04-12T23:20:50.52Z  
 modified:  
 type: string  
 format: date-time  
 description: The time stamp when the object was modified  
 example: 1985-04-12T23:20:50.52Z

### Alias

Often a user has the desire to create a custom name for an object. An alias allows to do that while while assosication auser defined name or *alias* to a previouly used name. The aliases could be shared with other users. A name could have one or more aliases.

#### Schema Alias

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/alias.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | The name of the alias |
| source | string | The original object name |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /alias | Returns a list of aliases |
| put | /alias | Set an alias |
| get | /alias/{name} | Returns the named alias |
| delete | /alias/{name} | Deletes the named alias |

##### /alias

###### GET /alias

Returns a list of all aliases

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of aliasses | array[[Alias](#alias)] |
| 400 | No alias found | Empty |
| 401 | Not authorized | Empty |

###### PUT /alias

Sets the named alias

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Alias updated | Empty |
| 400 | Error updating alias | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The new alias to create | True | [Alias](#alias) |  |

##### /alias/{name}

###### GET /alias/{name}

Returns an alias by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the alias | [Alias](#alias) |
| 400 | No alias found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named alias could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the alias | True | String |

###### DELETE /alias/{name}

Deletes an alias by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No alias found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named alias could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the alias | True | String |

#### alias.yaml

openapi: '3.0.2'  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: Alias  
 description: |-  
  
 Often a user has the desire to create a custom name for an object. An  
 alias allows to do that while while assosication auser defined name or  
 \*alias\* to a previouly used name. The aliases could be shared with other  
 users. A name could have one or more aliases.  
  
 termsOfService: 'https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt'  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /alias:  
 get:  
 tags:  
 - Alias  
 summary: Returns a list of aliases  
 description: Returns a list of all aliases  
 operationId: cloudmesh.alias.list  
 responses:  
 '200':  
 description: The list of aliasses  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Alias'  
 '400':  
 description: No alias found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Alias  
 summary: Set an alias  
 description: Sets the named alias  
 operationId: cloudmesh.alias.add  
 requestBody:  
 description: The new alias to create  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Alias'  
 responses:  
 '200':  
 description: Alias updated  
 '400':  
 description: Error updating alias  
 '401':  
 description: Not authorized  
 /alias/{name}:  
 get:  
 tags:  
 - Alias  
 summary: Returns the named alias  
 description: Returns an alias by name  
 operationId: cloudmesh.alias.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the alias  
 responses:  
 '200':  
 description: Returning the information of the alias  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Alias'  
 '400':  
 description: No alias found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named alias could not be found  
 delete:  
 tags:  
 - Alias  
 summary: Deletes the named alias  
 description: Deletes an alias by name  
 operationId: cloudmesh.alias.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the alias  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No alias found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named alias could not be found  
components:  
 schemas:  
 Alias:  
 type: object  
 description: the alias  
 properties:  
 name:  
 type: string  
 description: The name of the alias  
 source:  
 type: string  
 description: The original object name

### Variables

Variables are a simple string key value storage to store simple values. Each variable can have a datatype, so that it can be used for serialization into other formats. Internally they are storred as strings.

#### Schema Variable

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/variable.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the variable |
| value | string | Type of the variable |
| description | string | A description of the variable |
| datatype | string | The data type of the variable which can be used for serialization |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /variable | Returns a the variables |
| put | /variable | Set the value of a variable |
| get | /variable/{name} | Returns the named variable |
| delete | /variable/{name} | Deletes the named variable |

##### /variable

###### GET /variable

Returns the variables

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of variables | array[[Variable](#variable)] |
| 400 | No variable found | Empty |

###### PUT /variable

Set the value of the named variable

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Variable updated | Empty |
| 400 | Error updating variable | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The variable and its value | True | [Variable](#variable) |  |

##### /variable/{name}

###### GET /variable/{name}

Returns the named variable

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the variable | [Variable](#variable) |
| 400 | No variable found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named variable could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the variable | True | String |

###### DELETE /variable/{name}

Deletes a variable by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No variable found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named variable could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the variable | True | String |

#### variables.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: Variables  
 description: |-  
   
 Variables are a simple string key value storage to store simple  
 values. Each variable can have a datatype, so that it can be used for  
 serialization into other formats. Internally they are storred as strings.  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /variable:  
 get:  
 tags:  
 - Variable  
 summary: Returns a the variables  
 description: Returns the variables  
 operationId: cloudmesh.variable.list  
 responses:  
 '200':  
 description: The list of variables  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Variable'  
 '400':  
 description: No variable found  
 put:  
 tags:  
 - Variable  
 summary: Set the value of a variable  
 description: Set the value of the named variable  
 operationId: cloudmesh.variable.add  
 requestBody:  
 description: The variable and its value  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Variable'  
 responses:  
 '200':  
 description: Variable updated  
 '400':  
 description: Error updating variable  
 '401':  
 description: Not authorized  
 /variable/{name}:  
 get:  
 tags:  
 - Variable  
 summary: Returns the named variable  
 description: Returns the named variable  
 operationId: cloudmesh.variable.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the variable  
 responses:  
 '200':  
 description: Returning the information of the variable  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Variable'  
 '400':  
 description: No variable found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named variable could not be found  
 delete:  
 tags:  
 - Variable  
 summary: Deletes the named variable  
 description: Deletes a variable by name  
 operationId: cloudmesh.variable.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the variable  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No variable found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named variable could not be found  
components:  
 schemas:  
 Variable:  
 type: object  
 description: the variables  
 properties:  
 name:  
 type: string  
 description: Name of the variable  
 value:  
 type: string  
 description: Type of the variable  
 description:  
 type: string  
 description: A description of the variable  
 datatype:  
 type: string  
 description: The data type of the variable which can be used for  
 serialization

### Default

A default is a special variable that has a context associated with it. This allows one to define values that can be easily retrieved based on the associated context. For example, a default could be the image name for a cloud where the context is defined by the cloud name. In addition to the context, the service name is also specified since a service could have multiple contexts. To be able to define the kind of service there is also a kind attribute.

#### Schema Default

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/default.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | The name of the default |
| value | string | The value of the default |
| context | string | The context of the default |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /default | Returns a list of defaults |
| put | /default | Set a default |
| get | /default/{name} | Returns the named default |
| delete | /default/{name} | Deletes the named default |

##### /default

###### GET /default

Returns a list of all defaults

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of defaults | array[[Default](#default)] |
| 400 | No default found | Empty |

###### PUT /default

Sets the named default

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Default updated | Empty |
| 400 | Error updating default. | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The new default to create | True | [Default](#default) |  |

##### /default/{name}

###### GET /default/{name}

Returns a default by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the default | [Default](#default) |
| 400 | No default found | Empty |
| 404 | The named default could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the default | True | String |

###### DELETE /default/{name}

Deletes a default by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No default found | Empty |
| 404 | The named default could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the default | True | String |

#### default.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: Default  
 description: |-  
   
 A default is a special variable that has a context associated with  
 it. This allows one to define values that can be easily retrieved  
 based on the associated context. For example, a default could be  
 the image name for a cloud where the context is defined by the  
 cloud name. In addition to the context, the service name is also specified  
 since a service could have multiple contexts. To be able to  
 define the kind of service there is also a kind attribute.  
   
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /default:  
 get:  
 tags:  
 - Default  
 summary: Returns a list of defaults  
 description: Returns a list of all defaults  
 operationId: cloudmesh.default.list  
 responses:  
 '200':  
 description: The list of defaults  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Default'  
 '400':  
 description: No default found  
 put:  
 tags:  
 - Default  
 summary: Set a default  
 description: Sets the named default  
 operationId: cloudmesh.default.add  
 requestBody:  
 description: The new default to create  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Default'  
 responses:  
 '200':  
 description: Default updated  
 '400':  
 description: Error updating default.  
 /default/{name}:  
 get:  
 tags:  
 - Default  
 summary: Returns the named default  
 description: Returns a default by name  
 operationId: cloudmesh.default.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the default  
 responses:  
 '200':  
 description: Returning the information of the default  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Default'  
 '400':  
 description: No default found  
 '404':  
 description: The named default could not be found  
 delete:  
 tags:  
 - Default  
 summary: Deletes the named default  
 description: Deletes a default by name  
 operationId: cloudmesh.default.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the default  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No default found  
 '404':  
 description: The named default could not be found  
components:  
 schemas:  
 Default:  
 type: object  
 description: the defaults  
 properties:  
 name:  
 type: string  
 description: The name of the default  
 example: "image"  
 value:  
 type: string  
 description: The value of the default  
 example: "m1.medium"  
 context:  
 type: string  
 description: The context of the default  
 example: "cloud.vm.flavor"

## Data Management

### Filestore

A file store is a resource allowing storage of data as a traditional file. Instead of using the name *filestore* we simply use the name *file*. A file store can contanin any number of files with additional attributes describing the file. An file store is typically located on the same cloud services. This contrasts virtual directories that are just pointers to files, which could include files located in file stores.

#### Schema File

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/file.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | The name of the file |
| endpoint | string | The location of the file |
| checksum | string | The checksum of the file |
| size | integer | The size of the file in byte |
| content | string | the content of the file |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /file | Returns a list of files in the file store |
| put | /file | Uploads a file to the list of files in the file store |
| get | /file/{name} | Returns the named file in the file store |
| delete | /file/{name} | Deletes the named file in the file store |

##### /file

###### GET /file

Returns a list of all files

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of files | array[[File](#file)] |
| 400 | No files found | Empty |

###### PUT /file

Uploads a file to the list of files in the file store

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | File updated | Empty |
| 400 | Error updating file. | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The file to be uploaded | True | [File](#file) |  |

##### /file/{name}

###### GET /file/{name}

Returns an file by name in the file store

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the file store | [File](#file) |
| 400 | No file found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named file could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the file | True | String |

###### DELETE /file/{name}

Deletes an file by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No file found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named file could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the file | True | String |

#### filestore.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: File  
 description: |-  
   
 A file store is a resource allowing storage of data as a traditional file.  
 Instead of using the name \*filestore\* we simply use the name \*file\*.  
 A file store can contanin any number of files with additional attributes  
 describing the file. An file store is typically located on the same cloud  
 services. This contrasts virtual directories that are just pointers to  
 files, which could include files located in file stores.  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /file:  
 get:  
 tags:  
 - File  
 summary: Returns a list of files in the file store  
 description: Returns a list of all files  
 operationId: cloudmesh.file.list  
 responses:  
 '200':  
 description: The list of files  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/File'  
 '400':  
 description: No files found  
 put:  
 tags:  
 - File  
 summary: Uploads a file to the list of files in the file store  
 description: Uploads a file to the list of files in the file store  
 operationId: cloudmesh.file.add  
 requestBody:  
 description: The file to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/File'  
  
  
 responses:  
 '200':  
 description: File updated  
 '400':  
 description: Error updating file.  
  
 /file/{name}:  
 get:  
 tags:  
 - File  
 summary: Returns the named file in the file store  
 description: Returns an file by name in the file store  
 operationId: cloudmesh.file.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the file  
 responses:  
 '200':  
 description: Returning the information of the file store  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/File'  
 '400':  
 description: No file found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named file could not be found  
 delete:  
 tags:  
 - File  
 summary: Deletes the named file in the file store  
 description: Deletes an file by name  
 operationId: cloudmesh.file.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the file  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No file found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named file could not be found  
components:  
 schemas:  
 File:  
 type: object  
 description: an object representing a file  
 properties:  
 name:  
 type: string  
 description: The name of the file  
 endpoint:  
 type: string  
 description: The location of the file  
 checksum:  
 type: string  
 description: The checksum of the file  
 size:  
 type: integer  
 description: The size of the file in byte  
 content:  
 type: string  
 format: binary  
 description: the content of the file

### Replica

In many distributed systems, it is important that a file can be replicated among different systems to provide faster access. It is important to provide a mechanism to trace the pedigree of the file while pointing to its original source. A replica will point to a file in a file store and store the contents in the file store instead of the replica. The replica is just a pointer.

#### Schema Replica

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/replica.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | The name of the replica |
| filename | string | The original filename |
| endpoint | string | The location of the file |
| checksum | string | The checksum of the file |
| size | integer | The size of the file in byte |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /replica | Returns a list of replicas |
| put | /replica | Uploads a replica to the list of replicas |
| get | /replica/{name} | Returns the named replica |
| delete | /replica/{name} | Deletes the named replica |

##### /replica

###### GET /replica

Returns a list of all replicas

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of replicas | array[[Replica](#replica)] |
| 400 | No replicas found | Empty |
| 401 | Not authorized | Empty |

###### PUT /replica

Uploads a replica to the list of replicas

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Replica updated | Empty |
| 400 | Error updating replica. | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The replica to be uploaded | True | [Replica](#replica) |  |

##### /replica/{name}

###### GET /replica/{name}

Returns an replica by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the replica | [Replica](#replica) |
| 400 | No replica found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named replica could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the replica | True | String |

###### DELETE /replica/{name}

Deletes an replica by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No replica found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named replica could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the replica | True | String |

#### replica.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: TODO  
 title: Replica  
 description: |-  
   
 In many distributed systems, it is important that a file can be  
 replicated among different systems to provide faster access. It is  
 important to provide a mechanism to trace the pedigree of the file  
 while pointing to its original source. A replica will point to a file in  
 a file store and store the contents in the file store instead of the  
 replica. The replica is just a pointer.  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /replica:  
 get:  
 tags:  
 - Replica  
 summary: Returns a list of replicas  
 description: Returns a list of all replicas  
 operationId: cloudmesh.replica.list  
 responses:  
 '200':  
 description: The list of replicas  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Replica'  
 '400':  
 description: No replicas found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Replica  
 summary: Uploads a replica to the list of replicas  
 description: Uploads a replica to the list of replicas  
 operationId: cloudmesh.replica.add  
 requestBody:  
 description: The replica to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Replica'  
 responses:  
 '200':  
 description: Replica updated  
 '400':  
 description: Error updating replica.  
 '401':  
 description: Not authorized  
 /replica/{name}:  
 get:  
 tags:  
 - Replica  
 summary: Returns the named replica  
 description: Returns an replica by name  
 operationId: cloudmesh.replica.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the replica  
 responses:  
 '200':  
 description: Returning the information of the replica  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Replica'  
 '400':  
 description: No replica found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named replica could not be found  
 delete:  
 tags:  
 - Replica  
 summary: Deletes the named replica  
 description: Deletes an replica by name  
 operationId: cloudmesh.replica.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the replica  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No replica found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named replica could not be found  
components:  
 schemas:  
 Replica:  
 type: object  
 description: An entry representing a file replica record  
 properties:  
 name:  
 type: string  
 description: The name of the replica  
 filename:  
 type: string  
 description: The original filename  
 endpoint:  
 type: string  
 description: The location of the file  
 checksum:  
 type: string  
 description: The checksum of the file  
 size:  
 type: integer  
 description: The size of the file in byte

### Database

The database specification allows to register a database and perform elementary operations to use this database. We distinguish actions related to the registration, the adding of a schema, the insertion of data and the query of data. The data base is defined by a name an endpoint (e.g., host, port), and a protocol used (e.g., SQL, MongoDB, graphgl, and others).

#### Schema Database

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/database.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the database |
| description | string | Description of the database |
| endpoint | string | Endpoint of the database |
| kind | string | the kind of the database |

#### Schema Definition

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/definition.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the database |
| description | string | Description of the database |
| kind | string | The kind of the definition |
| schema | string | The schema associated with the table or collection |

#### Schema Record

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/record.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| status | string | The status of the reurn |
| result | string | The result of the quesry in json string format |

#### Schema Query

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/query.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| status | string | The query string |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /database | Returns all databases |
| delete | /database | Deletes a database from the list of databases |
| get | /database/{name} | Get the list of tables or collections for the database |
| put | /database/{name} | Upload a news table or collection |
| get | /database/{name}/{table\_or\_collection} | Query the table or collection |
| put | /database/{name}/{table\_or\_collection} | add data to the table or collection |

##### /database

###### GET /database

Returns all databases

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | List of databases | array[[Database](#database)] |
| 400 | No database found | Empty |
| 401 | Not authorized | Empty |
| 404 | Named database not found | Empty |

###### DELETE /database

Deletes a database from the list of databases

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No database found | Empty |
| 401 | Not authorized | Empty |
| 404 | Named database not found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | Name of the database | True | String |

##### /database/{name}

###### GET /database/{name}

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | successfully returned the list | array[[Definition](#definition)] |
| 400 | No database found | Empty |
| 401 | Not authorized | Empty |
| 404 | Named database not found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | Name of the collections or tables | True | String |

###### PUT /database/{name}

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | successfully returned the list | [Definition](#definition) |
| 400 | No database found | Empty |
| 401 | Not authorized | Empty |
| 404 | Named database not found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | Name of the collections or tables | True | String |

##### /database/{name}/{table\_or\_collection}

###### GET /database/{name}/{table\_or\_collection}

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Successfull query | array[[Record](#record)] |
| 400 | No database found | Empty |
| 401 | Not authorized | Empty |
| 404 | Named database not found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | Name of the database | True | String |
| table\_or\_collection | path | Name of the table or collection | True | String |
| query | path | Database Query | True | [Query](#query) |

###### PUT /database/{name}/{table\_or\_collection}

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | successfully uploaded | [Record](#record) |
| 400 | No database found | Empty |
| 401 | Not authorized | Empty |
| 404 | Named database not found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | Name of the database | True | String |
| table\_or\_collection | path | Name of the table or collection | True | String |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | Record to be uploaded | True | [Record](#record) |  |

#### database.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: TODO  
 title: Database  
 description: |-  
  
 The database specification allows to register a database and perform  
 elementary operations to use this database. We distinguish actions  
 related to the registration, the adding of a schema, the insertion of  
 data and the query of data. The data base is defined by a name an endpoint  
 (e.g., host, port), and a protocol used (e.g., SQL, MongoDB, graphgl, and  
 others).  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /database:  
 get:  
 tags:  
 - "Database Registry"  
 summary: Returns all databases  
 description: Returns all databases  
 operationId: cloudmesh.database.get  
 responses:  
 '200':  
 description: List of databases  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: "#/components/schemas/Database"  
 '400':  
 description: No database found  
 '401':  
 description: Not authorized  
 '404':  
 description: Named database not found  
 delete:  
 tags:  
 - "Database Registry"  
 summary: Deletes a database from the list of databases  
 description: Deletes a database from the list of databases  
 operationId: cloudmesh.database.delete  
 parameters:  
 - name: name  
 description: Name of the database  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No database found  
 '401':  
 description: Not authorized  
 '404':  
 description: Named database not found  
 /database/{name}:  
 get:  
 tags:  
 - "Database Definition"  
 summary: "Get the list of tables or collections for the database"  
 description: ""  
 operationId: "cloudmesh.database.get.collections\_or\_tables"  
 parameters:  
 - name: name  
 description: Name of the collections or tables  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: "successfully returned the list"  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: "#/components/schemas/Definition"  
 '400':  
 description: No database found  
 '401':  
 description: Not authorized  
 '404':  
 description: Named database not found  
 put:  
 tags:  
 - "Database Definition"  
 summary: "Upload a news table or collection"  
 description: ""  
 operationId: "cloudmesh.database.put.collections\_or\_tables"  
 parameters:  
 - name: name  
 description: Name of the collections or tables  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: "successfully returned the list"  
 content:  
 application/json:  
 schema:  
 $ref: "#/components/schemas/Definition"  
 '400':  
 description: No database found  
 '401':  
 description: Not authorized  
 '404':  
 description: Named database not found  
 /database/{name}/{table\_or\_collection}:  
 get:  
 tags:  
 - "Database Data Interaction"  
 summary: "Query the table or collection"  
 description: ""  
 operationId: "cloudmesh.database.data.get"  
 parameters:  
 - name: name  
 description: Name of the database  
 in: path  
 required: true  
 schema:  
 type: string  
 - name: table\_or\_collection  
 description: Name of the table or collection  
 in: path  
 required: true  
 schema:  
 type: string  
 - name: query  
 description: Database Query  
 in: path  
 required: true  
 schema:  
 $ref: '#/components/schemas/Query'  
 responses:  
 '200':  
 description: Successfull query  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: "#/components/schemas/Record"  
 '400':  
 description: No database found  
 '401':  
 description: Not authorized  
 '404':  
 description: Named database not found  
 put:  
 tags:  
 - "Database Data Interaction"  
 summary: "add data to the table or collection"  
 description: ""  
 operationId: "cloudmesh.database.data.put"  
 parameters:  
 - name: name  
 description: Name of the database  
 in: path  
 required: true  
 schema:  
 type: string  
 - name: table\_or\_collection  
 description: Name of the table or collection  
 in: path  
 required: true  
 schema:  
 type: string  
 requestBody:  
 description: Record to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: "#/components/schemas/Record"  
 responses:  
 '200':  
 description: "successfully uploaded"  
 content:  
 application/json:  
 schema:  
 $ref: "#/components/schemas/Record"  
 '400':  
 description: No database found  
 '401':  
 description: Not authorized  
 '404':  
 description: Named database not found  
components:  
 schemas:  
 Database:  
 type: object  
 description: Defines a database object as an entry  
 properties:  
 name:  
 type: string  
 description: Name of the database  
 description:  
 type: string  
 description: Description of the database  
 endpoint:  
 type: string  
 description: Endpoint of the database  
 kind:  
 type: string  
 description: the kind of the database  
 Definition:  
 type: object  
 description: Defines a database collection or Table  
 properties:  
 name:  
 type: string  
 description: Name of the database  
 description:  
 type: string  
 description: Description of the database  
 kind:  
 type: string  
 description: The kind of the definition  
 schema:  
 type: string  
 description: The schema associated with the table or collection  
 Record:  
 type: object  
 description: The result of a query  
 properties:  
 status:  
 type: string  
 description: The status of the reurn  
 result:  
 type: string  
 description: The result of the quesry in json string format  
 Query:  
 type: object  
 description: The query  
 properties:  
 status:  
 type: string  
 description: The query string

### Virtual Directory

A virtual directory is a collection of files, replicas, streams or other virtual directories.

#### Schema Virtualdirectory

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/virtualdirectory.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | The name of the virtual directory |
| description | string | Description of the virtual directory |
| host | string | Remote host of the virtual directory |
| location | string | Remote location, e.g., a directory with full path on a host |
| protocol | string | Access protocol (e.g. HTTP, FTP, SSH, etc.) |
| credential | object | Credential to access |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /virtualdirectory | Returns a list of virtual directories |
| put | /virtualdirectory | Uploads a virtual directory to the list of virtual directories |
| get | /virtualdirectory/{name} | Returns the named virtual directory |
| delete | /virtualdirectory/{name} | Deletes the named virtual directory |
| get | /virtualdirectory/{name}/{filename} | Returns the specific file of that virtual directory |
| put | /virtualdirectory/{name}/{filename} | Updates or adds a virtual file in the virtual directory |
| delete | /virtualdirectory/{name}/{filename} | Delete an user in the virtual directory |

##### /virtualdirectory

###### GET /virtualdirectory

Returns a list of all virtual directories

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of virtual directories | array[[Virtualdirectory](#virtualdirectory)] |
| 400 | No virtual directories found | Empty |
| 401 | Not authorized | Empty |

###### PUT /virtualdirectory

Uploads a virtual directory to the list of virtual directories

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Virtual directory updated | Empty |
| 400 | Error updating virtual directory. | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The virtual directory to be uploaded | True | [Virtualdirectory](#virtualdirectory) |  |

##### /virtualdirectory/{name}

###### GET /virtualdirectory/{name}

Returns an virtual directory by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the virtual directory | [Virtualdirectory](#virtualdirectory) |
| 400 | No virtual directory found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named virtual directory could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the virtual directory | True | String |

###### DELETE /virtualdirectory/{name}

Deletes an virtual directory by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Virtual directory found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named virtual directory could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the virtual directory | True | String |

##### /virtualdirectory/{name}/{filename}

###### GET /virtualdirectory/{name}/{filename}

Returns the specific file of that virtual directory

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | upload ok | [File](#file) |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the virtual directory | True | String |
| filename | path | The filename | True | String |

###### PUT /virtualdirectory/{name}/{filename}

Updates or adds a virtual file in the virtual directory

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | User added sucessfully | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the virtual directory | True | String |
| filename | path | The filename | True | String |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The user to be uploaded | True | [File](#file) |  |

###### DELETE /virtualdirectory/{name}/{filename}

Delete an user in the virtual directory

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the virtual directory | True | String |
| filename | path | The filename | True | String |

#### virtualdirectory.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: Virtual Directory  
 description: |-  
   
 A virtual directory is a collection of files, replicas, streams or other  
 virtual directories.  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup Service  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /virtualdirectory:  
 get:  
 tags:  
 - Virtual directory  
 summary: Returns a list of virtual directories  
 description: Returns a list of all virtual directories  
 operationId: cloudmesh.virtual directory.list  
 responses:  
 '200':  
 description: The list of virtual directories  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Virtualdirectory'  
 '400':  
 description: No virtual directories found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Virtual directory  
 summary: Uploads a virtual directory to the list of virtual directories  
 description: Uploads a virtual directory to the list of virtual directories  
 operationId: cloudmesh.virtual directory.add  
 requestBody:  
 description: The virtual directory to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Virtualdirectory'  
 responses:  
 '200':  
 description: Virtual directory updated  
 '400':  
 description: Error updating virtual directory.  
 '401':  
 description: Not authorized  
 /virtualdirectory/{name}:  
 get:  
 tags:  
 - Virtual directory  
 summary: Returns the named virtual directory  
 description: Returns an virtual directory by name  
 operationId: cloudmesh.virtualdirectory.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the virtual directory  
 responses:  
 '200':  
 description: Returning the information of the virtual directory  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Virtualdirectory'  
 '400':  
 description: No virtual directory found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named virtual directory could not be found  
 delete:  
 tags:  
 - Virtual directory  
 summary: Deletes the named virtual directory  
 description: Deletes an virtual directory by name  
 operationId: cloudmesh.virtualdirectory.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the virtual directory  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Virtual directory found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named virtual directory could not be found  
  
 /virtualdirectory/{name}/{filename}:  
 get:  
 tags:  
 - Virtual directory  
 summary: Returns the specific file of that virtual directory  
 description: Returns the specific file of that virtual directory  
 operationId: cloudmesh.virtualdirectory.file.get\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the virtual directory  
 - name: filename  
 description: The filename  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: upload ok  
 content:  
 application/json:  
 schema:  
 $ref: "filestore.yaml#/components/schemas/File"  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Virtual directory  
 summary: Updates or adds a virtual file in the virtual directory  
 description: Updates or adds a virtual file in the virtual directory  
 operationId: cloudmesh.virtualdirectory.file.add  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the virtual directory  
 - name: filename  
 description: The filename  
 in: path  
 required: true  
 schema:  
 type: string  
 requestBody:  
 description: The user to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: "filestore.yaml#/components/schemas/File"  
 responses:  
 '200':  
 description: User added sucessfully  
 '401':  
 description: Not authorized  
 delete:  
 tags:  
 - Virtual directory  
 summary: Delete an user in the virtual directory  
 description: Delete an user in the virtual directory  
 operationId: cloudmesh.virtualdirectory.file.delete  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the virtual directory  
 - name: filename  
 description: The filename  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Deletion successful  
 '401':  
 description: Not authorized  
components:  
 schemas:  
 Virtualdirectory:  
 type: object  
 description: the virtual directory  
 properties:  
 name:  
 description: The name of the virtual directory  
 type: string  
 description:  
 description: Description of the virtual directory  
 type: string  
 host:  
 description: Remote host of the virtual directory  
 type: string  
 location:  
 description: Remote location, e.g., a directory with full path on a  
 host  
 type: string  
 protocol:  
 description: Access protocol (e.g. HTTP, FTP, SSH, etc.)  
 type: string  
 credential:  
 description: Credential to access  
 type: object

## Compute Management - Virtual Clusters

### Virtual Cluster

A Virtual Cluster is modeled as manager node, and one or more compute nodes. The manager node usually serves as a login node and can be accessed from outside via a public IP. The compute nodes are connected to the manager node via a private, usually high performance (high throughput and low latency), network and thus accessible only from the manager node. To use the virtual cluster, login to the manager node, and from there one can login to any compute node, or submit a job to run on the compute nodes.

#### Schema Virtualcluster

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/virtualcluster.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | The name of the virtual cluster |
| description | string | A description of the virtual cluster |
| owner | string | Username of the owner of the virtual cluster |
| manager |  | Manager node of the virtual cluster |
| nodes | array[Node] | List of nodes of the virtual cluster |

#### Schema Node

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/node.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the node |
| state | string | Power state of the node |
| ncpu | integer | Number of virtual CPUs of the node |
| ram | string | RAM size of the node |
| disk | string | Disk size of the node |
| nics | array[NIC] | List of network interfaces of the node |

#### Schema NIC

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/nic.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| mac | string | MAC address of the node |
| ip | string | IP address of the node |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /virtualcluster | Returns a list of virtual clusters |
| put | /virtualcluster | Uploads an virtual cluster to the list of virtual clusters |
| get | /virtualcluster/{name} | Returns the named virtual cluster |
| delete | /virtualcluster/{name} | Deletes the named virtual cluster |
| get | /virtualcluster/{name}/{nodename} | Node of the named virtual cluster |
| put | /virtualcluster/{name}/{nodename} | Updates or adds a node to the virtual cluster |
| delete | /virtualcluster/{name}/{nodename} | Delete a node in the virtual cluster |

##### /virtualcluster

###### GET /virtualcluster

Returns a list of all virtual clusters

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of virtual clusters | array[[Virtualcluster](#virtualcluster)] |
| 401 | Not authorized | Empty |
| 400 | No Virtual clusters found | Empty |

###### PUT /virtualcluster

Uploads an virtual cluster to the list of virtual clusters

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Virtual cluster updated or created | Empty |
| 400 | Error updating Virtual cluster | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The virtual cluster to be uploaded | True | [Virtualcluster](#virtualcluster) |  |

##### /virtualcluster/{name}

###### GET /virtualcluster/{name}

Returns an virtual cluster by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the virtual cluster | [Virtualcluster](#virtualcluster) |
| 400 | No virtual cluster found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named virtual cluster could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the virtual cluster | True | String |

###### DELETE /virtualcluster/{name}

Deletes an virtual cluster by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Virtual cluster found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named virtual cluster could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the virtual cluster | True | String |

##### /virtualcluster/{name}/{nodename}

###### GET /virtualcluster/{name}/{nodename}

Returns the specific node of the named virtual cluster. If the node name is manager, the manager node is used. A compute node can not be named manager

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Node info | [Node](#node) |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the virtual cluster | True | String |
| nodename | path | The nodename | True | String |

###### PUT /virtualcluster/{name}/{nodename}

Updates or adds a node to the virtual cluster. If the node name is manager, the manager node is uploaded.

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Node added sucessfully | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the virtual cluster | True | String |
| nodename | path | The nodename | True | String |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The node to be uploaded | True | [Node](#node) |  |

###### DELETE /virtualcluster/{name}/{nodename}

Delete a node in the virtual cluster

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the virtual cluster | True | String |
| nodename | path | The nodename | True | String |

#### virtualcluster.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: TODO  
 title: Virtual Cluster  
 description: |-  
  
 A Virtual Cluster is modeled as manager node, and one or more  
 compute nodes. The manager node usually serves as a login node and  
 can be accessed from outside via a public IP. The compute nodes are  
 connected to the manager node via a private, usually high performance (high  
 throughput and low latency), network and thus accessible only from the  
 manager node. To use the virtual cluster, login to the manager node, and  
 from there one can login to any compute node, or submit a job to run on the  
 compute nodes.  
   
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup Service  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /virtualcluster:  
 get:  
 tags:  
 - Virtual cluster  
 summary: Returns a list of virtual clusters  
 description: Returns a list of all virtual clusters  
 operationId: cloudmesh.virtualcluster.list  
 responses:  
 '200':  
 description: The list of virtual clusters  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Virtualcluster'  
 '401':  
 description: Not authorized  
 '400':  
 description: No Virtual clusters found  
 put:  
 tags:  
 - Virtual cluster  
 summary: Uploads an virtual cluster to the list of virtual clusters  
 description: Uploads an virtual cluster to the list of virtual clusters  
 operationId: cloudmesh.virtualcluster.add  
 requestBody:  
 description: The virtual cluster to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Virtualcluster'  
 responses:  
 '200':  
 description: Virtual cluster updated or created  
 '400':  
 description: Error updating Virtual cluster  
 '401':  
 description: Not authorized  
 /virtualcluster/{name}:  
 get:  
 tags:  
 - Virtual cluster  
 summary: Returns the named virtual cluster  
 description: Returns an virtual cluster by name  
 operationId: cloudmesh.virtualcluster.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the virtual cluster  
 responses:  
 '200':  
 description: Returning the information of the virtual cluster  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Virtualcluster'  
 '400':  
 description: No virtual cluster found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named virtual cluster could not be found  
 delete:  
 tags:  
 - Virtual cluster  
 summary: Deletes the named virtual cluster  
 description: Deletes an virtual cluster by name  
 operationId: cloudmesh.virtualcluster.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the virtual cluster  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Virtual cluster found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named virtual cluster could not be found  
 /virtualcluster/{name}/{nodename}:  
 get:  
 tags:  
 - Virtual cluster  
 summary: Node of the named virtual cluster  
 description: Returns the specific node of the named virtual cluster. If  
 the node name is manager, the manager node is used. A  
 compute node can not be named manager  
 operationId: cloudmesh.virtualcluster.node.get\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the virtual cluster  
 - name: nodename  
 description: The nodename  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Node info  
 content:  
 application/json:  
 schema:  
 $ref: "#/components/schemas/Node"  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Virtual cluster  
 summary: Updates or adds a node to the virtual cluster  
 description: Updates or adds a node to the virtual cluster. If  
 the node name is manager, the manager node is uploaded.  
 operationId: cloudmesh.virtualcluster.node.add  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the virtual cluster  
 - name: nodename  
 description: The nodename  
 in: path  
 required: true  
 schema:  
 type: string  
 requestBody:  
 description: The node to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Node'  
 responses:  
 '200':  
 description: Node added sucessfully  
 '401':  
 description: Not authorized  
 delete:  
 tags:  
 - Virtual cluster  
 summary: Delete a node in the virtual cluster  
 description: Delete a node in the virtual cluster  
 operationId: cloudmesh.virtualcluster.node.delete  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the virtual cluster  
 - name: nodename  
 description: The nodename  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Deletion successful  
 '401':  
 description: Not authorized  
components:  
 schemas:  
 Virtualcluster:  
 type: object  
 properties:  
 name:  
 description: The name of the virtual cluster  
 type: string  
 description:  
 type: string  
 description: A description of the virtual cluster  
 owner:  
 type: string  
 description: Username of the owner of the virtual cluster  
 manager:  
 description: Manager node of the virtual cluster  
 $ref: "#/components/schemas/Node"  
 nodes:  
 description: List of nodes of the virtual cluster  
 type: array  
 items:  
 $ref: "#/components/schemas/Node"  
 Node:  
 type: object  
 properties:  
 name:  
 type: string  
 description: Name of the node  
 state:  
 type: string  
 description: Power state of the node  
 ncpu:  
 type: integer  
 description: Number of virtual CPUs of the node  
 ram:  
 type: string  
 description: RAM size of the node  
 disk:  
 type: string  
 description: Disk size of the node  
 nics:  
 type: array  
 description: List of network interfaces of the node  
 items:  
 $ref: "#/components/schemas/NIC"  
 NIC:  
 type: object  
 properties:  
 mac:  
 type: string  
 description: MAC address of the node  
 ip:  
 type: string  
 description: IP address of the node

### Scheduler

The queue is a special scheduler that allows tasks to be scheduled witle doing queue policies, such as LIFO, FIFO, and so on. A queue returns the next task to be executed. Tasks can be added and deleted.

#### Schema Task

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/task.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the scheduler |
| user | string | The username the task belongs to |
| description | string | The description of the task |
| kind | string | The kind of the task |

#### Schema Policy

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/policy.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the scheduler policy |
| description | string | The description of the policy |
| kind | string | The kind of the policy |
| parameters | string | parameters to define the behaviour of the scheduler |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /task/next | Returns the next tasks and removed it from the list of tasks |
| get | /task/info | Returns information about the task that is to be scheduled next |
| get | /task/pop | Removes the next task to be scheduled, but does not exualle executes it. scheduled next |
| get | /task | Returns a list of tasks |
| put | /task | Uploads a task to the list of tasks |
| get | /task/{name} | Returns the named task |
| delete | /task/{name} | Deletes the named task |
| get | /policy | Returns the policy |
| put | /policy | Uploads the policy |

##### /task/next

###### GET /task/next

Returns the next task

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The task | array[[Task](#task)] |
| 400 | No tasks found | Empty |
| 401 | Not authorized | Empty |

##### /task/info

###### GET /task/info

Returns the next task

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The task | array[[Task](#task)] |
| 400 | No tasks found | Empty |
| 401 | Not authorized | Empty |

##### /task/pop

###### GET /task/pop

Returns the next task

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The task | array[[Task](#task)] |
| 400 | No tasks found | Empty |
| 401 | Not authorized | Empty |

##### /task

###### GET /task

Returns a list of all tasks

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of tasks | array[[Task](#task)] |
| 400 | No tasks found | Empty |
| 401 | Not authorized | Empty |

###### PUT /task

Uploads a task to the list of tasks

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Task updated | Empty |
| 400 | Error updating task. | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The task to be uploaded | True | [Task](#task) |  |

##### /task/{name}

###### GET /task/{name}

Returns an task by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the task | [Task](#task) |
| 400 | No task found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named task could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the task | True | String |

###### DELETE /task/{name}

Deletes an task by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No task found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named task could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the task | True | String |

##### /policy

###### GET /policy

Returns the polocy

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The policy | array[[Policy](#policy)] |
| 400 | No tasks found | Empty |
| 401 | Not authorized | Empty |

###### PUT /policy

Uploads a task to the list of tasks

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Task updated | Empty |
| 400 | Error updating task | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The Policy | True | [Policy](#policy) |  |

#### scheduler.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: TODO  
 title: Scheduler  
 description: |-  
   
 The queue is a special scheduler that allows tasks to be scheduled  
 witle doing queue policies, such as LIFO, FIFO, and so on.  
 A queue returns the next task to be executed. Tasks can be added  
 and  
 deleted.  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3/scheduler  
paths:  
 /task/next:  
 get:  
 tags:  
 - Task  
 summary: Returns the next tasks and removed it from the list of tasks  
 description: Returns the next task  
 operationId: cloudmesh.task.next  
 responses:  
 '200':  
 description: The task  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Task'  
 '400':  
 description: No tasks found  
 '401':  
 description: Not authorized  
 /task/info:  
 get:  
 tags:  
 - Task  
 summary: Returns information about the task that is to be scheduled next  
 description: Returns the next task  
 operationId: cloudmesh.task.info  
 responses:  
 '200':  
 description: The task  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Task'  
 '400':  
 description: No tasks found  
 '401':  
 description: Not authorized  
 /task/pop:  
 get:  
 tags:  
 - Task  
 summary: Removes the next task to be scheduled, but does not  
 exualle executes it.  
 scheduled next  
 description: Returns the next task  
 operationId: cloudmesh.task.pop  
 responses:  
 '200':  
 description: The task  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Task'  
 '400':  
 description: No tasks found  
 '401':  
 description: Not authorized  
 /task:  
 get:  
 tags:  
 - Task  
 summary: Returns a list of tasks  
 description: Returns a list of all tasks  
 operationId: cloudmesh.task.list  
 responses:  
 '200':  
 description: The list of tasks  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Task'  
 '400':  
 description: No tasks found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Task  
 summary: Uploads a task to the list of tasks  
 description: Uploads a task to the list of tasks  
 operationId: cloudmesh.task.add  
 requestBody:  
 description: The task to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Task'  
 responses:  
 '200':  
 description: Task updated  
 '400':  
 description: Error updating task.  
 '401':  
 description: Not authorized  
 /task/{name}:  
 get:  
 tags:  
 - Task  
 summary: Returns the named task  
 description: Returns an task by name  
 operationId: cloudmesh.task.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the task  
 responses:  
 '200':  
 description: Returning the information of the task  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Task'  
 '400':  
 description: No task found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named task could not be found  
 delete:  
 tags:  
 - Task  
 summary: Deletes the named task  
 description: Deletes an task by name  
 operationId: cloudmesh.task.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the task  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No task found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named task could not be found  
 /policy:  
 get:  
 tags:  
 - Task  
 summary: Returns the policy  
 description: Returns the polocy  
 operationId: cloudmesh.task.policy.list  
 responses:  
 '200':  
 description: The policy  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Policy'  
 '400':  
 description: No tasks found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Task  
 summary: Uploads the policy  
 description: Uploads a task to the list of tasks  
 operationId: cloudmesh.task.policy.add  
 requestBody:  
 description: The Policy  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Policy'  
 responses:  
 '200':  
 description: Task updated  
 '400':  
 description: Error updating task  
 '401':  
 description: Not authorized  
components:  
 schemas:  
 Task:  
 type: object  
 description: The scheduler  
 properties:  
 name:  
 type: string  
 description: Name of the scheduler  
 user:  
 type: string  
 description: The username the task belongs to  
 description:  
 type: string  
 description: The description of the task  
 kind:  
 type: string  
 description: The kind of the task  
 Policy:  
 type: object  
 description: The policy of the scheduler  
 properties:  
 name:  
 type: string  
 description: Name of the scheduler policy  
 description:  
 type: string  
 description: The description of the policy  
 kind:  
 type: string  
 description: The kind of the policy  
 parameters:  
 type: string  
 description: parameters to define the behaviour of the scheduler

## Compute Management - Virtual Machines

This section summarizes a basic interface specification of virtual machines.

### Image

To execute virtual machines, we need an image that specifies the details of the operationg system.

#### Schema Image

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/image.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | A unique name of the image |
| cloud | string | The name of the cloud |
| label | string | A label that can be defined by the user for the image |
| description | string | A description for the image |
| osType | string | The OS type of the image |
| osVersion | string | The OS version of the image |
| status | string | The status of the image |
| progress | integer | The loading progress percentage of the image |
| visibility | string | The visibility of the image |
| requirement |  | Minimum requirement to run the image |

#### Schema Requirements

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/requirements.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| size | integer | Minimum disk size in bytes required for the image |
| ram | integer | Minimum ram size in bytes to run the image |
| cpu | string | CPU required to run the image |
| cores | integer | Minimum number of cores |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /image/{cloud} | Returns a list of images for the cloud |
| put | /image/{cloud} | Add a image |
| get | /image/{cloud}/{name} | Returns the named image |
| delete | /image/{cloud}/{name} | Deletes the named image |

##### /image/{cloud}

###### GET /image/{cloud}

Returns a list of all images

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of images | array[[Image](#image)] |
| 400 | No Image found | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |

###### PUT /image/{cloud}

Sets the named image

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Image updated | Empty |
| 400 | Error updating image. | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The image to add or modify | True | [Image](#image) |  |

##### /image/{cloud}/{name}

###### GET /image/{cloud}/{name}

Returns a image by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the image | [Image](#image) |
| 400 | No Image found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named image could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |
| name | path | The name of the image | True | String |

###### DELETE /image/{cloud}/{name}

Deletes a image by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Image found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named image could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |
| name | path | The name of the image | True | String |

#### image.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: TODO  
 title: Image  
 description: |-  
   
 To execute virtual machines, we need an image that specifies the  
 details of the operationg system.  
   
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /image/{cloud}:  
 get:  
 tags:  
 - Image  
 summary: Returns a list of images for the cloud  
 description: Returns a list of all images  
 operationId: cloudmesh.image.list  
 parameters:  
 - name: cloud  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the cloud  
 responses:  
 '200':  
 description: The list of images  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Image'  
 '400':  
 description: No Image found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Image  
 summary: Add a image  
 description: Sets the named image  
 operationId: cloudmesh.image.add  
 parameters:  
 - name: cloud  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the cloud  
 requestBody:  
 description: The image to add or modify  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Image'  
 responses:  
 '200':  
 description: Image updated  
 '400':  
 description: Error updating image.  
 '401':  
 description: Not authorized  
 /image/{cloud}/{name}:  
 get:  
 tags:  
 - Image  
 summary: Returns the named image  
 description: Returns a image by name  
 operationId: cloudmesh.image.find\_by\_name  
 parameters:  
 - name: cloud  
 in: path  
 description: The name of the cloud  
 required: true  
 schema:  
 type: string  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the image  
 responses:  
 '200':  
 description: Returning the information of the image  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Image'  
 '400':  
 description: No Image found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named image could not be found  
 delete:  
 tags:  
 - Image  
 summary: Deletes the named image  
 description: Deletes a image by name  
 operationId: cloudmesh.image.delete\_by\_name  
 parameters:  
 - name: cloud  
 description: The name of the cloud  
 in: path  
 required: true  
 schema:  
 type: string  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the image  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Image found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named image could not be found  
components:  
 schemas:  
 Image:  
 type: object  
 properties:  
 name:  
 type: string  
 description: A unique name of the image  
 cloud:  
 type: string  
 description: The name of the cloud  
 label:  
 type: string  
 description: A label that can be defined by the user for the image  
 description:  
 type: string  
 description: A description for the image  
 osType:  
 type: string  
 description: The OS type of the image  
 osVersion:  
 type: string  
 description: The OS version of the image  
 status:  
 type: string  
 description: The status of the image  
 progress:  
 type: integer  
 description: The loading progress percentage of the image  
 visibility:  
 description: The visibility of the image  
 type: string  
 requirement:  
 $ref: "#/components/schemas/Requirements"  
 description: Minimum requirement to run the image  
 Requirements:  
 type: object  
 properties:  
 size:  
 type: integer  
 description: Minimum disk size in bytes required for the image  
 ram:  
 type: integer  
 description: Minimum ram size in bytes to run the image  
 cpu:  
 type: string  
 description: CPU required to run the image  
 cores:  
 type: integer  
 description: Minimum number of cores

### Flavor

The flavor specifies elementary information about a virtual machine or compute node. This information includes name, id, label, ram size, swap size, disk space, availability of ephemeral disk, available bandwidth, price value, cloud name. Flavors and the corresponding information are essential to size a virtual cluster appropriately.

#### Schema Flavor

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/flavor.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the flavor |
| id | string | The id of the flavor for the named cloud |
| label | string | A label that a user can set for this flavor |
| description | string | A description for the flavor |
| ram | integer | Number of bytes used for the image in RAM |
| swap | integer | Number of bytes used for the image in SWAP |
| disk | integer | Number of bytes used for the disk |
| ephemeral\_disk | boolean | Specifies whether the flavor features an ephemeral disk |
| bandwidth | integer | Bandwidth of the node |
| price | number | Price for the flavor |
| cloud | string | Name of the cloud this flavor is used in |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /flavor/{cloud} | Returns a list of flavors for the cloud |
| put | /flavor/{cloud} | Add a flavor |
| get | /flavor/{cloud}/{name} | Returns the named flavor |
| delete | /flavor/{cloud}/{name} | Deletes the named flavor |

##### /flavor/{cloud}

###### GET /flavor/{cloud}

Returns a list of all flavors

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of flavors | array[[Flavor](#flavor)] |
| 400 | No Flavor found | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |

###### PUT /flavor/{cloud}

Sets the named flavor

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Flavor updated | Empty |
| 400 | Error updating flavor. | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The flavor to add or modify | True | [Flavor](#flavor) |  |

##### /flavor/{cloud}/{name}

###### GET /flavor/{cloud}/{name}

Returns a flavor by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the flavor | [Flavor](#flavor) |
| 400 | No Flavor found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named flavor could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |
| name | path | The name of the flavor | True | String |

###### DELETE /flavor/{cloud}/{name}

Deletes a flavor by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Flavor found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named flavor could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |
| name | path | The name of the flavor | True | String |

#### flavor.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: defined  
 title: Flavor  
 description: |-  
   
 The flavor specifies elementary information about a virtual machine  
 or compute node. This information includes name, id, label, ram size,  
 swap size, disk space, availability of ephemeral disk, available  
 bandwidth, price value, cloud name. Flavors and the corresponding  
 information are essential to size a  
 virtual cluster appropriately.  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /flavor/{cloud}:  
 get:  
 tags:  
 - Flavor  
 summary: Returns a list of flavors for the cloud  
 description: Returns a list of all flavors  
 operationId: cloudmesh.flavor.list  
 parameters:  
 - name: cloud  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the cloud  
 responses:  
 '200':  
 description: The list of flavors  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Flavor'  
 '400':  
 description: No Flavor found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Flavor  
 summary: Add a flavor  
 description: Sets the named flavor  
 operationId: cloudmesh.flavor.add  
 parameters:  
 - name: cloud  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the cloud  
 requestBody:  
 description: The flavor to add or modify  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Flavor'  
 responses:  
 '200':  
 description: Flavor updated  
 '400':  
 description: Error updating flavor.  
 '401':  
 description: Not authorized  
 /flavor/{cloud}/{name}:  
 get:  
 tags:  
 - Flavor  
 summary: Returns the named flavor  
 description: Returns a flavor by name  
 operationId: cloudmesh.flavor.find\_by\_name  
 parameters:  
 - name: cloud  
 in: path  
 description: The name of the cloud  
 required: true  
 schema:  
 type: string  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the flavor  
 responses:  
 '200':  
 description: Returning the information of the flavor  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Flavor'  
 '400':  
 description: No Flavor found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named flavor could not be found  
 delete:  
 tags:  
 - Flavor  
 summary: Deletes the named flavor  
 description: Deletes a flavor by name  
 operationId: cloudmesh.flavor.delete\_by\_name  
 parameters:  
 - name: cloud  
 description: The name of the cloud  
 in: path  
 required: true  
 schema:  
 type: string  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the flavor  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Flavor found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named flavor could not be found  
components:  
 schemas:  
 Flavor:  
 type: object  
 description: The flavor  
 properties:  
 name:  
 type: string  
 description: Name of the flavor  
 id:  
 type: string  
 description: The id of the flavor for the named cloud  
 label:  
 type: string  
 description: A label that a user can set for this flavor  
 description:  
 type: string  
 description: A description for the flavor  
 ram:  
 type: integer  
 description: Number of bytes used for the image in RAM  
 swap:  
 type: integer  
 description: Number of bytes used for the image in SWAP  
 disk:  
 type: integer  
 description: Number of bytes used for the disk  
 ephemeral\_disk:  
 type: boolean  
 description: Specifies whether the flavor features an ephemeral disk  
 bandwidth:  
 type: integer  
 description: Bandwidth of the node  
 price:  
 type: number  
 description: Price for the flavor  
 cloud:  
 type: string  
 description: Name of the cloud this flavor is used in

### Virtual Machine

Vm is a service to manage virtual machines.

#### Schema Vm

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/vm.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| provider | string | Name of the provider |
| name | string | the unique name of the virtual machine |
| image | string | the image name for the virtual machine |
| flavor | string | the flavor name for the virtual machine |
| region | string | an optional region |
| state | string | The state of the virtual machine |
| private\_ips | string | The private IPs |
| public\_ips | string | The public IPS |
| metadata | string | The meta data passed along to the virtual machine |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /vm/{cloud} | Returns a list of virtual machines for the cloud |
| put | /vm/{cloud} | Add a virtual machine |
| get | /vm/{cloud}/{name} | Returns the named virtual machine |
| delete | /vm/{cloud}/{name} | Deletes the named virtual machine |

##### /vm/{cloud}

###### GET /vm/{cloud}

Returns a list of all virtual machines

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of virtual machines | array[[Vm](#vm)] |
| 400 | No Vm found | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |

###### PUT /vm/{cloud}

Sets the named virtual machine

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Vm updated | Empty |
| 400 | Error updating virtual machine | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The virtual machine to add or modify | True | [Vm](#vm) |  |

##### /vm/{cloud}/{name}

###### GET /vm/{cloud}/{name}

Returns a virtual machine by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the virtual machine | [Vm](#vm) |
| 400 | No Vm found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named virtual machine could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |
| name | path | The name of the virtual machine | True | String |

###### DELETE /vm/{cloud}/{name}

Deletes a virtual machine by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Vm found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named virtual machine could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| cloud | path | The name of the cloud | True | String |
| name | path | The name of the virtual machine | True | String |

#### vm.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: TODO  
 title: Virtual Machine  
 description: |-  
  
 Vm is a service to manage virtual machines.  
   
 termsOfService: https://github.com/cloudmesh-community/nist/blob/master/LICENSE.txt  
 contact:  
 name: NIST BDRA Interface Subgroup Service  
 url: https://cloudmesh-community.github.io/nist/spec/  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /vm/{cloud}:  
 get:  
 tags:  
 - Vm  
 summary: Returns a list of virtual machines for the cloud  
 description: Returns a list of all virtual machines  
 operationId: cloudmesh.vm.list  
 parameters:  
 - name: cloud  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the cloud  
 responses:  
 '200':  
 description: The list of virtual machines  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Vm'  
 '400':  
 description: No Vm found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Vm  
 summary: Add a virtual machine  
 description: Sets the named virtual machine  
 operationId: cloudmesh.vm.add  
 parameters:  
 - name: cloud  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the cloud  
 requestBody:  
 description: The virtual machine to add or modify  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Vm'  
 responses:  
 '200':  
 description: Vm updated  
 '400':  
 description: Error updating virtual machine  
 '401':  
 description: Not authorized  
 /vm/{cloud}/{name}:  
 get:  
 tags:  
 - Vm  
 summary: Returns the named virtual machine  
 description: Returns a virtual machine by name  
 operationId: cloudmesh.vm.find\_by\_name  
 parameters:  
 - name: cloud  
 in: path  
 description: The name of the cloud  
 required: true  
 schema:  
 type: string  
 - name: name  
 in: path  
 description: The name of the virtual machine  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Returning the information of the virtual machine  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Vm'  
 '400':  
 description: No Vm found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named virtual machine could not be found  
 delete:  
 tags:  
 - Vm  
 summary: Deletes the named virtual machine  
 description: Deletes a virtual machine by name  
 operationId: cloudmesh.vm.delete\_by\_name  
 parameters:  
 - name: cloud  
 description: The name of the cloud  
 in: path  
 required: true  
 schema:  
 type: string  
 - name: name  
 in: path  
 description: The name of the virtual machine  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Vm found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named virtual machine could not be found  
components:  
 schemas:  
 Vm:  
 type: object  
 properties:  
 provider:  
 type: string  
 description: Name of the provider  
 name:  
 type: string  
 description: the unique name of the virtual machine  
 image:  
 type: string  
 description: the image name for the virtual machine  
 flavor:  
 type: string  
 description: the flavor name for the virtual machine  
 region:  
 type: string  
 description: an optional region  
 state:  
 type: string  
 description: The state of the virtual machine  
 private\_ips:  
 type: string  
 description: The private IPs  
 public\_ips:  
 type: string  
 description: The public IPS  
 metadata:  
 type: string  
 description: The meta data passed along to the virtual machine

### Secgroup

A security group defines the incoming and outgoing security rules which can then be assigned to a node. The connection to and from the node will be determined by the security group rules, in addition to any other possible rules applied on network devices or from the instance’s firewall settings. A security group may have one or multiple rules and a node may be associated with one or more security groups.

#### Schema Secgroup

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/secgroup.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the security group |
| description | string | Describes what the security group is for |
| rules | array[SecGroupRule] | List of Security group rules |

#### Schema SecGroupRule

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/secgrouprule.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Unique name of the rule |
| ingress | boolean | The defined security group rule is for ingress if True |
| egress | boolean | The defined security group rule is for egress if True |
| remote\_group | string | Name of the group if the rule is defined by group instead of IP range |
| protocol | string | The protocol used such as TCP, UDP, ICMP |
| from\_port | integer | Port range starting port |
| to\_port | integer | Port range ending port |
| cidr | string | The source or destination network in CIDR notation, e.g., 129.79.0.0/16 |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /secgroup | Returns all security groups |
| get | /secgroup/{name} | Return the security group by name |
| post | /secgroup/{name} | Create a new named security group |
| post | /secgroup/{name}/rule | Create a new rule in the specified security group |
| get | /secgroup/{name}/rule/{rule} | Get an existing rule from the specified security group |
| delete | /secgroup/{name}/rule/{rule} | Delete an existing rule from the specified security group |

##### /secgroup

###### GET /secgroup

Returns all security groups

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | security group information | array[[Secgroup](#secgroup)] |
| 400 | Not found | Empty |
| 401 | Not authorized | Empty |

##### /secgroup/{name}

###### GET /secgroup/{name}

Return the security group by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | security group information | [Secgroup](#secgroup) |
| 400 | Not found | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | name of the security group | True | String |

###### POST /secgroup/{name}

Create a new named security group

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 201 | Created | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the security group to create | True | String |

##### /secgroup/{name}/rule

###### POST /secgroup/{name}/rule

Create a new rule in security group

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Created | Empty |
| 400 | Not found | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the new security group to create | True | String |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The new security group rule to create | True | [SecGroupRule](#secgrouprule) |  |

##### /secgroup/{name}/rule/{rule}

###### GET /secgroup/{name}/rule/{rule}

Create a new rule in security group

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The security group rule definition info | [Secgrouprule](#secgrouprule) |
| 400 | Not found | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The named of the security group from which the rule will be deleted | True | String |
| rule | path | The rule to be added | True | String |

###### DELETE /secgroup/{name}/rule/{rule}

Create a new rule in security group

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deleted sucessfully | Empty |
| 400 | Not found | Empty |
| 401 | Not authorized | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The named of the secgroup from which the rule will be deleted | True | String |
| rule | path | The rule to be deleted | True | String |

#### secgroup.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: TODO  
 title: Secgroup  
 description: |-  
   
 A security group defines the incoming and outgoing security rules  
 which can then be assigned to a node. The connection to and from the node  
 will be determined by the security group rules, in addition to any other  
 possible rules applied on network devices or from the instance's firewall  
 settings. A security group may have one or multiple rules and a node may be  
 associated with one or more security groups.  
   
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /secgroup:  
 get:  
 tags:  
 - Security group  
 summary: Returns all security groups  
 description: Returns all security groups  
 operationId: cloudmesh.secgroup.get  
 responses:  
 '200':  
 description: security group information  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: "#/components/schemas/Secgroup"  
 '400':  
 description: Not found  
 '401':  
 description: Not authorized  
 /secgroup/{name}:  
 get:  
 tags:  
 - Security group  
 summary: Return the security group by name  
 description: Return the security group by name  
 operationId: cloudmesh.secgroup.get\_by\_name  
 parameters:  
 - name: name  
 description: name of the security group  
 in: path  
 required: true  
 schema:  
 type: string  
 responses:  
 '200':  
 description: security group information  
 content:  
 application/json:  
 schema:  
 $ref: "#/components/schemas/Secgroup"  
 '400':  
 description: Not found  
 '401':  
 description: Not authorized  
 post:  
 tags:  
 - Security group  
 summary: Create a new named security group  
 description: Create a new named security group  
 operationId: cloudmesh.secgroup.add  
 parameters:  
 - in: path  
 name: name  
 required: true  
 description: The name of the security group to create  
 schema:  
 type: string  
 responses:  
 '201':  
 description: Created  
 '401':  
 description: Not authorized  
 /secgroup/{name}/rule:  
 post:  
 tags:  
 - Security group  
 summary: Create a new rule in the specified security group  
 description: Create a new rule in security group  
 operationId: cloudmesh.secgroup.add\_rule  
 parameters:  
 - in: path  
 name: name  
 required: true  
 description: The name of the new security group to create  
 schema:  
 type: string  
 requestBody:  
 description: The new security group rule to create  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/SecGroupRule'  
 responses:  
 '200':  
 description: Created  
 '400':  
 description: Not found  
 '401':  
 description: Not authorized  
 /secgroup/{name}/rule/{rule}:  
 get:  
 tags:  
 - Security group  
 summary: Get an existing rule from the specified security group  
 description: Create a new rule in security group  
 operationId: cloudmesh.secgroup.get\_rule  
 parameters:  
 - in: path  
 name: name  
 required: true  
 description: The named of the security group from which the rule will be deleted  
 schema:  
 type: string  
 - in: path  
 name: rule  
 required: true  
 description: The rule to be added  
 schema:  
 type: string  
 responses:  
 '200':  
 description: The security group rule definition info  
 content:  
 application/json:  
 schema:  
 $ref: "#/components/schemas/SecGroupRule"  
 '400':  
 description: Not found  
 '401':  
 description: Not authorized  
 delete:  
 tags:  
 - Security group  
 summary: Delete an existing rule from the specified security group  
 description: Create a new rule in security group  
 operationId: cloudmesh.secgroup.delete\_rule  
 parameters:  
 - in: path  
 name: name  
 required: true  
 description: The named of the secgroup from which the rule will be deleted  
 schema:  
 type: string  
 - in: path  
 name: rule  
 required: true  
 description: The rule to be deleted  
 schema:  
 type: string  
 responses:  
 '200':  
 description: Deleted sucessfully  
 '400':  
 description: Not found  
 '401':  
 description: Not authorized  
components:  
 schemas:  
 Secgroup:  
 type: object  
 description: the security group object  
 properties:  
 name:  
 type: string  
 description: Name of the security group  
 description:  
 type: string  
 description: Describes what the security group is for  
 rules:  
 type: array  
 description: List of Security group rules  
 items:  
 $ref: "#/components/schemas/SecGroupRule"  
 SecGroupRule:  
 type: object  
 description: security group rule  
 properties:  
 name:  
 type: string  
 description: Unique name of the rule  
 ingress:  
 type: boolean  
 description: The defined security group rule is for ingress if True  
 egress:  
 type: boolean  
 description: The defined security group rule is for egress if True  
 remote\_group:  
 type: string  
 description: Name of the group if the rule is defined by group instead of IP range  
 protocol:  
 type: string  
 description: The protocol used such as TCP, UDP, ICMP  
 from\_port:  
 type: integer  
 description: Port range starting port  
 to\_port:  
 type: integer  
 description: Port range ending port  
 cidr:  
 type: string  
 description: The source or destination network in CIDR notation, e.g., 129.79.0.0/16

### Nic

A service to store Network Interface Controller (NIC) information.

#### Schema Nic

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/nic.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the network interface controller |
| kind | string | Kind of the network interface controller (wifi, WAN, …) |
| mac | string | The mac address |
| ip | string | The IP address |
| mask | string | The network mask |
| broadcast | string | The broadcast address |
| gateway | string | The gateway address |
| mtu | integer | The MTU of the NIC |
| bandwidth | integer | The bandwidth in bps |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /nic | Returns a list of network interface controllers |
| put | /nic | Set a network interface controller |
| get | /nic/{name} | Returns the named network interface controller |
| delete | /nic/{name} | Deletes the named network interface controller |

##### /nic

###### GET /nic

Returns a list of all network interface controllers

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of network interface controllers | array[[Nic](#nic)] |
| 400 | No Nic found | Empty |
| 401 | Not authorized | Empty |

###### PUT /nic

Sets the named network interface controller

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Nic updated | Empty |
| 400 | Error updating network interface controller | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The new nic to create | True | [Nic](#nic) |  |

##### /nic/{name}

###### GET /nic/{name}

Returns a network interface controller by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the network interface controller | [Nic](#nic) |
| 400 | No Nic found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named network interface controller could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the network interface controller | True | String |

###### DELETE /nic/{name}

Deletes a network interface controller by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Nic found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named network interface controller could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the network interface controller | True | String |

#### nic.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: TODO  
 title: Nic  
 description: |-  
   
 A service to store Network Interface Controller (NIC) information.  
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /nic:  
 get:  
 tags:  
 - Nic  
 summary: Returns a list of network interface controllers  
 description: Returns a list of all network interface controllers  
 operationId: cloudmesh.nic.list  
 responses:  
 '200':  
 description: The list of network interface controllers  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Nic'  
 '400':  
 description: No Nic found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Nic  
 summary: Set a network interface controller  
 description: Sets the named network interface controller  
 operationId: cloudmesh.nic.add  
 requestBody:  
 description: The new nic to create  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Nic'  
 responses:  
 '200':  
 description: Nic updated  
 '400':  
 description: Error updating network interface controller  
 '401':  
 description: Not authorized  
 /nic/{name}:  
 get:  
 tags:  
 - Nic  
 summary: Returns the named network interface controller  
 description: Returns a network interface controller by name  
 operationId: cloudmesh.nic.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the network interface controller  
 responses:  
 '200':  
 description: Returning the information of the network interface controller  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Nic'  
 '400':  
 description: No Nic found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named network interface controller could not be found  
 delete:  
 tags:  
 - Nic  
 summary: Deletes the named network interface controller  
 description: Deletes a network interface controller by name  
 operationId: cloudmesh.nic.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the network interface controller  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Nic found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named network interface controller could not be found  
components:  
 schemas:  
 Nic:  
 type: object  
 description: The network interface controller  
 properties:  
 name:  
 type: string  
 description: Name of the network interface controller  
 kind:  
 type: string  
 description: Kind of the network interface controller (wifi, WAN, ...)  
 mac:  
 type: string  
 description: The mac address  
 ip:  
 type: string  
 description: The IP address  
 mask:  
 type: string  
 description: The network mask  
 broadcast:  
 type: string  
 description: The broadcast address  
 gateway:  
 type: string  
 description: The gateway address  
 mtu:  
 type: integer  
 description: The MTU of the NIC  
 bandwidth:  
 type: integer  
 description: The bandwidth in bps

## Compute Management - Containers

### Containers

Numerous different containers are likely to be created and handling them becomes more and more time consuming as their number increases. This service helps to solve that issue by storing containers and their corresponding information.

#### Schema Container

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/container.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the container |
| version | string | Version of the container |
| label | string | Label of the container |
| type | string | Type of the container |
| definition | string | Definition or manifest of the container |
| imgURI | string | URI of the container |
| tags | array[string] | Tags of the container |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /container | Returns a list of containers |
| put | /container | Set an container |
| get | /container/{name} | Returns the named container |
| delete | /container/{name} | Deletes the named container |

##### /container

###### GET /container

Returns a list of all containers

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of containerses | array[[Container](#container)] |
| 400 | No Container found | Empty |

###### PUT /container

Sets the named container

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Container updated | Empty |
| 401 | Not authorized | Empty |
| 400 | Error updating container. | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The new container to create | True | [Container](#container) |  |

##### /container/{name}

###### GET /container/{name}

Returns an container by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the container | [Container](#container) |
| 400 | No Container found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named container could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the container | True | String |

###### DELETE /container/{name}

Deletes an container by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Container found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named container could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the container | True | String |

#### containers.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: Containers  
 description: |-  
  
 Numerous different containers are likely to be created and handling them   
 becomes more and more time consuming as their number increases. This service   
 helps to solve that issue by storing containers and their corresponding   
 information.   
  
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /container:  
 get:  
 tags:  
 - Container  
 summary: Returns a list of containers  
 description: Returns a list of all containers  
 operationId: cloudmesh.container.list  
 responses:  
 '200':  
 description: The list of containerses  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Container'  
 '400':  
 description: No Container found  
 put:  
 tags:  
 - Container  
 summary: Set an container  
 description: Sets the named container  
 operationId: cloudmesh.container.add  
 requestBody:  
 description: The new container to create  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Container'  
 responses:  
 '200':  
 description: Container updated  
 '401':  
 description: Not authorized  
 '400':  
 description: Error updating container.  
 /container/{name}:  
 get:  
 tags:  
 - Container  
 summary: Returns the named container  
 description: Returns an container by name  
 operationId: cloudmesh.container.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the container  
 responses:  
 '200':  
 description: Returning the information of the container  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Container'  
 '400':  
 description: No Container found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named container could not be found  
 delete:  
 tags:  
 - Container  
 summary: Deletes the named container  
 description: Deletes an container by name  
 operationId: cloudmesh.container.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the container  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Container found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named container could not be found  
components:  
 schemas:  
 Container:  
 type: object  
 description: A record representing a container  
 properties:  
 name:  
 type: string  
 description: Name of the container  
 version:  
 type: string  
 description: Version of the container  
 label:  
 type: string  
 description: Label of the container  
 type:  
 type: string  
 description: Type of the container  
 definition:  
 type: string  
 description: Definition or manifest of the container  
 imgURI:  
 type: string  
 description: URI of the container  
 tags:  
 type: array  
 description: Tags of the container  
 items:  
 type: string

## Compute Management - Functions

### Microservice

As part of microservices, a function with parameters that can be invoked has been defined.

#### Schema Microservice

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/microservice.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the microservice |
| endpoint | string | The end point of the microservice |
| function | string | The function the microservice represents |
| description |  | The description of the microservice |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /microservice | Returns a list of microservicees |
| put | /microservice | Set an microservice |
| get | /microservice/{name} | Returns the named microservice |
| delete | /microservice/{name} | Deletes the named microservice |

##### /microservice

###### GET /microservice

Returns a list of all microservicees

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of microserviceses | array[[Microservice](#microservice)] |
| 400 | No Microservice found | Empty |
| 401 | Not authorized | Empty |

###### PUT /microservice

Sets the named microservice

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Microservice updated | Empty |
| 400 | Error updating microservice | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The new microservice to create | True | [Microservice](#microservice) |  |

##### /microservice/{name}

###### GET /microservice/{name}

Returns the named microservice

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returns the microservice | [Microservice](#microservice) |
| 400 | No Microservice found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named microservice could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the microservice | True | String |

###### DELETE /microservice/{name}

Deletes an microservice by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Microservice found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named microservice could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the microservice | True | String |

#### microservice.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: Microservice  
 description: |-  
   
 As part of microservices, a function with parameters that can be  
 invoked has been defined.  
  
 termsOfService: "https://github.com/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /microservice:  
 get:  
 tags:  
 - Microservice  
 summary: Returns a list of microservicees  
 description: Returns a list of all microservicees  
 operationId: cloudmesh.microservice.list  
 responses:  
 '200':  
 description: The list of microserviceses  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Microservice'  
 '400':  
 description: No Microservice found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Microservice  
 summary: Set an microservice  
 description: Sets the named microservice  
 operationId: cloudmesh.microservice.add  
 requestBody:  
 description: The new microservice to create  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Microservice'  
 responses:  
 '200':  
 description: Microservice updated  
 '400':  
 description: Error updating microservice  
 '401':  
 description: Not authorized  
 /microservice/{name}:  
 get:  
 tags:  
 - Microservice  
 summary: Returns the named microservice  
 description: Returns the named microservice  
 operationId: cloudmesh.microservice.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the microservice  
 responses:  
 '200':  
 description: Returns the microservice  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Microservice'  
 '400':  
 description: No Microservice found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named microservice could not be found  
 delete:  
 tags:  
 - Microservice  
 summary: Deletes the named microservice  
 description: Deletes an microservice by name  
 operationId: cloudmesh.microservice.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the microservice  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Microservice found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named microservice could not be found  
components:  
 schemas:  
 Microservice:  
 type: object  
 description: The microservice  
 properties:  
 name:  
 type: string  
 description: Name of the microservice  
 endpoint:  
 type: string  
 description: The end point of the microservice  
 function:  
 type: string  
 description: The function the microservice represents  
 description:  
 type string:  
 description: The description of the microservice

## Reservation

### Reservation

Some services may consume a considerable amount of resources, necessitating the reservation of resources.

#### Schema Reservation

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/reservation.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the reservation |
| service | string | The name of the service on which we reserve |
| description | string | The description of the reservation |
| start | string | The start time and date |
| end | string | The end time and date |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /reservation | Returns a list of reservations |
| put | /reservation | Uploads a reservation to the list of reservations |
| get | /reservation/{name} | Returns the named reservation |
| delete | /reservation/{name} | Deletes the named reservation |

##### /reservation

###### GET /reservation

Returns a list of all reservations

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of reservations | array[[Reservation](#reservation)] |
| 400 | No Reservations found | Empty |

###### PUT /reservation

Uploads a reservation to the list of reservations

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Reservation updated | Empty |
| 400 | Error updating reservation | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The reservation to be uploaded | True | [Reservation](#reservation) |  |

##### /reservation/{name}

###### GET /reservation/{name}

Returns an reservation by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the reservation | [Reservation](#reservation) |
| 400 | No reservation found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named reservation could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the reservation | True | String |

###### DELETE /reservation/{name}

Deletes an reservation by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No reservation found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named reservation could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the reservation | True | String |

#### reservation.yaml

openapi: '3.0.2'  
info:  
 version: 3.2.0  
 x-date: 14-06-2019  
 x-status: defined  
 title: Reservation  
 description: |-  
   
 Some services may consume a considerable amount of resources,  
 necessitating the reservation of resources.  
   
 termsOfService: 'https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt'  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /reservation:  
 get:  
 tags:  
 - Reservation  
 summary: Returns a list of reservations  
 description: Returns a list of all reservations  
 operationId: cloudmesh.reservation.list  
 responses:  
 '200':  
 description: The list of reservations  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Reservation'  
 '400':  
 description: No Reservations found  
 put:  
 tags:  
 - Reservation  
 summary: Uploads a reservation to the list of reservations  
 description: Uploads a reservation to the list of reservations  
 operationId: cloudmesh.reservation.add  
 requestBody:  
 description: The reservation to be uploaded  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Reservation'  
 responses:  
 '200':  
 description: Reservation updated  
 '400':  
 description: Error updating reservation  
 /reservation/{name}:  
 get:  
 tags:  
 - Reservation  
 summary: Returns the named reservation  
 description: Returns an reservation by name  
 operationId: cloudmesh.reservation.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the reservation  
 responses:  
 '200':  
 description: Returning the information of the reservation  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Reservation'  
 '400':  
 description: No reservation found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named reservation could not be found  
 delete:  
 tags:  
 - Reservation  
 summary: Deletes the named reservation  
 description: Deletes an reservation by name  
 operationId: cloudmesh.reservation.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the reservation  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No reservation found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named reservation could not be found  
components:  
 schemas:  
 Reservation:  
 type: object  
 description: The reservation  
 properties:  
 name:  
 type: string  
 description: Name of the reservation  
 service:  
 type: string  
 description: The name of the service on which we reserve  
 description:  
 type: string  
 description: The description of the reservation  
 start:  
 type: string  
 format: date  
 description: The start time and date  
 end:  
 type: string  
 format: date  
 description: The end time and date

## Data Streams

### Stream

The stream object describes a data flow, providing information about the rate and number of items exchanged while issuing requests to the stream. A stream may return data items in a specific format that is defined by the stream.

#### Schema Stream

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/stream.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the stream |
| format | string | Format of the stream |
| rate | integer | The rate of messages |
| limit | integer | The limit of items send |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /stream | Returns a list of streams |
| put | /stream | Set an stream |
| get | /stream/{name} | Returns the named stream |
| delete | /stream/{name} | Deletes the named stream |

##### /stream

###### GET /stream

Returns a list of all streams

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of streamses | array[[Stream](#stream)] |
| 400 | No Stream found | Empty |
| 401 | Not authorized | Empty |

###### PUT /stream

Sets the named stream

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Stream updated | Empty |
| 400 | Error updating stream | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The new stream to create | True | [Stream](#stream) |  |

##### /stream/{name}

###### GET /stream/{name}

Returns an stream by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the stream | [Stream](#stream) |
| 400 | No Stream found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named stream could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the stream | True | String |

###### DELETE /stream/{name}

Deletes an stream by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Stream found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named stream could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the stream | True | String |

#### stream.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: TODO  
 title: Stream  
 description: |-  
   
 The stream object describes a data flow, providing information  
 about the rate and number of items exchanged while issuing requests  
 to the stream. A stream may return data items in a specific format  
 that is defined by the stream.  
  
   
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /stream:  
 get:  
 tags:  
 - Stream  
 summary: Returns a list of streams  
 description: Returns a list of all streams  
 operationId: cloudmesh.stream.list  
 responses:  
 '200':  
 description: The list of streamses  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Stream'  
 '400':  
 description: No Stream found  
 '401':  
 description: Not authorized  
 put:  
 tags:  
 - Stream  
 summary: Set an stream  
 description: Sets the named stream  
 operationId: cloudmesh.stream.add  
 requestBody:  
 description: The new stream to create  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Stream'  
 responses:  
 '200':  
 description: Stream updated  
 '400':  
 description: Error updating stream  
 '401':  
 description: Not authorized  
 /stream/{name}:  
 get:  
 tags:  
 - Stream  
 summary: Returns the named stream  
 description: Returns an stream by name  
 operationId: cloudmesh.stream.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the stream  
 responses:  
 '200':  
 description: Returning the information of the stream  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Stream'  
 '400':  
 description: No Stream found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named stream could not be found  
 delete:  
 tags:  
 - Stream  
 summary: Deletes the named stream  
 description: Deletes an stream by name  
 operationId: cloudmesh.stream.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the stream  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Stream found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named stream could not be found  
components:  
 schemas:  
 Stream:  
 type: object  
 description: The stream  
 properties:  
 name:  
 type: string  
 description: Name of the stream  
 format:  
 type: string  
 description: Format of the stream  
 rate:  
 type: integer  
 description: The rate of messages  
 limit:  
 type: integer  
 description: The limit of items send

### Filter

Filters can operate on a variety of objects and reduce the information received based on a search criterion.

#### Schema Filter

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/filter.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of the filter |
| function | string | The function of the data exchanged in the stream |
| kind | string | The filter kind or type |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /filter | Returns a list of filteres |
| put | /filter | Set an filter |
| get | /filter/{name} | Returns the named filter |
| delete | /filter/{name} | Deletes the named filter |

##### /filter

###### GET /filter

Returns a list of all filteres

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of filterses | array[[Filter](#filter)] |
| 400 | No Filter found | Empty |

###### PUT /filter

Sets the named filter

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Filter updated | Empty |
| 401 | Not authorized | Empty |
| 400 | Error updating filter | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The new filter to create | True | [Filter](#filter) |  |

##### /filter/{name}

###### GET /filter/{name}

Returns an filter by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the filter | [Filter](#filter) |
| 400 | No Filter found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named filter could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the filter | True | String |

###### DELETE /filter/{name}

Deletes an filter by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Filter found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named filter could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the filter | True | String |

#### filter.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: TODO  
 title: Filter  
 description: |-  
   
 Filters can operate on a variety of objects and reduce the  
 information received based on a search criterion.  
   
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /filter:  
 get:  
 tags:  
 - Filter  
 summary: Returns a list of filteres  
 description: Returns a list of all filteres  
 operationId: cloudmesh.filter.list  
 responses:  
 '200':  
 description: The list of filterses  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Filter'  
 '400':  
 description: No Filter found  
 put:  
 tags:  
 - Filter  
 summary: Set an filter  
 description: Sets the named filter  
 operationId: cloudmesh.filter.add  
 requestBody:  
 description: The new filter to create  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Filter'  
 responses:  
 '200':  
 description: Filter updated  
 '401':  
 description: Not authorized  
 '400':  
 description: Error updating filter  
 /filter/{name}:  
 get:  
 tags:  
 - Filter  
 summary: Returns the named filter  
 description: Returns an filter by name  
 operationId: cloudmesh.filter.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the filter  
 responses:  
 '200':  
 description: Returning the information of the filter  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Filter'  
 '400':  
 description: No Filter found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named filter could not be found  
 delete:  
 tags:  
 - Filter  
 summary: Deletes the named filter  
 description: Deletes an filter by name  
 operationId: cloudmesh.filter.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the filter  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Filter found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named filter could not be found  
components:  
 schemas:  
 Filter:  
 type: object  
 description: The filter  
 properties:  
 name:  
 type: string  
 description: Name of the filter  
 function:  
 type: string  
 description: The function of the data exchanged in the stream  
 kind:  
 type: string  
 description: The filter kind or type

## Deployment

### Deployment

A service to store software stack deployments.

#### Schema Deployment

[Reference: ☁️](https://github.com/cloudmesh/cloudmesh-nist/blob/master/spec/deployment.yaml)

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| kind | string | the kind of the deployment |
| specification | string | the specification of the deployment |

#### Paths

|  |  |  |
| --- | --- | --- |
| HTTP | Path | Summary |
| get | /deployment | Returns a list of deploymentes |
| put | /deployment | Set an deployment |
| get | /deployment/{name} | Returns the named deployment |
| delete | /deployment/{name} | Deletes the named deployment |

##### /deployment

###### GET /deployment

Returns a list of all deploymentes

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | The list of deploymentses | array[[Deployment](#deployment)] |
| 400 | No Deployment found | Empty |

###### PUT /deployment

Sets the named deployment

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deployment updated | Empty |
| 400 | Error updating deployment | Empty |
| 401 | Not authorized | Empty |

Request Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Located in | Description | Required | Schema |  |
| Body | The new deployment to create | True | [Deployment](#deployment) |  |

##### /deployment/{name}

###### GET /deployment/{name}

Returns an deployment by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Returning the information of the deployment | [Deployment](#deployment) |
| 400 | No Deployment found | Empty |
| 404 | The named deployment could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the deployment | True | String |

###### DELETE /deployment/{name}

Deletes an deployment by name

Responses

|  |  |  |
| --- | --- | --- |
| Code | Description | Schema |
| 200 | Deletion successful | Empty |
| 400 | No Deployment found | Empty |
| 401 | Not authorized | Empty |
| 404 | The named deployment could not be found | Empty |

Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Located in | Description | Required | Schema |
| name | path | The name of the deployment | True | String |

#### deployment.yaml

openapi: "3.0.2"  
info:  
 version: 3.2.0  
 x-date: 08-06-2019  
 x-status: TODO  
 title: Deployment  
 description: |-  
   
 A service to store software stack deployments.  
   
 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"  
 contact:  
 name: NIST BDRA Interface Subgroup  
 url: https://cloudmesh-community.github.io/nist  
 license:  
 name: Apache 2.0  
 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt  
servers:  
 - url: /cloudmesh/v3  
paths:  
 /deployment:  
 get:  
 tags:  
 - Deployment  
 summary: Returns a list of deploymentes  
 description: Returns a list of all deploymentes  
 operationId: cloudmesh.deployment.list  
 responses:  
 '200':  
 description: The list of deploymentses  
 content:  
 application/json:  
 schema:  
 type: array  
 items:  
 $ref: '#/components/schemas/Deployment'  
 '400':  
 description: No Deployment found  
 put:  
 tags:  
 - Deployment  
 summary: Set an deployment  
 description: Sets the named deployment  
 operationId: cloudmesh.deployment.add  
 requestBody:  
 description: The new deployment to create  
 required: true  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Deployment'  
 responses:  
 '200':  
 description: Deployment updated  
 '400':  
 description: Error updating deployment  
 '401':  
 description: Not authorized  
 /deployment/{name}:  
 get:  
 tags:  
 - Deployment  
 summary: Returns the named deployment  
 description: Returns an deployment by name  
 operationId: cloudmesh.deployment.find\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the deployment  
 responses:  
 '200':  
 description: Returning the information of the deployment  
 content:  
 application/json:  
 schema:  
 $ref: '#/components/schemas/Deployment'  
 '400':  
 description: No Deployment found  
 '404':  
 description: The named deployment could not be found  
 delete:  
 tags:  
 - Deployment  
 summary: Deletes the named deployment  
 description: Deletes an deployment by name  
 operationId: cloudmesh.deployment.delete\_by\_name  
 parameters:  
 - name: name  
 in: path  
 required: true  
 schema:  
 type: string  
 description: The name of the deployment  
 responses:  
 '200':  
 description: Deletion successful  
 '400':  
 description: No Deployment found  
 '401':  
 description: Not authorized  
 '404':  
 description: The named deployment could not be found  
components:  
 schemas:  
 Deployment:  
 type: object  
 description: the deployment  
 properties:  
 kind:  
 type: string  
 description: the kind of the deployment  
 specification:  
 type: string  
 description: the specification of the deployment

# Status Codes and Error Responses

In case of an error or a successful response, the response header contains a HTTP code (see <https://tools.ietf.org/html/rfc7231>). The response body usually contains the following:

* The HTTP response code;
* An accompanying message for the HTTP response code; and
* A field or object where the error occurred.

Table 1: HTTP Response Codes

|  |  |
| --- | --- |
| HTTP Response | Description Code |
| 200 | *OK* success code, for GET or HEAD request. |
| 201 | *Created* success code, for POST request. |
| 204 | *No Content* success code, for DELETE request. |
| 300 | The value returned when an external ID exists in more than one record. |
| 304 | The request content has not changed since a specified date and time. |
| 400 | The request could not be understood. |
| 401 | The session ID or OAuth token used has expired or is invalid. |
| 403 | The request has been refused. |
| 404 | The requested resource could not be found. |
| 405 | The method specified in the Request-Line isn’t allowed for the resource specified in the URI. |
| 415 | The entity in the request is in a format that’s not supported by the specified method. |

# Acronyms and Terms

The following acronyms and terms are used in this volume.

ACID

Atomicity, Consistency, Isolation, Durability

API

Application Programming Interface

ASCII

American Standard Code for Information Interchange

BASE

Basically Available, Soft state, Eventual consistency

Container

See <http://csrc.nist.gov/publications/drafts/800-180/sp800-180_draft.pdf>

Cloud Computing

The practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer. See <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>.

DevOps

A clipped compound of software DEVelopment and information technology OPerationS

Deployment

The action of installing software on resources

HTTP

HyperText Transfer Protocol HTTPS HTTP Secure

Hybrid

Cloud See <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>.

IaaS

Infrastructure as a Service SaaS Software as a Service

ITL

Information Technology Laboratory

Microservice Architecture

Is an approach to build applications based on many smaller modular services. Each module supports a specific goal and uses a simple, well-defined interface to communicate with other sets of services.

NBD-PWG

NIST Big Data Public Working Group

NBDRA

NIST Big Data Reference Architecture

NBDRAI

NIST Big Data Reference Architecture Interface

NIST

National Institute of Standards and Technology

OS

Operating System

REST

REpresentational State Transfer

Replica

A duplicate of a file on another resource to avoid costly transfer costs in case of frequent access.

Serverless Computing

Serverless computing specifies the paradigm of function as a service (FaaS). It is a cloud computing code execution model in which a cloud provider manages the function deployment and utilization while clients can utilize them. The charge model is based on execution of the function rather than the cost to manage and host the VM or container.

Software Stack

A set of programs and services that are installed on a resource to support applications.

Virtual File System

An abstraction layer on top of a distributed physical file system to allow easy access to the files by the user or application.

Virtual Machine

A VM is a software computer that, like a physical computer, runs an operating system and applications. The VM is composed of a set of specification and configuration files and is backed by the physical resources of a host.

Virtual Cluster

A virtual cluster is a software cluster that integrate either VMs, containers, or physical resources into an agglomeration of compute resources. A virtual cluster allows users to authenticate and authorize to the virtual compute nodes to utilize them for calculations. Optional high-level services that can be deployed on a virtual cluster may simplify interaction with the virtual cluster or provide higher-level services.

Workflow

The sequence of processes or tasks

WWW

World Wide Web

# Bibliography

Department of Defense. 2010. “The Dodaf Architecture Framework Version 2.02.” Report 2.02. Department of Defense. <https://dodcio.defense.gov/library/dod-architecture-framework/>.

Internet2 Middleware Architecture Committee for Education. 2016. “EduPerson Object Class Specification.” 201602. 2016: Internet2. <http://software.internet2.edu/eduperson/internet2-mace-dir-eduperson-201602.html>.

Laszewski, Gregor von. 2019a. “Configuration File Example: Coudmesh4.yaml.” GitHub. <https://github.com/cloudmesh/cloudmesh-cloud/blob/master/cloudmesh/etc/cloudmesh4.yaml>.

———. 2019b. “NIST Bdra Vol 8. GitHub Issues.” GitHub. <https://github.com/cloudmesh/cloudmesh-nist/issues>.

———. 2019c. “NIST Bdra Vol 8. OpenAPI Specifications.” GitHub. <https://github.com/cloudmesh-community/nist/tree/master/spec>.

———. 2019d. “Nist Bdra Volume 8 Github History.” GitHub. <https://github.com/cloudmesh/cloudmesh-nist/commits/master/docs/nistvol8-2.epub>.

Laszewski, Gregor von, Wo Chang, Fugang Wang, Badi Abdhul Wahid, Geoffrey C. Fox, Pratik Thakkar, Alicia Mara Zuniga-Alvarado, and Robert C. Whetsel. 2015. “NIST Big Data Interoperability Framework: Volume 8, Reference Architecture Interfaces.” Special Publication (NIST SP). Gaithersburg, MD: National Institute of Standards; Technology (NIST). <https://github.com/cloudmesh/cloudmesh-nist/raw/master/history/NIST.SP.1500-9.pdf>.

Laszewski, Gregor von, Fugang Wang, Badi Abdul-Wahid, Hyungro Lee, Geoffrey C. Fox, and and Wo Chang. 2017. “Cloudmesh in Support of the Nist Big Data Architecture Framework.” Indiana University. <https://github.com/cyberaide/nist/blob/master/vonLaszewski-nist.pdf>.

NIST. 2015a. “21.0 Final Version Page of the Nbd-Pwg Website.” Web Page. <https://bigdatawg.nist.gov/V2_output_docs.php>.

———. 2015b. “V1.0 Final Version Page of the Nbd-Pwg Website.” Web Page. <https://bigdatawg.nist.gov/V1_output_docs.php>.

NIST. n.d. “Big Data Public Working Group (NBD-PWG).” <https://bigdatawg.nist.gov/>.

The White House Office of Science and Technology Policy. 2014. “Big Data is a Big Deal.” OSTP Blog. <http://www.whitehouse.gov/blog/2012/03/29/big-data-big-deal>.

W. L. Chang (Co-Chair), A. Roy (Subgroup Co-chair), M. Underwood (Subgroup Co-chair), and NIST Big Data Public Working Group. 2019. “NIST Big Data Interoperability Framework: Volume 4, Big Data Security and Privacy (NIST SP 1500-4 VERSION 3).” Gaithersburg, MD: National Institute of Standards; Technology (NIST). <https://bigdatawg.nist.gov/show_InputDoc.php>.

W. L. Chang (Co-Chair), D. Boyd (Subgroup Co-chair), O. Levin (Version 1 Subgroup Co-Chair), and NIST Big Data Public Working Group. 2019. “NIST Big Data Interoperability Framework: Volume 6, Big Data Reference Architecture (NIST SP 1500-6 VERSION 3).” Gaithersburg, MD: National Institute of Standards; Technology (NIST). <https://bigdatawg.nist.gov/show_InputDoc.php>.

W. L. Chang (Co-Chair), G. Fox (Subgroup Co-chair), and NIST Big Data Public Working Group. 2019. “NIST Big Data Interoperability Framework: Volume 3, Big Data Use Cases and General Requirements (NIST SP 1500-3 VERSION 3).” Gaithersburg, MD: National Institute of Standards; Technology (NIST). <https://bigdatawg.nist.gov/show_InputDoc.php>.

W. L. Chang (Co-Chair), G. von Laszewski (Editor), and NIST Big Data Public Working Group. 2019. “NIST Big Data Interoperability Framework: Volume 8, Big Data Reference Architecture Interfaces (NIST SP 1500-9 VERSION 2).” Gaithersburg, MD: National Institute of Standards; Technology (NIST). <https://bigdatawg.nist.gov/show_InputDoc.php>.

W. L. Chang (Co-Chair), N. Grady (Subgroup Co-chair), and NIST Big Data Public Working Group. 2019a. “NIST Big Data Interoperability Framework: Volume 1, Big Data Definitions (NIST SP 1500-1 VERSION 3).” Gaithersburg, MD: National Institute of Standards; Technology (NIST). <https://bigdatawg.nist.gov/show_InputDoc.php>.

———. 2019b. “NIST Big Data Interoperability Framework: Volume 2, Big Data Taxonomies (NIST SP 1500-2 VERSION 3).” Gaithersburg, MD: National Institute of Standards; Technology (NIST). <https://bigdatawg.nist.gov/show_InputDoc.php>.

W. L. Chang (Co-Chair), R. Reinsch (Subgroup Co-chair), C. Austin (Editor), and NIST Big Data Public Working Group. 2019. “NIST Big Data Interoperability Framework: Volume 9, Adoption and Modernization (NIST SP 1500-10 VERSION 2).” Gaithersburg, MD: National Institute of Standards; Technology (NIST). <https://bigdatawg.nist.gov/show_InputDoc.php>.

W. L. Chang (Co-Chair), R. Reinsch (Subgroup Co-chair), D. Boyd (Version 1 Subgroup Co-chair), C. Buffington (Version 1 Subgroup Co-chair), and NIST Big Data Public Working Group. 2019. “NIST Big Data Interoperability Framework: Volume 7, Big Data Standards Roadmap (NIST SP 1500-7 VERSION 3).” Gaithersburg, MD: National Institute of Standards; Technology (NIST). <https://bigdatawg.nist.gov/show_InputDoc.php>.

W. L. Chang (Co-Chair), S. Mishra (Editor), and NIST Big Data Public Working Group. 2019. “NIST Big Data Interoperability Framework: Volume 5, Big Data Architectures White Paper Survey (NIST SP 1500-5 VERSION 1).” Gaithersburg, MD: National Institute of Standards; Technology (NIST). <https://bigdatawg.nist.gov/show_InputDoc.php>.