

NIST Special Publication 1500-9r1

**NIST Big Data Interoperability
Framework:
Volume 8, Reference Architecture
Interfaces**

Version 3

NIST Big Data Public Working Group
Definitions and Taxonomies Subgroup

This publication is available free of charge from:
<https://doi.org/10.6028/NIST.SP.1500-9r1>

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

NIST Special Publication 1500-9r1

**NIST Big Data Interoperability
Framework:
Volume 8, Reference Architecture
Interfaces**

Version 3

NIST Big Data Public Working Group
Definitions and Taxonomies Subgroup
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

This publication is available free of charge from:
<https://doi.org/10.6028/NIST.SP.1500-9r1>

October 2019



U.S. Department of Commerce
Wilbur L. Ross, Jr., Secretary

National Institute of Standards and Technology
Walter Copan, NIST Director and Undersecretary of Commerce for Standards and Technology

National Institute of Standards and Technology (NIST) Special Publication 1500-9r1
168 pages (October 2019)

NIST Special Publication series 1500 is intended to capture external perspectives related to NIST standards, measurement, and testing-related efforts. These external perspectives can come from industry, academia, government, and others. These reports are intended to document external perspectives and do not represent official NIST positions.

Certain commercial entities, equipment, or materials may be identified in this document to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all publications during public comment periods and provide feedback to NIST. All NIST publications are available at <http://www.nist.gov/publication-portal.cfm>.

Copyrights and Permissions

Official publications of the National Institute of Standards and Technology are not subject to copyright in the United States. Foreign rights are reserved. Questions concerning the possibility of copyrights in foreign countries should be referred to the Office of Chief Counsel at NIST via email to nistcounsel@nist.gov.

Comments on this publication may be submitted to Wo Chang

National Institute of Standards and Technology
Attn: Wo Chang, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8900) Gaithersburg, MD 20899-8930
Email: SP1500comments@nist.gov

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at NIST promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems. This document reports on ITL's research, guidance, and outreach efforts in Information Technology and its collaborative activities with industry, government, and academic organizations.

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 (San Diego Supercomputer Center), 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 (Vistrionix), and Dan McClary (Oracle), for the Standards Roadmap Subgroup.

The editors for this document were the following:

- **Version 1:** This volume resulted from Stage 2 work and was not part of the Version 1 scope.
- **Version 2:** Gregor von Laszewski (Indiana University) and Wo Chang (NIST).
- **Version 3:** Gregor von Laszewski (Indiana University) and 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 contributions¹ to this volume, during Version 1, Version 2, and/or Version 3 activities, by the following NBD-PWG members:

Wo Chang <i>National Institute of Standard and Technology</i>	Badi Abdhul Wahid <i>Indiana University</i>
Geoffrey C. Fox <i>Indiana University</i>	Fugang Wang <i>Indiana University</i>
Pratik Thakkar <i>Philips</i>	Robert C. Whetsel <i>DISA/NBIS</i>
Gregor von Laszewski <i>Indiana University</i>	Alicia Zuniga-Alvarado <i>Consultant</i>

¹ “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.

TABLE OF CONTENTS

1	INTRODUCTION	1
1.1	BACKGROUND	1
1.2	SCOPE AND OBJECTIVES OF THE REFERENCE ARCHITECTURES SUBGROUP	3
1.3	REPORT PRODUCTION	3
1.4	REPORT STRUCTURE	4
2	NBDRA INTERFACE REQUIREMENTS	6
2.1	HIGH-LEVEL REQUIREMENTS OF THE INTERFACE APPROACH	7
2.1.1	<i>Technology- and Vendor-Agnostic</i>	7
2.1.2	<i>Support of Plug-In Compute Infrastructure</i>	7
2.1.3	<i>Orchestration of Infrastructure and Services</i>	8
2.1.4	<i>Orchestration of Big Data Applications and Experiments</i>	8
2.1.5	<i>Reusability</i>	8
2.1.6	<i>Execution Workloads</i>	8
2.1.7	<i>Security and Privacy Fabric Requirements</i>	9
2.2	COMPONENT-SPECIFIC INTERFACE REQUIREMENTS	9
2.2.1	<i>System Orchestrator Interface Requirements</i>	9
2.2.2	<i>Data Provider Interface Requirements</i>	10
2.2.3	<i>Data Consumer Interface Requirements</i>	10
2.2.4	<i>Big Data Application Provider Interface Requirements</i>	10
2.2.5	<i>Big Data Framework Provider Interface Requirements</i>	12
2.2.6	<i>Big Data Application Provider to Big Data Framework Provider Interface</i>	13
3	SPECIFICATION PARADIGM	14
3.1	HYBRID AND MULTIPLE FRAMEWORKS	14
3.2	DESIGN BY RESOURCE-ORIENTED ARCHITECTURE	14
3.3	INTERFACE COMPLIANCY.....	14
4	SPECIFICATION.....	15
4.1	LIST OF SPECIFICATIONS	15
4.2	AUTHENTICATION.....	17
4.3	STATUS CODES AND ERROR RESPONSES.....	17
4.4	TIMESTAMP	18
4.4.1	<i>Timestamp</i>	18
4.5	IDENTITY	20
4.5.1	<i>Organization</i>	20
4.5.2	<i>User</i>	32
4.5.3	<i>Account</i>	36
4.5.4	<i>Public Key Store</i>	40
4.6	VARIABLE, DEFAULT, AND ALIAS.....	43
4.6.1	<i>Alias</i>	43
4.6.2	<i>Variables</i>	47
4.6.3	<i>Default</i>	50
4.7	DATA MANAGEMENT.....	54
4.7.1	<i>Filestore</i>	54
4.7.2	<i>Replica</i>	58

4.7.3	Database.....	61
4.7.4	Virtual Directory.....	69
4.8	COMPUTE MANAGEMENT: VIRTUAL CLUSTERS	75
4.8.1	Virtual Cluster	75
4.8.2	Network of Nodes	82
4.8.3	Scheduler	90
4.8.4	Queue.....	96
4.9	COMPUTE MANAGEMENT: VIRTUAL MACHINES.....	102
4.9.1	Image.....	102
4.9.2	Flavor	107
4.9.3	Virtual Machine	111
4.9.4	Secgroup	116
4.9.5	Nic.....	123
4.10	COMPUTE MANAGEMENT: CONTAINERS.....	126
4.10.1	Containers	126
4.11	COMPUTE MANAGEMENT: MAPREDUCE	130
4.11.1	MapReduce.....	130
4.12	COMPUTE MANAGEMENT: FUNCTIONS	137
4.12.1	Microservice	137
4.13	RESERVATION	140
4.13.1	Reservation.....	140
4.14	DATA STREAMS	144
4.14.1	Stream	144
4.14.2	Filter.....	148
4.15	DEPLOYMENT	151
4.15.1	Deployment	151
APPENDIX A: ACRONYMS AND TERMS.....		156
APPENDIX B: BIBLIOGRAPHY		158

FIGURES

FIGURE 1: NBDIF DOCUMENTS NAVIGATION DIAGRAM PROVIDES CONTENT FLOW BETWEEN VOLUMES	5
FIGURE 2: NIST BIG DATA REFERENCE ARCHITECTURE (NBDRA).....	6
FIGURE 3: PROVIDER VIEW.....	16
FIGURE 4: RESOURCE VIEW	17

TABLES

TABLE 1: SPECIFICATIONS.....	15
TABLE 2: HTTP RESPONSE CODES	18

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 interfaces 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 forth 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 a 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 NBD-PWG [1].

The 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 [2]
- Volume 2, Taxonomies [3]
- Volume 3, Use Cases and General Requirements [4]
- Volume 4, Security and Privacy [5]
- Volume 5, Architectures White Paper Survey [6]
- Volume 6, Reference Architecture [7]
- Volume 7, Standards Roadmap [8]
- Volume 8, Reference Architecture Interfaces (this volume)
- Volume 9, Adoption and Modernization [9]

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 (https://bigdatawg.nist.gov/V1_output_docs.php).

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

44 general interfaces, and demonstrate how the NBDRA can be used. As a result of the Stage 2 work, the
45 need for NBDIF Volume 8 and NBDIF Volume 9 was identified and the two new volumes were created.
46 Version 2 of the NBDIF volumes, resulting from Stage 2 work, can be downloaded from the V2.0 Final
47 Version page of the NBD-PWG website (https://bigdatawg.nist.gov/V2_output_docs.php).

48 This document is the result of Stage 3 work of the NBD-PWG. Coordination of the group is conducted on
49 the NBD-PWG web page [1].

50

1 INTRODUCTION

1.1 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 [10]. 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

92 for the development of a Big Data Standards Roadmap. Forum participants noted that this roadmap
 93 should define and prioritize Big Data requirements, including interoperability, portability, reusability,
 94 extensibility, data usage, analytics, and technology infrastructure. In doing so, the roadmap would
 95 accelerate the adoption of the most secure and effective Big Data techniques and technology.

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

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

112 Stage 3: Validate the NBDRA by building Big Data general applications through the general
 113 interfaces;

114 Stage 2: Define general interfaces between the NBDRA components; and

115 Stage 1: Identify the high-level Big Data reference architecture key components, which are
 116 technology-, infrastructure-, and vendor-agnostic.

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

- 119 • Volume 1, Definitions [11]
- 120 • Volume 2, Taxonomies (this volume)
- 121 • Volume 3, Use Cases and General Requirements [12]
- 122 • Volume 4, Security and Privacy [13]
- 123 • Volume 5, Architectures White Paper Survey [6]
- 124 • Volume 6, Reference Architecture [14]
- 125 • Volume 7, Standards Roadmap [15]
- 126 • Volume 8, Reference Architecture Interfaces [16]
- 127 • Volume 9, Adoption and Modernization [17]

128 During Stage 1, Volumes 1 through 7 were conceptualized, organized, and written. The finalized Version
 129 1 documents can be downloaded from the V1.0 Final Version page of the NBD-PWG website
 130 (https://bigdatawg.nist.gov/V1_output_docs.php).

131 During Stage 2, the NBD-PWG developed Version 2 of the NBDIF Version 1 volumes, with the
 132 exception of Volume 5, which contained the completed architecture survey work that was used to inform
 133 Stage 1 work of the NBD-PWG. The goals of Version 2 were to enhance the Version 1 content, define
 134 general interfaces between the NBDRA components by aggregating low-level interactions into high-level
 135 general interfaces, and demonstrate how the NBDRA can be used. As a result of the Stage 2 work, the
 136 need for NBDIF Volume 8 and NBDIF Volume 9 was identified and the two new volumes were created.
 137 Version 2 of the NBDIF volumes, resulting from Stage 2 work, can be downloaded from the V2.0 Final
 138 Version page of the NBD-PWG website (https://bigdatawg.nist.gov/V2_output_docs.php).

1.2 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” [18]. 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 Section 1.1.

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.

The following is a list of milestone releases for this volume:

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

190 1.4 REPORT STRUCTURE

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

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

- 206 • Section 2 presents the interface requirements;
- 207 • Section 3 presents the specification paradigm that is used; and
- 208 • Section 4 presents several objects grouped by functional use, with a summary table of selected
209 proposed objects in Section 4.1.

210 While each NBDIF volume was created with a specific focus within Big Data, all volumes are
211 interconnected. During the creation of the volumes, information from some volumes was used as input for
212 other volumes. Broad topics (e.g., definition, architecture) may be discussed in several volumes with each
213 discussion circumscribed by the volume's particular focus. Arrows shown in Figure 1 indicate the main
214 flow of information input and/or output from the volumes. Volumes 2, 3, and 5 (blue circles) are
215 essentially standalone documents that provide output to other volumes (e.g., to Volume 6). These
216 volumes contain the initial situational awareness research. During the creation of Volumes 4, 7, 8, and 9
217 (green circles), input from other volumes was used. The development of these volumes took into account
218 work on the other volumes. Volumes 1 and 6 (red circles) were developed using the initial situational
219 awareness research and continued to be modified based on work in other volumes. The information from
220 these volumes was also used as input to the volumes in the green circles.

221



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

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

222

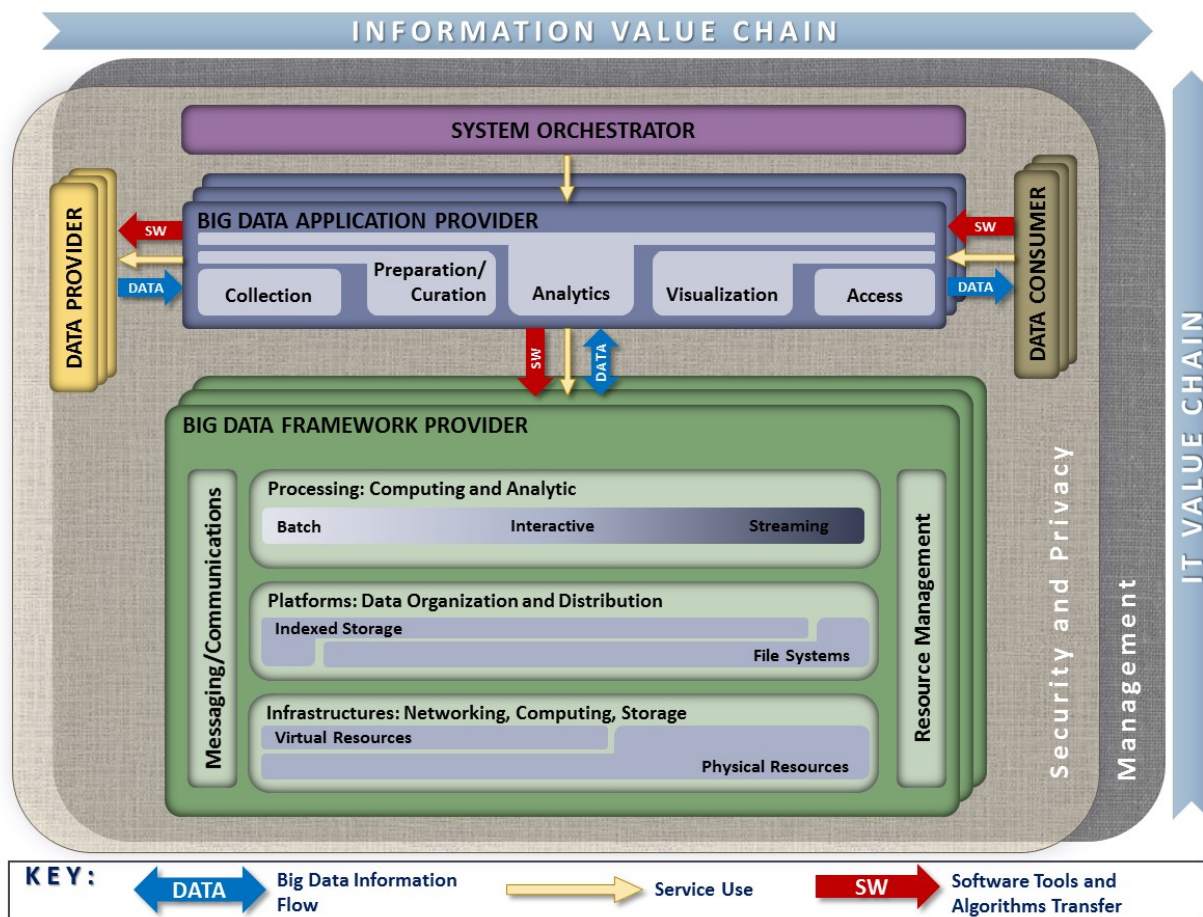
223

224

225

2 NBDRA INTERFACE REQUIREMENTS

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



234

235

Figure 2: NIST Big Data Reference Architecture (NBDRA)

236 This vendor-neutral, technology- and infrastructure-agnostic conceptual model, the NBDRA, is shown in
 237 Figure 2 and represents a Big Data system composed of five logical functional components connected by
 238 interoperability interfaces (i.e., services). Two fabrics envelop the components, representing the
 239 interwoven nature of management and security and privacy with all five of the components. These two
 240 fabrics provide services and functionality to the five main roles in the areas specific to Big Data and are
 241 crucial to any Big Data solution. Note: None of the terminology or diagrams in these documents is

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

242 intended to be normative or to imply any business or deployment model. The terms *provider* and
243 *consumer* as used are descriptive of general roles and are meant to be informative in nature.

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

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

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

264 Prior to outlining the specific interfaces, the general requirements are introduced and the interfaces are
265 defined in the following sections.

266 **2.1 HIGH-LEVEL REQUIREMENTS OF THE INTERFACE** 267 **APPROACH**

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

270 **2.1.1 TECHNOLOGY- AND VENDOR-AGNOSTIC**

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

276 **2.1.2 SUPPORT OF PLUG-IN COMPUTE INFRASTRUCTURE**

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

- 281 • Access to hierarchy of compute resources from the laptop/desktop, servers, data clusters, and
282 clouds;

- 283 • The ability to integrate special-purpose hardware such as graphics processing units (GPUs) and
- 284 field-programmable gate arrays (FPGAs) that are used in accelerated analysis of data; and
- 285 • The integration of services including microservices that allow the analysis of the data by
- 286 delegating them to hosted or dynamically deployed services on the infrastructure of choice.

287 **2.1.3 ORCHESTRATION OF INFRASTRUCTURE AND SERVICES**

288 From review of the use case collection, presented in the *NBDIF: Volume 3, Use Cases and General*
 289 *Requirements* document [4], the need arose to address the mechanism of preparing suitable infrastructures
 290 for various use cases. As not every infrastructure is suited for every use case, a custom infrastructure may
 291 be needed. As such, this document is not attempting to deliver a single deployed NBDRA, but allow the
 292 setup of an infrastructure that satisfies the particular use case. To achieve this task, it is necessary to
 293 provision software stacks and services while orchestrating their deployment and leveraging
 294 infrastructures. It is not the focus of this document to replace existing orchestration software and services,
 295 but provide an interface to them to leverage them as part of defining and creating the infrastructure.
 296 Various orchestration frameworks and services could therefore be leveraged, even as part of the same
 297 framework, and work in orchestrated fashion to achieve the goal of preparing an infrastructure suitable for
 298 one or more applications.

299 **2.1.4 ORCHESTRATION OF BIG DATA APPLICATIONS AND EXPERIMENTS**

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

314 **2.1.5 REUSABILITY**

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

320 **2.1.6 EXECUTION WORKLOADS**

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

327 **2.1.7 SECURITY AND PRIVACY FABRIC REQUIREMENTS**

328 Although the focus of this document is not security and privacy, which are documented in the *NBDIF:*
 329 *Volume 4, Security and Privacy* [5], the interfaces defined herein must be capable of integration into a
 330 secure reference architecture that supports secure execution, secure data transfer, and privacy.
 331 Consequently, the interfaces defined herein can be augmented with frameworks and solutions that provide
 332 such mechanisms. Thus, diverse requirement needs stemming from different use cases addressing security
 333 need to be distinguished. To contrast that the security requirements between applications can vary
 334 drastically, the following example is provided. Although many of the interfaces and their objects to
 335 support Big Data applications in physics are similar to those in healthcare, they differ in the integration of
 336 security interfaces and policies. While in physics the protection of data is less of an issue, it is a stringent
 337 requirement in healthcare. Thus, deriving architectural frameworks for both may use largely similar
 338 components, but addressing security issues will be very different. The security of interfaces may be
 339 addressed in other documents. In this document, they are considered an advanced use case showcasing
 340 that the validity of the specifications introduced here is preserved, even if security and privacy
 341 requirements differ vastly among application use cases.

342 **2.2 COMPONENT-SPECIFIC INTERFACE REQUIREMENTS**

343 This section summarizes the requirements for the interfaces of the NBDRA components. The five
 344 components are listed in Figure 2 and addressed in Section 2.2.1 (System Orchestrator Interface
 345 Requirements) through Section 2.2.6 (Big Data Application Provider to Big Data Framework Provider
 346 Interface) of this document. The five main functional components of the NBDRA represent the different
 347 technical roles within a Big Data system and are the following:

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

361 **2.2.1 SYSTEM ORCHESTRATOR INTERFACE REQUIREMENTS**

362 The System Orchestrator role includes defining and integrating the required data application activities
 363 into an operational vertical system. Typically, the System Orchestrator involves a collection of more
 364 specific roles, performed by one or more actors, which manage and orchestrate the operation of the Big
 365 Data system. These actors may be human components, software components, or some combination of the
 366 two. The function of the System Orchestrator is to configure and manage the other components of the Big
 367 Data architecture to implement one or more workloads that the architecture is designed to execute. The
 368 workloads managed by the System Orchestrator may be assigning/provisioning framework components to
 369 individual physical or virtual nodes at the lower level, or providing a graphical user interface that supports
 370 the specification of workflows linking together multiple applications and components at the higher level.
 371 The System Orchestrator may also, through the Management Fabric, monitor the workloads and system to

372 confirm that specific quality of service requirements is met for each workload, and may elastically assign
 373 and provision additional physical or virtual resources to meet workload requirements resulting from
 374 changes/surges in the data or number of users/transactions. The interface to the System Orchestrator must
 375 be capable of specifying the task of orchestration the deployment, configuration, and the execution of
 376 applications within the NBDRA. A simple vendor-neutral specification to coordinate the various parts
 377 either as simple parallel language tasks or as a workflow specification is needed to facilitate the overall
 378 coordination. Integration of existing tools and services into the System Orchestrator as extensible
 379 interfaces is desirable.

380 **2.2.2 DATA PROVIDER INTERFACE REQUIREMENTS**

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

389 **2.2.3 DATA CONSUMER INTERFACE REQUIREMENTS**

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

- 394 • Search and Retrieve,
- 395 • Download,
- 396 • Analyze Locally,
- 397 • Reporting,
- 398 • Visualization, and
- 399 • Data to Use for Their Own Processes.

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

402 **2.2.4 BIG DATA APPLICATION PROVIDER INTERFACE REQUIREMENTS**

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

412 **2.2.4.1 Collection**

413 In general, the collection activity of the Big Data Application Provider handles the interface with the Data
 414 Provider. This may be a general service, such as a file server or web server configured by the System
 415 Orchestrator to accept or perform specific collections of data, or it may be an application-specific service

416 designed to pull data or receive pushes of data from the Data Provider. Since this activity is receiving data
417 at a minimum, it must store/buffer the received data until it is persisted through the Big Data Framework
418 Provider. This persistence need not be to physical media but may simply be to an in-memory queue or
419 other service provided by the processing frameworks of the Big Data Framework Provider. The collection
420 activity is likely where the extraction portion of the Extract, Transform, Load (ETL)/Extract, Load,
421 Transform (ELT) cycle is performed. At the initial collection stage, sets of data (e.g., data records) of
422 similar structure are collected (and combined), resulting in uniform security, policy, and other
423 considerations. Initial metadata is created (e.g., subjects with keys are identified) to facilitate subsequent
424 aggregation or look-up methods.

425 **2.2.4.2 Preparation**

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

435 **2.2.4.3 Analytics**

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

448 **2.2.4.4 Visualization**

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

460 **2.2.4.5 Access**

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

471 **2.2.5 BIG DATA FRAMEWORK PROVIDER INTERFACE REQUIREMENTS**

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

- 475 • Interfaces to files,
- 476 • Interfaces to virtual data directories,
- 477 • Interfaces to data streams, and
- 478 • Interfaces to data filters.

479 **2.2.5.1 Infrastructures Interface Requirements**

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

484 **2.2.5.2 Platforms Interface Requirements**

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

487 **2.2.5.3 Processing Interface Requirements**

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

494 **2.2.5.4 Crosscutting Interface Requirements**

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

501 2.2.5.5 Messaging/Communications Frameworks

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

506 2.2.5.6 Resource Management Framework

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

**514 2.2.6 BIG DATA APPLICATION PROVIDER TO BIG DATA FRAMEWORK PROVIDER
515 INTERFACE**

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

521

522 3 SPECIFICATION PARADIGM

523 This section summarizes the elementary specification paradigm.

524 3.1 HYBRID AND MULTIPLE FRAMEWORKS

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

527 3.2 DESIGN BY RESOURCE-ORIENTED ARCHITECTURE

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

535 3.3 INTERFACE COMPLIANCY

536 Due to the easy extensibility of the resource objects specified in this document and their interfaces, it is
537 important to introduce a terminology that allows the definition of interface compliancy. Three levels of
538 interface compliance are defined as follows:

- 539 • **Full Compliance:** These are reference implementations that provide full compliance to the
540 objects defined in this document. A version number is added to assure that the snapshot in time of
541 the objects is associated with the version. A full compliant framework implements all objects.
- 542 • **Partial Compliance:** These are reference implementations that provide partial compliance to the
543 objects defined in this document. A version number is added to assure that the snapshot in time of
544 the objects is associated with the version. This reference implementation implements a partial list
545 of the objects and interfaces. A document is added that specifies the differences to a full
546 compliant implementation.
- 547 • **Extended Compliance:** In addition to full and partial compliance, additional resources can be
548 identified while documenting additional resource objects and interfaces that are not included in
549 the current specification. The extended compliance document can lead to additional
550 improvements of the current specification.

551

552 4 SPECIFICATION

553 The specifications in this section are provided through an automated document creation process using the
 554 actual OpenAPI specifications yaml files as the source. It is a demonstration that showcases the
 555 generation of a fully functioning REST service based on the specifications provided in this document.
 556 However, it is expected that scalability, distribution of services, and other advanced options need to be
 557 addressed based on application requirements. Limitations of the current implementation are provided in
 558 this section.

559 4.1 LIST OF SPECIFICATIONS

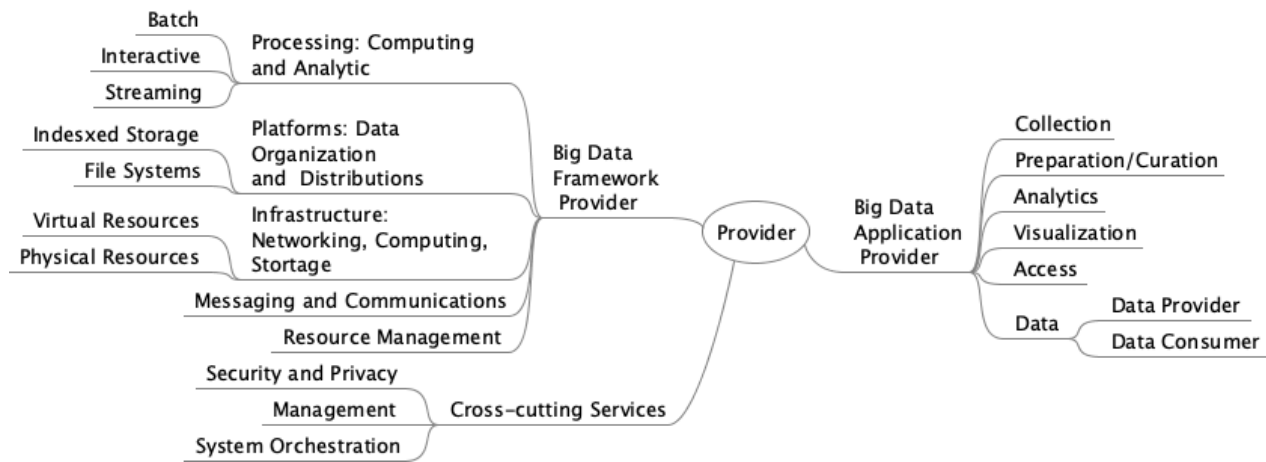
560 Table 1 shows the list of specifications included in this version of the document.

561 *Table 1: Specifications*

Service	Version	Date	Section
Alias	3.2.0	17-06-2019	Section 4.6.1
Account	3.2.0	17-06-2019	Section 4.5.3
Containers	3.2.0	17-06-2019	Section 4.10.1
Database	3.2.0	17-06-2019	Section 4.7.3
Default	3.2.0	17-06-2019	Section 4.6.3
Deployment	3.2.0	17-06-2019	Section 4.15.1
Filestore	3.2.0	17-06-2019	Section 4.7.1
Filter	3.2.0	17-06-2019	Section 4.14.2
Flavor	3.2.0	17-06-2019	Section 4.9.2
Image	3.2.0	17-06-2019	Section 4.9.1
MapReduce	3.2.0	17-06-2019	Section 4.11.1
Microservice	3.2.0	17-06-2019	Section 4.12.1
Nic	3.2.0	17-06-2019	Section 4.9.5
Network of Nodes	3.2.0	17-06-2019	Section 4.8.2
Organization	3.2.0	17-06-2019	Section 4.5.1
Public Key Store	3.2.0	17-06-2019	Section 4.5.4
Queue	3.2.0	17-06-2019	Section 4.8.4
Replica	3.2.0	17-06-2019	Section 4.7.2
Reservation	3.2.0	17-06-2019	Section 4.13.1
Scheduler	3.2.0	17-06-2019	Section 4.8.3

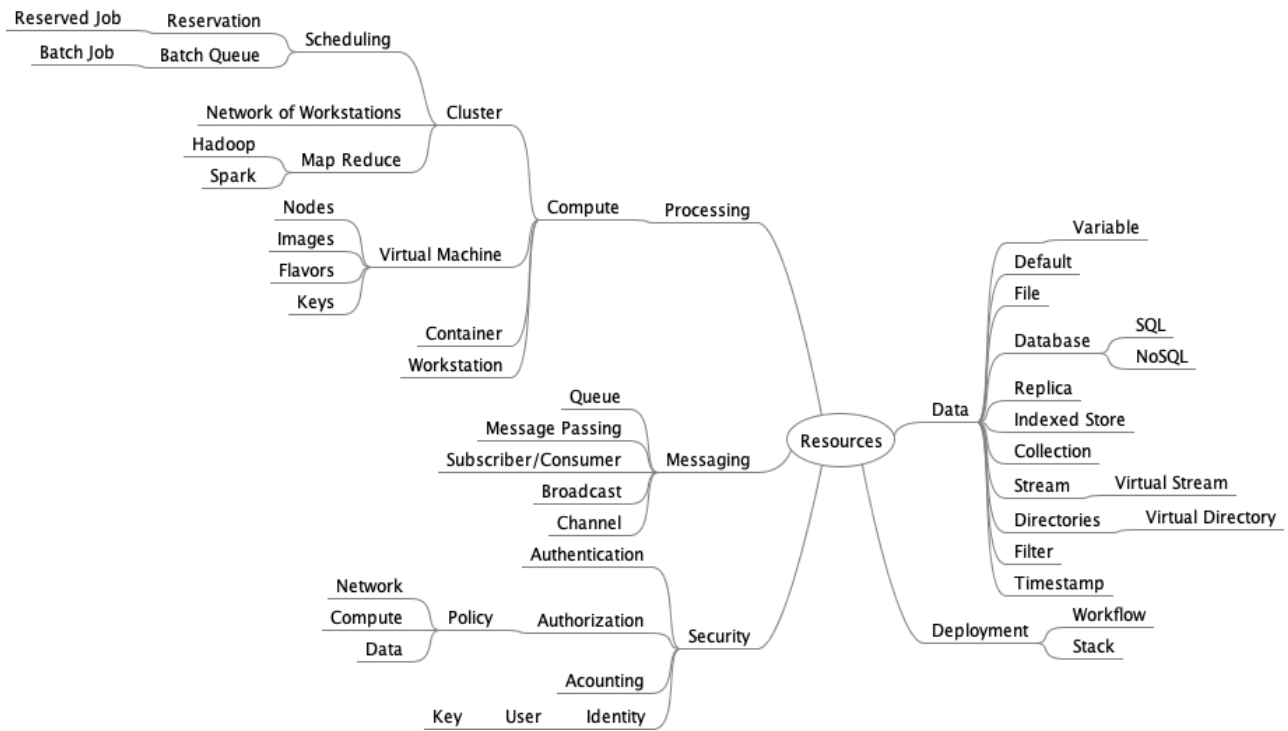
Service	Version	Date	Section
Secgroup	3.2.0	17-06-2019	Section 4.9.4
Stream	3.2.0	17-06-2019	Section 4.14.1
Timestamp	3.2.0	17-06-2019	Section 4.4.1
User	3.2.0	17-06-2019	Section 4.5.2
Variables	3.2.0	17-06-2019	Section 4.6.2
Virtual Cluster	3.2.0	17-06-2019	Section 4.8.1
Virtual Directory	3.2.0	17-06-2019	Section 4.7.4
Virtual Machine	3.2.0	17-06-2019	Section 4.9.3

562 Figure 3 shows the provider view of the specification resources.



563 *Figure 3: Provider View*

564 Figure 4 shows the resources view of the specification resources.



565 **Figure 4: Resource View**

566 **4.2 AUTHENTICATION**

567 Mechanisms are not included in this specification to manage authentication to external services. However,
 568 the working group has shown multiple solutions to this as part of Cloudmesh. These include the
 569 possibility of the following:

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

576 **4.3 STATUS CODES AND ERROR RESPONSES**

577 In case of an error or a successful response, the response header contains a HyperText Transfer Protocol
 578 (HTTP) code [19]. The response body usually contains the following:

- 579 • The HTTP response code;
- 580 • An accompanying message for the HTTP response code; and
- 581 • A field or object where the error occurred.

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

582

Table 2: HTTP Response Codes

HTTP Response	Operation	Description
200 Ok	GET, PUT, DELETE	No error, operation successful.
201 Created	POST	Successful creation of a resource.
204 No Content	GET, PUT, DELETE	Successful but no content.
400 Bad Request	GET, POST, PUT, DELETE	The request could not be understood.
401 Unauthorized	GET, POST, PUT, DELETE	User must authorize.
403 Forbidden	GET, POST, PUT, DELETE	The request has been refused due to authorization failure.
404 Not Found	GET, POST, PUT, DELETE	The requested resource could not be found.
405 Not Allowed	GET, POST, PUT, DELETE	The method is not allowed on the resource.
500 Server Error	GET, POST PUT	Internal Server error.

583 In the specification such responses are indicated and if a simple response is returned the term *Message* is
 584 used.

585 4.4 TIMESTAMP

586 Timestamps can be used in conjunction with any server side implementation of the interfaces. It can be
 587 useful to return information about when a particular resource has been created, updated, or accessed. To
 588 simplify the specification in the document, a timestamp is not explicitly listed as part of the resource, but
 589 it can be assumed that it may be added as part of the service implementation. To obtain an example
 590 timestamp a simple get function is provided.

591 4.4.1 TIMESTAMP

592 Data often needs to be time stamped to indicate when it has been accessed, created, or modified. All
 593 objects defined in this document will have, in their final version, a timestamp. The date-time string is
 594 defined in [RFC3339](#) [20].

595 4.4.1.1 Schema Timestamp

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

596 **4.4.1.2 Paths**

HTTP	Path	Summary
get	/timestamp	Returns the timestamp

597 **4.4.1.2.1 /timestamp**

598 **4.4.1.2.1.1 GET /timestamp**

599 Returns the timestamp

600 Responses

Code	Description	Schema
200	The current time and date	string

601 **4.4.1.3 timestamp.yaml**

```

602 openapi: "3.0.2"
603 info:
604   version: 3.2.0
605   x-date: 17-06-2019
606   x-status: defined
607   title: Timestamp
608   description: |-
609
610     Data often needs to be time stamped to indicate when it has been
611     accessed, created, or modified. All objects defined in this
612     document will have, in their final version, a timestamp.
613     The date-time string is defined in
614     [RFC3339](https://xml2rfc.ietf.org/public/rfc/html/rfc3339.html#anchor14).
615
616   termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
617   contact:
618     name: NIST BDRA Interface Subgroup
619     url: https://cloudmesh-community.github.io/nist/spec/
620   license:
621     name: Apache 2.0
622     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
623 servers:
624   - url: /cloudmesh/v3
625 paths:
626   /timestamp:
627     get:
628       summary: Returns the timestamp
629       description: Returns the timestamp
630       responses:
631         '200':
632           description: The current time and date
633           content:
634             application/json:
635               schema:
636                 type: string
637                 example: 1985-04-12T23:20:50.52Z
638 components:
639   schemas:
640     Timestamp:
641       type: object
642       description: the timestamp
643       properties:
644         accessed:
645           type: string
646           format: date-time
647           description: The time stamp when the object was last accessed
    
```

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

648         example: 1985-04-12T23:20:50.52Z
649     created:
650         type: string
651         format: date-time
652         description: The time stamp when the object was created
653         example: 1985-04-12T23:20:50.52Z
654     modified:
655         type: string
656         format: date-time
657         description: The time stamp when the object was modified
658         example: 1985-04-12T23:20:50.52Z
    
```

659 4.5 IDENTITY

660 As part of services an identity often needs to be specified. Such persons [21] are often part of groups.
 661 Three important terms related to the identity are distinguished as follows:

- 662 • *Organization*: The information representing an Organization that manages a Big Data Service
 663 (Section 4.5.1)
- 664 • *Group*: A group that a person may belong to that is important to define access to services
 665 (included in Section 4.5.1)
- 666 • *User*: The information identifying the profile of a person (Section 4.5.2)

667 4.5.1 ORGANIZATION

668 An important concept in many services is the management of a group of users in an organization. Within
 669 an organization, different groups of users are distinguished. Groups can be used to characterize roles that
 670 users can fulfill. Users can belong to multiple groups. Such groups can also be used to specify access
 671 rights to services.

672 4.5.1.1 Schema Organization

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

673 4.5.1.2 Schema Group

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

674 4.5.1.3 Paths

HTTP	Path	Summary
get	/organization	Returns a list of organizations
get	/organization/{name}	Returns the named organization
put	/organization/{name}	Uploads an organization to the list of organizations
delete	/organization/{name}	Deletes the named organization

HTTP	Path	Summary
get	/organization/{name}/user	Returns all users of the organization
get	/organization/{name}/user/{user}	Returns the specific user of that organization
put	/organization/{name}/user/{user}	Updates or adds a user in the organization
delete	/organization/{name}/user/{user}	Delete an user in the organization
get	/organization/{name}/group/	Returns all group names
get	/organization/{name}/group/{group}	Returns the specific group of that organization
put	/organization/{name}/group/{group}	Updates or adds a group in the organization
delete	/organization/{name}/group/{group}	Delete a group in the organization
put	/organization/{name}/group/{group}/{user}	Updates or adds a user name to the group
delete	/organization/{name}/group/{group}/{user}	Delete a user in the group

675 **4.5.1.3.1 /organization**

676 **4.5.1.3.1.1 GET /organization**

677 Returns a list of all organizations

678 Responses

Code	Description	Schema
200	The list of organizations	array[Organization]
401	Not authorized	String

679 **4.5.1.3.2 /organization/{name}**

680 **4.5.1.3.2.1 GET /organization/{name}**

681 Returns an organization by name

682 Responses

Code	Description	Schema
200	Returning the information of the organization	Organization
401	Not authorized	String
404	The named organization could not be found	String

683 Parameters

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

684 **4.5.1.3.2.2 PUT /organization/{name}**

685 Uploads an organization to the list of organizations

686 Responses

Code	Description	Schema
200	Organization created or updated	String
401	Not authorized	String
404	The organization could not be found	String

687 Request Body

Located in	Description	Required	Schema
Body	The organization to be uploaded	True	Organization

688 **4.5.1.3.2.3 DELETE /organization/{name}**

689 Deletes an organization by name

690 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named organization could not be found	String

691 Parameters

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

692 **4.5.1.3.3 /organization/{name}/user**

693 **4.5.1.3.3.1 GET /organization/{name}/user**

694 Returns all users of the organization

695 Responses

Code	Description	Schema
200	The organization	Organization
401	Not authorized	String

696 Parameters

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

697 **4.5.1.3.4 /organization/{name}/user/{user}**

698 **4.5.1.3.4.1 GET /organization/{name}/user/{user}**

699 Returns the specific user of that organization

700 Responses

Code	Description	Schema
200	The user	User
401	Not authorized	String
404	The organization or user could not be found	String

701 Parameters

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

702 **4.5.1.3.4.2 PUT /organization/{name}/user/{user}**

703 Updates or adds a user in the organization

704 Responses

Code	Description	Schema
200	User added successfully	String
401	Not authorized	String
404	The organization or user could not be found\	String

705 Parameters

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

706 Request Body

Located in	Description	Required	Schema
Body	The user to be uploaded	True	User

707 **4.5.1.3.4.3 DELETE /organization/{name}/user/{user}**

708 Delete a user in the organization

709 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The organization or user could not be found	String

710 Parameters

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

711 **4.5.1.3.5 /organization/{name}/group/**

712 **4.5.1.3.5.1 GET /organization/{name}/group/**

713 Returns all group names

714 Responses

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

715 Parameters

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

716 **4.5.1.3.6 /organization/{name}/group/{group}**

717 **4.5.1.3.6.1 GET /organization/{name}/group/{group}**

718 Returns the specific group of that organization

719 Responses

Code	Description	Schema
200	The group	Group
401	Not authorized	String
404	The organization or group could not be found	String

720 Parameters

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

721 **4.5.1.3.6.2 PUT /organization/{name}/group/{group}**

722 Updates or adds a group in the organization

723 Responses

Code	Description	Schema
200	Group added successfully	String
401	Not authorized	String
404	The organization or group could not be found	String

724 Parameters

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

725 **4.5.1.3.6.3 DELETE /organization/{name}/group/{group}**

726 Delete a group in the organization

727 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The organization or group could not be found	String

728 Parameters

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

729 **4.5.1.3.7 /organization/{name}/group/{group}/{user}**

730 **4.5.1.3.7.1 PUT /organization/{name}/group/{group}/{user}**

731 Updates or adds a user name to the group

732 Responses

Code	Description	Schema
200	User added successfully	String
401	Not authorized	String
404	The organization, group, or user could not be found	String

733 Parameters

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

734 **4.5.1.3.7.2 DELETE /organization/{name}/group/{group}/{user}**

735 Delete a user in the group

736 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The organization, group, or user could not be found	String

737 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the organization	True	String
group	path	The group name	True	String
user	path	The user name	True	String

738 **4.5.1.4 organization.yaml**

739 openapi: "3.0.2"

740 info:

741 version: 3.2.0

742 x-date: 17-06-2019

743 x-status: defined

744 title: Organization

745 description: |-

746

747 An important concept in many services is the management of a group
 748 of users in an organization. Within an organization we distinguish
 749 different groups of users. Groups can be used to characterize roles
 750 users can fulfill. Users can belong to multiple groups. Such groups can
 751 also be used to specify access rights to services.

752

753 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"

754 contact:

755 name: NIST BDRA Interface Subgroup

756 url: https://cloudmesh-community.github.io/nist/spec/

757 license:

758 name: Apache 2.0

759 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt

760 servers:

761 - url: /cloudmesh/v3

762 paths:

763 /organization:

764 get:

765 tags:

766 - Organization

767 summary: Returns a list of organizations

768 description: Returns a list of all organizations

769 operationId: cloudmesh.organization.list

770 responses:

771 '200':

772 description: The list of organizations

773 content:

774 application/json:

775 schema:

776 type: array

777 items:

778 \$ref: '#/components/schemas/Organization'

779 '401':

780 description: Not authorized

```

781 /organization/{name}:
782   get:
783     tags:
784       - Organization
785     summary: Returns the named organization
786     description: Returns an organization by name
787     operationId: cloudmesh.organization.find_by_name
788     parameters:
789       - name: name
790         in: path
791         required: true
792         schema:
793           type: string
794         description: The name of the organization
795     responses:
796       '200':
797         description: Returning the information of the organization
798         content:
799           application/json:
800             schema:
801               $ref: '#/components/schemas/Organization'
802       '401':
803         description: Not authorized
804       '404':
805         description: The named organization could not be found
806   put:
807     tags:
808       - Organization
809     summary: Uploads an organization to the list of organizations
810     description: Uploads an organization to the list of organizations
811     operationId: cloudmesh.organization.add
812     requestBody:
813       description: The organization to be uploaded
814       required: true
815       content:
816         application/json:
817           schema:
818             $ref: '#/components/schemas/Organization'
819     responses:
820       '200':
821         description: Organization created or updated
822       '401':
823         description: Not authorized
824       '404':
825         description: The organization could not be found
826   delete:
827     tags:
828       - Organization
829     summary: Deletes the named organization
830     description: Deletes an organization by name
831     operationId: cloudmesh.organization.delete_by_name
832     parameters:
833       - name: name
834         in: path
835         required: true
836         schema:
837           type: string
838         description: The name of the organization
839     responses:
840       '200':
841         description: Deletion successful
842       '401':
843         description: Not authorized
844       '404':
845         description: The named organization could not be found
846 /organization/{name}/user:
847   get:
848     tags:
849       - Organization

```

```

850     summary: Returns all users of the organization
851     description: Returns all users of the organization
852     operationId: cloudmesh.organization.user.list
853     parameters:
854       - name: name
855         in: path
856         required: true
857         schema:
858           type: string
859         description: The name of the organization
860     responses:
861       '200':
862         description: The organization
863         content:
864           application/json:
865             schema:
866               $ref: "#/components/schemas/Organization"
867       '401':
868         description: Not authorized
869 /organization/{name}/user/{user}:
870 get:
871   tags:
872     - Organization
873   summary: Returns the specific user of that organization
874   description: Returns the specific user of that organization
875   operationId: cloudmesh.organization.user.get_by_name
876   parameters:
877     - name: name
878       in: path
879       required: true
880       schema:
881         type: string
882       description: The name of the organization
883     - name: user
884       description: The user name
885       in: path
886       required: true
887       schema:
888         type: string
889   responses:
890     '200':
891       description: The user
892       content:
893         application/json:
894           schema:
895             $ref: "user.yaml#/components/schemas/User"
896     '401':
897       description: Not authorized
898     '404':
899       description: The organization or user could not be found
900 put:
901   tags:
902     - Organization
903   summary: Updates or adds a user in the organization
904   description: Updates or adds a user in the organization
905   operationId: cloudmesh.organization.user.add
906   parameters:
907     - name: name
908       in: path
909       required: true
910       schema:
911         type: string
912       description: The name of the organization
913     - name: user
914       description: The user name
915       in: path
916       required: true
917       schema:
918         type: string

```

```

919     requestBody:
920       description: The user to be uploaded
921       required: true
922       content:
923         application/json:
924           schema:
925             $ref: 'user.yaml#/components/schemas/User'
926     responses:
927       '200':
928         description: User added successfully
929       '401':
930         description: Not authorized
931       '404':
932         description: The organization or user could not be found\
933 delete:
934   tags:
935     - Organization
936   summary: Delete an user in the organization
937   description: Delete an user in the organization
938   operationId: cloudmesh.organization.user.delete
939   parameters:
940     - name: name
941       in: path
942       required: true
943       schema:
944         type: string
945         description: The name of the organization
946     - name: user
947       description: The user name
948       in: path
949       required: true
950       schema:
951         type: string
952   responses:
953     '200':
954       description: Deletion successful
955     '401':
956       description: Not authorized
957     '404':
958       description: The organization or user could not be found
959 /organization/{name}/group/:
960 get:
961   tags:
962     - Organization
963   summary: Returns all group names
964   description: Returns all group names
965   operationId: cloudmesh.organization.group.list
966   parameters:
967     - name: name
968       in: path
969       required: true
970       schema:
971         type: string
972         description: The name of the organization
973   responses:
974     '200':
975       description: Returning the information of the group
976       content:
977         application/json:
978           schema:
979             type: array
980             items:
981               type: string
982     '400':
983       description: No group found
984     '401':
985       description: Not authorized
986     '404':
987       description: The organization or group could not be found

```

```

988 /organization/{name}/group/{group}:
989   get:
990     tags:
991       - Organization
992     summary: Returns the specific group of that organization
993     description: Returns the specific group of that organization
994     operationId: cloudmesh.organization.group.get_by_name
995     parameters:
996       - name: name
997         in: path
998         required: true
999         schema:
1000           type: string
1001         description: The name of the organization
1002       - name: group
1003         description: The group name
1004         in: path
1005         required: true
1006         schema:
1007           type: string
1008     responses:
1009       '200':
1010         description: The group
1011         content:
1012           application/json:
1013             schema:
1014               $ref: "#/components/schemas/Group"
1015       '401':
1016         description: Not authorized
1017       '404':
1018         description: The organization or group could not be found
1019   put:
1020     tags:
1021       - Organization
1022     summary: Updates or adds a group in the organization
1023     description: Updates or adds a group in the organization
1024     operationId: cloudmesh.organization.group.add
1025     parameters:
1026       - name: name
1027         in: path
1028         required: true
1029         schema:
1030           type: string
1031         description: The name of the group
1032       - name: group
1033         description: The group name
1034         in: path
1035         required: true
1036         schema:
1037           type: string
1038     responses:
1039       '200':
1040         description: Group added successfully
1041       '401':
1042         description: Not authorized
1043       '404':
1044         description: The organization or group could not be found
1045   delete:
1046     tags:
1047       - Organization
1048     summary: Delete a group in the organization
1049     description: Delete a group in the organization
1050     operationId: cloudmesh.organization.greop.delete
1051     parameters:
1052       - name: name
1053         in: path
1054         required: true
1055         schema:
1056           type: string

```

```

1057     description: The name of the organization
1058     - name: group
1059       description: The group name
1060       in: path
1061       required: true
1062       schema:
1063         type: string
1064     responses:
1065       '200':
1066         description: Deletion successful
1067       '401':
1068         description: Not authorized
1069       '404':
1070         description: The organization or group could not be found
1071 /organization/{name}/group/{group}/{user}:
1072 put:
1073   tags:
1074     - Organization
1075   summary: Updates or adds a user name to the group
1076   description: Updates or adds a user name to the group
1077   operationId: cloudmesh.organization.group.user.add
1078   parameters:
1079     - name: name
1080       in: path
1081       required: true
1082       schema:
1083         type: string
1084       description: The name of the group
1085     - name: group
1086       description: The group name
1087       in: path
1088       required: true
1089       schema:
1090         type: string
1091     - name: user
1092       description: The user name
1093       in: path
1094       required: true
1095       schema:
1096         type: string
1097   responses:
1098     '200':
1099       description: User added successfully
1100     '401':
1101       description: Not authorized
1102     '404':
1103       description: The organization, group, or user could not be found
1104 delete:
1105   tags:
1106     - Organization
1107   summary: Delete a user in the group
1108   description: Delete a user in the group
1109   operationId: cloudmesh.organization.greop.delete.user
1110   parameters:
1111     - name: name
1112       in: path
1113       required: true
1114       schema:
1115         type: string
1116       description: The name of the organization
1117     - name: group
1118       description: The group name
1119       in: path
1120       required: true
1121       schema:
1122         type: string
1123     - name: user
1124       description: The user name
1125       in: path

```



```

1126         required: true
1127         schema:
1128           type: string
1129       responses:
1130         '200':
1131           description: Deletion successful
1132         '401':
1133           description: Not authorized
1134         '404':
1135           description: The organization, group, or user could not be found
1136     components:
1137       schemas:
1138         Organization:
1139           type: object
1140           properties:
1141             name:
1142               description: Name of the organization
1143               type: string
1144             users:
1145               description: List of users
1146               type: array
1147               items:
1148                 $ref: "user.yaml#/components/schemas/User"
1149         Group:
1150           type: object
1151           description: The groups
1152           properties:
1153             name:
1154               type: string
1155             description:
1156               description: The name of the group
1157               type: string
1158             description:
1159               description: The description of the group
1160             users:
1161               description: The user names that are members of the group
1162               type: array
1163               items:
1164                 type: string

```

1165 **4.5.2 USER**

1166 Services need to specify which users have access to them. User information can be reused in other
 1167 services and organized in a virtual organization. A user can be added to a named list of users within this
 1168 organization. A group associated with the user can be used to augment users to be part of one or more
 1169 groups.

1170 **4.5.2.1 Schema User**

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

1171 **4.5.2.2 Paths**

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

1172 **4.5.2.2.1 /user**

1173 **4.5.2.2.1.1 GET /user**

1174 Returns a list of all users

1175 Responses

Code	Description	Schema
200	The list of users	array[User]
401	Not authorized	String

1176 **4.5.2.2.2 /user/{name}**

1177 **4.5.2.2.2.1 GET /user/{name}**

1178 Returns a user by name

1179 Responses

Code	Description	Schema
200	Returning the information of the user	User
401	Not authorized	String
404	The named user could not be found	String

1180 Parameters

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

1181 **4.5.2.2.2.2 PUT /user/{name}**

1182 Uploads a user to the list of users

1183 Responses

Code	Description	Schema
200	User updated	String
401	Not authorized	String
404	The named user could not be found	String

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

1184	Request Body			
	Located in	Description	Required	Schema
	Body	The user to be uploaded	True	User

1185 **4.5.2.2.2.3 DELETE /user/{name}**

1186 Deletes a user by name

1187 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named user could not be found	String

1188 Parameters

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

1189 **4.5.2.3 user.yaml**

```

1190 openapi: "3.0.2"
1191 info:
1192   version: "3.2.0"
1193   x-date: 17-06-2019
1194   x-status: defined
1195   title: User
1196   description: |-
1197
1198     Services need to specify which users have access to them. User
1199     information can be reused in other services and organized in a virtual
1200     organization. A user can be added to a named list of users within this
1201     organization. A group associated with the user can be used to augment
1202     users to be part of one or more groups.
1203
1204   termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
1205   contact:
1206     name: Cloudmesh User
1207     url: https://cloudmesh-community.github.io/nist/spec/
1208   license:
1209     name: Apache 2.0
1210     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
1211   servers:
1212     - url: /cloudmesh/v3
1213   paths:
1214     /user:
1215       get:
1216         tags:
1217           - User
1218         summary: Returns a list of users
1219         description: Returns a list of all users
1220         operationId: cloudmesh.user.list
1221         responses:
1222           '200':
1223             description: The list of users
1224             content:
1225               application/json:
1226                 schema:
1227                   type: array
1228                   items:

```

```

1229         $ref: '#/components/schemas/User'
1230     '401':
1231         description: Not authorized
1232 /user/{name}:
1233     get:
1234         tags:
1235             - User
1236         summary: Returns the named user
1237         description: Returns an user by name
1238         operationId: cloudmesh.user.find_by_name
1239         parameters:
1240             - name: name
1241               in: path
1242               required: true
1243               schema:
1244                 type: string
1245             description: The name of the user
1246         responses:
1247             '200':
1248                 description: Returning the information of the user
1249                 content:
1250                     application/json:
1251                         schema:
1252                             $ref: '#/components/schemas/User'
1253             '401':
1254                 description: Not authorized
1255             '404':
1256                 description: The named user could not be found
1257     put:
1258         tags:
1259             - User
1260         summary: Uploads a user to the list of users
1261         description: Uploads a user to the list of users
1262         operationId: cloudmesh.user.add
1263         requestBody:
1264             description: The user to be uploaded
1265             required: true
1266             content:
1267                 application/json:
1268                     schema:
1269                         $ref: '#/components/schemas/User'
1270         responses:
1271             '200':
1272                 description: User updated
1273             '401':
1274                 description: Not authorized
1275             '404':
1276                 description: The named user could not be found
1277     delete:
1278         tags:
1279             - User
1280         summary: Deletes the named user
1281         description: Deletes an user by name
1282         operationId: cloudmesh.user.delete_by_name
1283         parameters:
1284             - name: name
1285               in: path
1286               required: true
1287               schema:
1288                 type: string
1289             description: The name of the user
1290         responses:
1291             '200':
1292                 description: Deletion successful
1293             '401':
1294                 description: Not authorized
1295             '404':
1296                 description: The named user could not be found
1297 components:

```

```

1298 schemas:
1299   User:
1300     type: object
1301     properties:
1302       username:
1303         type: string
1304         description: The unique username associated with the user
1305       firstname:
1306         type: string
1307         description: The firstname of the user
1308       lastname:
1309         type: string
1310         description: The lastname of the user
1311       email:
1312         type: string
1313         description: The email of the user
1314       comment:
1315         type: string
1316         description: A comment regarding the user
1317       publickey:
1318         type: string
1319         description: The public key of the user
    
```

1320 **4.5.3 ACCOUNT**

1321 Accounting can be used to charge the use of resources. Accounting can be implemented on a variety of
 1322 resources, such as users, groups, or organizations. It is up to the implementer to provide rules and cost for
 1323 the accounting. If needed, multiple accounting resources can be implemented.

1324 **4.5.3.1 Schema Account**

Property	Type	Description
name	string	name of account
description	string	the purpose of the account
charge	integer	The current charge of the account
unit	string	the unit in which the account is charged and the charge value is stored

1325 **4.5.3.2 Paths**

HTTP	Path	Summary
get	/account	Returns the accounts
get	/account/{name}	Returns the named account
put	/account/{name}	Set the value of an account
delete	/account/{name}	Deletes the named account

1326 **4.5.3.2.1 /account**

1327 **4.5.3.2.1.1 GET /account**

1328 Returns the accounts

1329 Responses

Code	Description	Schema
200	The list of accounts	array[Account]
401	Not authorized	String

1330 **4.5.3.2.2 /account/{name}**

1331 **4.5.3.2.2.1 GET /account/{name}**

1332 Returns the named account

1333 Responses

Code	Description	Schema
200	Returning the information of the account	Account
401	Not authorized	String
404	The named account could not be found	String

1334 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the account	True	String

1335 **4.5.3.2.2.2 PUT /account/{name}**

1336 Set the value of the named account

1337 Responses

Code	Description	Schema
200	Account updated or created	String
400	Error updating account	String
401	Not authorized	String

1338 Request Body

Located in	Description	Required	Schema
Body	The account and its value	True	Account

1339 **4.5.3.2.2.3 DELETE /account/{name}**

1340 Deletes an account by name

1341 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named account could not be found	String

1342 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the account	True	String

1343 **4.5.3.3 account.yaml**

1344 openapi: "3.0.2"
 1345 info:
 1346 version: 3.2.0
 1347 x-date: 17-06-2019
 1348 x-status: defined
 1349 title: Account
 1350 description: |-
 1351
 1352 To charge the use of resources accounting can be used. Accounting can be
 1353 implemented on a variety of resources, such as users, groups or
 1354 organizations. It is up to the implementer to provide rules and cost for
 1355 it. If needed, multiple accounting resources can be implemented.
 1356
 1357 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
 1358 contact:
 1359 name: NIST BDRA Interface Subgroup
 1360 license:
 1361 name: Apache 2.0
 1362 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
 1363 externalDocs:
 1364 description: more information about the NIST BDRA
 1365 url: https://github.com/cloudmesh-community/nist/blob/master/docs/nistvol8-2.epub?raw=true
 1366 servers:
 1367 - url: /cloudmesh/v3
 1368 paths:
 1369 /account:
 1370 get:
 1371 tags:
 1372 - Account
 1373 summary: Returns the accounts
 1374 description: Returns the accounts
 1375 operationId: cloudmesh.account.list
 1376 responses:
 1377 '200':
 1378 description: The list of accounts
 1379 content:
 1380 application/json:
 1381 schema:
 1382 type: array
 1383 items:
 1384 \$ref: '#/components/schemas/Account'
 1385 '401':
 1386 description: Not authorized
 1387 /account/{name}:
 1388 get:
 1389 tags:
 1390 - Account
 1391 summary: Returns the named account
 1392 description: Returns the named account
 1393 operationId: cloudmesh.account.find_by_name
 1394 parameters:
 1395 - name: name
 1396 in: path
 1397 required: true
 1398 schema:
 1399 type: string
 1400 description: The name of the account
 1401 responses:
 1402 '200':
 1403 description: Returning the information of the account

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

1404     content:
1405       application/json:
1406         schema:
1407           $ref: '#/components/schemas/Account'
1408     '401':
1409       description: Not authorized
1410     '404':
1411       description: The named account could not be found
1412   put:
1413     tags:
1414       - Account
1415     summary: Set the value of an account
1416     description: Set the value of the named account
1417     operationId: cloudmesh.account.add
1418     requestBody:
1419       description: The account and its value
1420       required: true
1421       content:
1422         application/json:
1423           schema:
1424             $ref: '#/components/schemas/Account'
1425     responses:
1426       '200':
1427         description: Account updated or created
1428       '400':
1429         description: Error updating account
1430       '401':
1431         description: Not authorized
1432   delete:
1433     tags:
1434       - Account
1435     summary: Deletes the named account
1436     description: Deletes an account by name
1437     operationId: cloudmesh.account.delete_by_name
1438     parameters:
1439       - name: name
1440         in: path
1441         required: true
1442         schema:
1443           type: string
1444         description: The name of the account
1445     responses:
1446       '200':
1447         description: Deletion successful
1448       '401':
1449         description: Not authorized
1450       '404':
1451         description: The named account could not be found
1452   components:
1453     schemas:
1454       Account:
1455         type: object
1456         description: account information including finances and summary.
1457         properties:
1458           name:
1459             type: string
1460             description: name of account
1461             example: GroupAccount
1462           description:
1463             type: string
1464             description: the purpose of the account
1465             example: "Charging all users for the account access"
1466           charge:
1467             type: integer
1468             minimum: 0
1469             description: The current charge of the account
1470             example: 1
1471           unit:
1472             type: string

```


1473 description: the unit in which the account is charged and the charge value is stored
 1474 example: SU

1475 **4.5.4 PUBLIC KEY STORE**

1476 Many services and frameworks use Secure Shell (SSH) keys to authenticate. This service allows the
 1477 convenient storage of the public keys.

1478 **4.5.4.1 Schema Key**

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 Rivest–Shamir–Adleman (RSA) and Digital Signature Algorithm (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 Uniform Resource Identifier (URI) of the public key if any
fingerprint	string	The fingerprint of the public key

1479 **4.5.4.2 Paths**

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

1480 **4.5.4.2.1 /key**

1481 **4.5.4.2.1.1 GET /key**

1482 Returns a list of all keys

1483 Responses

Code	Description	Schema
200	The list of keys	array[Key]
401	Not authorized	String

1484 **4.5.4.2.2 /key/{name}**

1485 **4.5.4.2.2.1 GET /key/{name}**

1486 Returns a key by name

1487 Responses

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

Code	Description	Schema
200	Returning the information of the key	Key
401	Not authorized	String
404	The named key could not be found	String

1488 Parameters

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

1489 **4.5.4.2.2.2 PUT /key/{name}**

1490 Sets the named key

1491 Responses

Code	Description	Schema
200	Key updated	String
401	Not authorized	String
404	The named key could not be found	String

1492 Request Body

Located in	Description	Required	Schema
Body	The new key to create	True	Key

1493 **4.5.4.2.2.3 DELETE /key/{name}**

1494 Deletes a key by name

1495 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named key could not be found	String

1496 Parameters

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

1497 **4.5.4.3 publickeystore.yaml**

1498 openapi: "3.0.2"

1499 info:

1500 version: 3.2.0

1501 x-date: 17-06-2019

1502 x-status: defined

1503 title: Public Key Store

1504 description: |-

1505
 1506 Many services and frameworks use Secure Shell (SSH) keys to
 1507 authenticate. This service allows the convenient storage of the

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

1508     public keys.
1509
1510 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
1511 contact:
1512   name: NIST BDRA Interface Subgroup
1513   url: https://cloudmesh-community.github.io/nist
1514 license:
1515   name: Apache 2.0
1516   url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
1517 servers:
1518   - url: /cloudmesh/v3
1519 paths:
1520   /key:
1521     get:
1522       tags:
1523         - Key
1524       summary: Returns a list of keys
1525       description: Returns a list of all keys
1526       operationId: cloudmesh.key.list
1527       responses:
1528         '200':
1529           description: The list of keys
1530           content:
1531             application/json:
1532               schema:
1533                 type: array
1534                 items:
1535                   $ref: '#/components/schemas/Key'
1536         '401':
1537           description: Not authorized
1538   /key/{name}:
1539     get:
1540       tags:
1541         - Key
1542       summary: Returns the named key
1543       description: Returns a key by name
1544       operationId: cloudmesh.key.find_by_name
1545       parameters:
1546         - name: name
1547           in: path
1548           required: true
1549           schema:
1550             type: string
1551           description: The name of the key
1552       responses:
1553         '200':
1554           description: Returning the information of the key
1555           content:
1556             application/json:
1557               schema:
1558                 $ref: '#/components/schemas/Key'
1559         '401':
1560           description: Not authorized
1561         '404':
1562           description: The named key could not be found
1563     put:
1564       tags:
1565         - Key
1566       summary: Set a key
1567       description: Sets the named key
1568       operationId: cloudmesh.key.add
1569       requestBody:
1570         description: The new key to create
1571         required: true
1572         content:
1573           application/json:
1574             schema:
1575               $ref: '#/components/schemas/Key'
1576       responses:

```

```

1577     '200':
1578         description: Key updated
1579     '401':
1580         description: Not authorized
1581     '404':
1582         description: The named key could not be found
1583 delete:
1584     tags:
1585         - Key
1586     summary: Deletes the named key
1587     description: Deletes a key by name
1588     operationId: cloudmesh.key.delete_by_name
1589     parameters:
1590         - name: name
1591           in: path
1592           required: true
1593           schema:
1594             type: string
1595           description: The name of the key
1596     responses:
1597         '200':
1598             description: Deletion successful
1599         '401':
1600             description: Not authorized
1601         '404':
1602             description: The named key could not be found
1603 components:
1604     schemas:
1605         Key:
1606             type: object
1607             description: the public key
1608             properties:
1609                 name:
1610                     type: string
1611                     description: The name of the public key
1612                 value:
1613                     type: string
1614                     description: The value of the public key
1615                 kind:
1616                     type: string
1617                     description: The key kind such as rsa, dsa
1618                 group:
1619                     type: string
1620                     description: An optional group name allowing to group keys to create
1621                               custom key groups within the public key store
1622             comment:
1623                 type: string
1624                 description: A comment for the public key
1625             uri:
1626                 type: string
1627                 description: The uri of the public key if any
1628             fingerprint:
1629                 type: string
1630                 description: The fingerprint of the public key

```

1631 4.6 VARIABLE, DEFAULT, AND ALIAS

1632 4.6.1 ALIAS

1633 Often a user has the desire to create a custom name for an object. An alias allows for this possibility by
 1634 associating a user defined name or *alias* to a previously used name. The aliases could be shared with other
 1635 users. A name could have one or more aliases.

1636 **4.6.1.1 Schema Alias**

Property	Type	Description
Name	string	The name of the alias
Source	string	The original unique object name

1637 **4.6.1.2 Paths**

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

1638 **4.6.1.2.1 /alias**

1639 **4.6.1.2.1.1 GET /alias**

1640 Returns a list of all aliases

1641 Responses

Code	Description	Schema
200	The list of aliases	array[Alias]
400	No alias found	String
401	Not authorized	String

1642 **4.6.1.2.2 /alias/{name}**

1643 **4.6.1.2.2.1 GET /alias/{name}**

1644 Returns an alias by name

1645 Responses

Code	Description	Schema
200	Returning the information of the alias	Alias
401	Not authorized	String
404	The named alias could not be found	String

1646 Parameters

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

1647 **4.6.1.2.2.2 PUT /alias/{name}**

1648 Sets the named alias

1649 Responses

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

Code	Description	Schema
200	Alias updated or created	String
401	Not authorized	String

1650 Request Body

Located in	Description	Required	Schema
Body	The new alias to create	True	Alias

1651 **4.6.1.2.2.3 DELETE /alias/{name}**

1652 Deletes an alias by name

1653 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named alias could not be found	String

1654 Parameters

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

1655 **4.6.1.3 alias.yaml**

```

1656 openapi: '3.0.2'
1657 info:
1658   version: 3.2.0
1659   x-date: 17-06-2019
1660   x-status: defined
1661   title: Alias
1662   description: |-
1663
1664     Often a user has the desire to create a custom name for an object. An
1665     alias allows to do that while associating a user defined name or
1666     *alias* to a previously used name. The aliases could be shared with other
1667     users. A name could have one or more aliases.
1668
1669     termsOfService: 'https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt'
1670     contact:
1671       name: NIST BDRA Interface Subgroup
1672       url: https://cloudmesh-community.github.io/nist/spec/
1673     license:
1674       name: Apache 2.0
1675       url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
1676     servers:
1677       - url: /cloudmesh/v3
1678     paths:
1679       /alias:
1680         get:
1681           tags:
1682             - Alias
1683           summary: Returns a list of aliases
1684           description: Returns a list of all aliases
1685           operationId: cloudmesh.alias.list
1686           responses:
1687             '200':
1688               description: The list of aliases
    
```

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

1689         content:
1690             application/json:
1691                 schema:
1692                     type: array
1693                     items:
1694                         $ref: '#/components/schemas/Alias'
1695             '400':
1696                 description: No alias found
1697             '401':
1698                 description: Not authorized
1699 /aliases/{name}:
1700 get:
1701     tags:
1702     - Alias
1703     summary: Returns the named alias
1704     description: Returns an alias by name
1705     operationId: cloudmesh.alias.find_by_name
1706     parameters:
1707     - name: name
1708       in: path
1709       required: true
1710       schema:
1711         type: string
1712       description: The name of the alias
1713     responses:
1714     '200':
1715         description: Returning the information of the alias
1716         content:
1717             application/json:
1718                 schema:
1719                     $ref: '#/components/schemas/Alias'
1720     '401':
1721         description: Not authorized
1722     '404':
1723         description: The named alias could not be found
1724 put:
1725     tags:
1726     - Alias
1727     summary: Set an alias
1728     description: Sets the named alias
1729     operationId: cloudmesh.alias.add
1730     requestBody:
1731         description: The new alias to create
1732         required: true
1733         content:
1734             application/json:
1735                 schema:
1736                     $ref: '#/components/schemas/Alias'
1737     responses:
1738     '200':
1739         description: Alias updated or created
1740     '401':
1741         description: Not authorized
1742 delete:
1743     tags:
1744     - Alias
1745     summary: Deletes the named alias
1746     description: Deletes an alias by name
1747     operationId: cloudmesh.alias.delete_by_name
1748     parameters:
1749     - name: name
1750       in: path
1751       required: true
1752       schema:
1753         type: string
1754       description: The name of the alias
1755     responses:
1756     '200':
1757         description: Deletion successful

```

```

1758         '401':
1759             description: Not authorized
1760         '404':
1761             description: The named alias could not be found
1762     components:
1763         schemas:
1764             Alias:
1765                 type: object
1766                 description: the alias
1767                 properties:
1768                     name:
1769                         type: string
1770                         description: The name of the alias
1771                 source:
1772                     type: string
1773                     description: The original unique object name
    
```

1774 **4.6.2 VARIABLES**

1775 Variables are a simple string key value storage to store simple values. Each variable can have a datatype
 1776 so that it can be used for serialization into other formats. Internally they are stored as strings.

1777 **4.6.2.1 Schema Variable**

Property	Type	Description
Name	string	Name of the variable
Value	string	Value of the variable
Description	string	A description of the variable
Datatype	string	The data type of the variable which can be used for serialization

1778 **4.6.2.2 Paths**

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

1779 **4.6.2.2.1 /variable**

1780 **4.6.2.2.1.1 GET /variable**

1781 Returns the variables

1782 Responses

Code	Description	Schema
200	The list of variables	array[Variable]
400	No variable found	String

1783 **4.6.2.2.2 /variable/{name}**

1784 **4.6.2.2.2.1 GET /variable/{name}**

1785 Returns the named variable

1786 Responses

Code	Description	Schema
200	Returning the information of the variable	Variable
401	Not authorized	String
404	The named variable could not be found	String

1787 Parameters

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

1788 **4.6.2.2.2.2 PUT /variable/{name}**

1789 Sets the value of the named variable

1790 Responses

Code	Description	Schema
200	Variable updated or created	String
400	Error updating variable	String
401	Not authorized	String

1791 Request Body

Located in	Description	Required	Schema
Body	The variable and its value	True	Variable

1792 **4.6.2.2.2.3 DELETE /variable/{name}**

1793 Deletes a variable by name

1794 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named variable could not be found	String

1795 Parameters

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

1796 **4.6.2.3 variables.yaml**

```

1797 openapi: "3.0.2"
1798 info:
1799   version: 3.2.0
1800   x-date: 17-06-2019
1801   x-status: defined
1802   title: Variables
1803   description: |-
1804
1805     Variables are a simple string key value storage to store simple
1806     values. Each variable can have a datatype, so that it can be used for
1807     serialization into other formats. Internally they are stored as strings.
1808
1809   termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
1810   contact:
1811     name: NIST BDRA Interface Subgroup
1812     url: https://cloudmesh-community.github.io/nist/spec/
1813   license:
1814     name: Apache 2.0
1815     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
1816 servers:
1817   - url: /cloudmesh/v3
1818 paths:
1819   /variable:
1820     get:
1821       tags:
1822         - Variable
1823       summary: Returns the variables
1824       description: Returns the variables
1825       operationId: cloudmesh.variable.list
1826       responses:
1827         '200':
1828           description: The list of variables
1829           content:
1830             application/json:
1831               schema:
1832                 type: array
1833                 items:
1834                   $ref: '#/components/schemas/Variable'
1835         '400':
1836           description: No variable found
1837   /variable/{name}:
1838     get:
1839       tags:
1840         - Variable
1841       summary: Returns the named variable
1842       description: Returns the named variable
1843       operationId: cloudmesh.variable.find_by_name
1844       parameters:
1845         - name: name
1846           in: path
1847           required: true
1848           schema:
1849             type: string
1850           description: The name of the variable
1851       responses:
1852         '200':
1853           description: Returning the information of the variable
1854           content:
1855             application/json:
1856               schema:
1857                 $ref: '#/components/schemas/Variable'
1858         '401':
1859           description: Not authorized
1860         '404':
1861           description: The named variable could not be found
1862     put:
1863       tags:

```

```

1864     - Variable
1865     summary: Set the value of a variable
1866     description: Set the value of the named variable
1867     operationId: cloudmesh.variable.add
1868     requestBody:
1869       description: The variable and its value
1870       required: true
1871       content:
1872         application/json:
1873           schema:
1874             $ref: '#/components/schemas/Variable'
1875     responses:
1876       '200':
1877         description: Variable updated or created
1878       '400':
1879         description: Error updating variable
1880       '401':
1881         description: Not authorized
1882     delete:
1883       tags:
1884         - Variable
1885       summary: Deletes the named variable
1886       description: Deletes a variable by name
1887       operationId: cloudmesh.variable.delete_by_name
1888       parameters:
1889         - name: name
1890           in: path
1891           required: true
1892           schema:
1893             type: string
1894           description: The name of the variable
1895     responses:
1896       '200':
1897         description: Deletion successful
1898       '401':
1899         description: Not authorized
1900       '404':
1901         description: The named variable could not be found
1902     components:
1903       schemas:
1904         Variable:
1905           type: object
1906           description: the variables
1907           properties:
1908             name:
1909               type: string
1910               description: Name of the variable
1911             value:
1912               type: string
1913               description: Value of the variable
1914           description:
1915             type: string
1916             description: A description of the variable
1917           datatype:
1918             type: string
1919             description: The data type of the variable which can be used for
1920               serialization

```

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

4.6.3 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.

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

1925 **4.6.3.1 Schema Default**

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

1926 **4.6.3.2 Paths**

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

1927 **4.6.3.2.1 /default**

1928 **4.6.3.2.1.1 GET /default**

1929 Returns a list of all defaults

1930 Responses

Code	Description	Schema
200	The list of defaults	array[Default]
401	Not authorized	String

1931 **4.6.3.2.2 /default/{name}**

1932 **4.6.3.2.2.1 GET /default/{name}**

1933 Returns a default by name

1934 Responses

Code	Description	Schema
200	Returning the information of the default	Default
401	Not authorized	String
404	The named default could not be found	String

1935 Parameters

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

1936 **4.6.3.2.2.2 PUT /default/{name}**

1937 Sets the named default

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

1938 Responses

Code	Description	Schema
200	Default updated or created	String
401	Not authorized	String

1939 Request Body

Located in	Description	Required	Schema
Body	The new default to create	True	Default

1940 **4.6.3.2.2.3 DELETE /default/{name}**

1941 Deletes a default by name

1942 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named default could not be found	String

1943 Parameters

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

1944 **4.6.3.3 default.yaml**

1945 openapi: "3.0.2"

1946 info:

1947 version: 3.2.0

1948 x-date: 17-06-2019

1949 x-status: defined

1950 title: Default

1951 description: |-

1952

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.

1953

termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"

1954

contact:

1955

name: NIST BDRA Interface Subgroup

1956

url: https://cloudmesh-community.github.io/nist/spec/

1957

license:

1958

name: Apache 2.0

1959

url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt

1960

servers:

1961

- url: /cloudmesh/v3

1962

paths:

1963

/default:

1964

get:

1965

tags:

1966

- Default

1967

summary: Returns a list of defaults

1968

description: Returns a list of all defaults

1969

operationId: cloudmesh.default.list

```

1976     responses:
1977       '200':
1978         description: The list of defaults
1979         content:
1980           application/json:
1981             schema:
1982               type: array
1983               items:
1984                 $ref: '#/components/schemas/Default'
1985       '401':
1986         description: Not authorized
1987 /default/{name}:
1988   get:
1989     tags:
1990     - Default
1991     summary: Returns the named default
1992     description: Returns a default by name
1993     operationId: cloudmesh.default.find_by_name
1994     parameters:
1995     - name: name
1996       in: path
1997       required: true
1998       schema:
1999         type: string
2000     description: The name of the default
2001   responses:
2002     '200':
2003       description: Returning the information of the default
2004       content:
2005         application/json:
2006           schema:
2007             $ref: '#/components/schemas/Default'
2008     '401':
2009       description: Not authorized
2010     '404':
2011       description: The named default could not be found
2012   put:
2013     tags:
2014     - Default
2015     summary: Set a default
2016     description: Sets the named default
2017     operationId: cloudmesh.default.add
2018     requestBody:
2019       description: The new default to create
2020       required: true
2021       content:
2022         application/json:
2023           schema:
2024             $ref: '#/components/schemas/Default'
2025     responses:
2026     '200':
2027       description: Default updated or created
2028     '401':
2029       description: Not authorized
2030   delete:
2031     tags:
2032     - Default
2033     summary: Deletes the named default
2034     description: Deletes a default by name
2035     operationId: cloudmesh.default.delete_by_name
2036     parameters:
2037     - name: name
2038       in: path
2039       required: true
2040       schema:
2041         type: string
2042     description: The name of the default
2043     responses:
2044     '200':

```

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

```

2045         description: Deletion successful
2046         '401':
2047           description: Not authorized
2048         '404':
2049           description: The named default could not be found
2050 components:
2051   schemas:
2052     Default:
2053       type: object
2054       description: the defaults
2055       properties:
2056         name:
2057           type: string
2058           description: The name of the default
2059           example: "image"
2060         value:
2061           type: string
2062           description: The value of the default
2063           example: "m1.medium"
2064         context:
2065           type: string
2066           description: The context of the default
2067           example: "cloud.vm.flavor"
    
```

2068 4.7 DATA MANAGEMENT

2069 4.7.1 FILESTORE

2070 A file store is a resource allowing storage of data as a traditional file. A file store can contain any number
 2071 of files with additional attributes describing the file. A file store is located on the physical server. It
 2072 contains access to the content of the file. This contrasts virtual directories that are just pointers to files,
 2073 which could include files located in different file stores. A virtual directory also does not contain the
 2074 content of the file, but just a pointer where to find the file.

2075 4.7.1.1 Schema File

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(binary)	the content of the file

2076 4.7.1.2 Paths

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

2077 **4.7.1.2.1 /file**

2078 **4.7.1.2.1.1 GET /file**

2079 Returns a list of all files

2080 Responses

Code	Description	Schema
200	The list of files	array[File]
401	Not authorized	String

2081 **4.7.1.2.2 /file/{name}**

2082 **4.7.1.2.2.1 GET /file/{name}**

2083 Returns a file by name in the file store

2084 Responses

Code	Description	Schema
200	Returning the information of the file store	File
401	Not authorized	String
404	The named file could not be found	String

2085 Parameters

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

2086 **4.7.1.2.2.2 PUT /file/{name}**

2087 Uploads a file to the list of files in the file store

2088 Responses

Code	Description	Schema
200	File updated or created	String
401	Not authorized	String

2089 Request Body

Located in	Description	Required	Schema
Body	The file to be uploaded	True	File

2090 **4.7.1.2.2.3 DELETE /file/{name}**

2091 Deletes a file by name

2092 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named file could not be found	String

2093 Parameters

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

2094 **4.7.1.3 filestore.yaml**

```

2095 openapi: "3.0.2"
2096 info:
2097   version: 3.2.0
2098   x-date: 17-06-2019
2099   x-status: defined
2100   title: File
2101   description: |-
2102
2103     A file store is a resource allowing storage of data as a traditional file.
2104     A file store can contain any number of files with additional attributes
2105     describing the file. A file store is located on the physical server. It
2106     contains access to the content of the file. This contrasts virtual
2107     directories that are just pointers to files, which could include files
2108     located in different file stores. A virtual directory also does not
2109     contain the content of the file, but just a pointer where to find the file.
2110
2111   termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
2112   contact:
2113     name: NIST BDRA Interface Subgroup
2114     url: https://cloudmesh-community.github.io/nist/spec/
2115   license:
2116     name: Apache 2.0
2117     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
2118 servers:
2119   - url: /cloudmesh/v3
2120 paths:
2121   /file:
2122     get:
2123       tags:
2124         - File
2125       summary: Returns a list of files in the file store
2126       description: Returns a list of all files
2127       operationId: cloudmesh.file.list
2128       responses:
2129         '200':
2130           description: The list of files
2131           content:
2132             application/json:
2133               schema:
2134                 type: array
2135                 items:
2136                   $ref: '#/components/schemas/File'
2137         '401':
2138           description: Not authorized
2139   /file/{name}:
2140     get:
2141       tags:
2142         - File
2143       summary: Returns the named file in the file store
2144       description: Returns an file by name in the file store

```

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

2145     operationId: cloudmesh.file.find_by_name
2146     parameters:
2147       - name: name
2148         in: path
2149         required: true
2150         schema:
2151           type: string
2152         description: The name of the file
2153     responses:
2154       '200':
2155         description: Returning the information of the file store
2156         content:
2157           application/json:
2158             schema:
2159               $ref: '#/components/schemas/File'
2160       '401':
2161         description: Not authorized
2162       '404':
2163         description: The named file could not be found
2164     put:
2165       tags:
2166         - File
2167       summary: Uploads a file to the list of files in the file store
2168       description: Uploads a file to the list of files in the file store
2169       operationId: cloudmesh.file.add
2170       requestBody:
2171         description: The file to be uploaded
2172         required: true
2173         content:
2174           application/json:
2175             schema:
2176               $ref: '#/components/schemas/File'
2177       responses:
2178         '200':
2179           description: File updated or created
2180         '401':
2181           description: Not authorized
2182     delete:
2183       tags:
2184         - File
2185       summary: Deletes the named file in the file store
2186       description: Deletes an file by name
2187       operationId: cloudmesh.file.delete_by_name
2188       parameters:
2189         - name: name
2190           in: path
2191           required: true
2192           schema:
2193             type: string
2194           description: The name of the file
2195       responses:
2196         '200':
2197           description: Deletion successful
2198         '401':
2199           description: Not authorized
2200         '404':
2201           description: The named file could not be found
2202     components:
2203       schemas:
2204         File:
2205           type: object
2206           description: an object representing a file
2207           properties:
2208             name:
2209               type: string
2210               description: The name of the file
2211           endpoint:
2212             type: string
2213             description: The location of the file

```

2214 checksum:
 2215 type: string
 2216 description: The checksum of the file
 2217 size:
 2218 type: integer
 2219 description: The size of the file in byte
 2220 content:
 2221 type: string
 2222 format: binary
 2223 description: the content of the file

2224 **4.7.2 REPLICA**

2225 In many distributed systems, it is important that a file can be replicated among different systems to
 2226 provide faster access. It is important to provide a mechanism to trace the pedigree of the file while
 2227 pointing to its original source. A replica will point to a file in a file store and store the contents in the file
 2228 store instead of the replica. The replica is just a pointer.

2229 **4.7.2.1 Schema Replica**

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

2230 **4.7.2.2 Paths**

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

2231 **4.7.2.2.1 /replica**

2232 **4.7.2.2.1.1 GET /replica**

2233 Returns a list of all replicas

2234 Responses

Code	Description	Schema
200	The list of replicas	array[Replica]
401	Not authorized	String

2235 **4.7.2.2.2 /replica/{name}**

2236 **4.7.2.2.2.1 GET /replica/{name}**

2237 Returns a replica by name

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

2238 Responses

Code	Description	Schema
200	Returning the information of the replica	Replica
401	Not authorized	String
404	The named replica could not be found	String

2239 Parameters

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

2240 **4.7.2.2.2 PUT /replica/{name}**

2241 Uploads a replica to the list of replicas

2242 Responses

Code	Description	Schema
200	Replica updated or created	String
401	Not authorized	String

2243 Request Body

Located in	Description	Required	Schema
Body	The replica to be uploaded	True	Replica

2244 **4.7.2.2.3 DELETE /replica/{name}**

2245 Deletes a replica by name

2246 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named replica could not be found	String

2247 Parameters

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

2248 **4.7.2.3 replica.yaml**

2249 openapi: "3.0.2"

2250 info:

2251 version: 3.2.0

2252 x-date: 17-06-2019

2253 x-status: defined

2254 title: Replica

2255 description: |-

2256
2257 In many distributed systems, it is important that a file can be
2258 replicated among different systems to provide faster access. It is

```

2259     important to provide a mechanism to trace the pedigree of the file
2260     while pointing to its original source. A replica will point to a file in
2261     a file store and store the contents in the file store instead of the
2262     replica. The replica is just a pointer.
2263     termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
2264     contact:
2265       name: NIST BDRA Interface Subgroup
2266       url: https://cloudmesh-community.github.io/nist/spec/
2267     license:
2268       name: Apache 2.0
2269       url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
2270     servers:
2271       - url: /cloudmesh/v3
2272     paths:
2273       /replica:
2274         get:
2275           tags:
2276             - Replica
2277           summary: Returns a list of replicas
2278           description: Returns a list of all replicas
2279           operationId: cloudmesh.replica.list
2280           responses:
2281             '200':
2282               description: The list of replicas
2283               content:
2284                 application/json:
2285                   schema:
2286                     type: array
2287                     items:
2288                       $ref: '#/components/schemas/Replica'
2289             '401':
2290               description: Not authorized
2291       /replica/{name}:
2292         get:
2293           tags:
2294             - Replica
2295           summary: Returns the named replica
2296           description: Returns an replica by name
2297           operationId: cloudmesh.replica.find_by_name
2298           parameters:
2299             - name: name
2300               in: path
2301               required: true
2302               schema:
2303                 type: string
2304               description: The name of the replica
2305           responses:
2306             '200':
2307               description: Returning the information of the replica
2308               content:
2309                 application/json:
2310                   schema:
2311                     $ref: '#/components/schemas/Replica'
2312             '401':
2313               description: Not authorized
2314             '404':
2315               description: The named replica could not be found
2316         put:
2317           tags:
2318             - Replica
2319           summary: Uploads a replica to the list of replicas
2320           description: Uploads a replica to the list of replicas
2321           operationId: cloudmesh.replica.add
2322           requestBody:
2323             description: The replica to be uploaded
2324             required: true
2325             content:
2326               application/json:
2327                 schema:

```

```

2328         $ref: '#/components/schemas/Replica'
2329     responses:
2330         '200':
2331             description: Replica updated or created
2332         '401':
2333             description: Not authorized
2334     delete:
2335         tags:
2336             - Replica
2337         summary: Deletes the named replica
2338         description: Deletes an replica by name
2339         operationId: cloudmesh.replica.delete_by_name
2340         parameters:
2341             - name: name
2342               in: path
2343               required: true
2344               schema:
2345                 type: string
2346               description: The name of the replica
2347         responses:
2348             '200':
2349                 description: Deletion successful
2350             '401':
2351                 description: Not authorized
2352             '404':
2353                 description: The named replica could not be found
2354     components:
2355         schemas:
2356             Replica:
2357                 type: object
2358                 description: An entry representing a file replica record
2359                 properties:
2360                     name:
2361                         type: string
2362                         description: The name of the replica
2363                     filename:
2364                         type: string
2365                         description: The original filename
2366                     endpoint:
2367                         type: string
2368                         description: The location of the file
2369                     checksum:
2370                         type: string
2371                         description: The checksum of the file
2372                     size:
2373                         type: integer
2374                         description: The size of the file in byte

```

2375 **4.7.3 DATABASE**

2376 The database specification allows registration of a database and performance of elementary operations to
 2377 use the database. Distinguished below are actions related to the registration of a database, the adding of a
 2378 schema, the insertion of data, and the query of data. The database is defined by the name of an endpoint
 2379 (e.g., host, port) and the protocol used (e.g., SQL, NoSQL, graph-based databases).

2380 **4.7.3.1 Schema Database**

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

2381 **4.7.3.2 Schema Schema**

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

2382 **4.7.3.3 Schema Record**

Property	Type	Description
status	string	The status of the return
result	string	The result of the query in json string format

2383 **4.7.3.4 Schema Query**

Property	Type	Description
status	string	The query string

2384 **4.7.3.5 Paths**

HTTP	Path	Summary
get	/database	Returns all databases
get	/database/{name}/schema	Get the list of the schema
put	/database/{name}/schema	Upload a schema
delete	/database/{name}/schema	Deletes a database from the list of databases
get	/database/{name}	Query the named database
put	/database/{name}	add data to the table or collection
delete	/database/{name}	Delete the objects matching the query

2385 **4.7.3.5.1 /database**

2386 **4.7.3.5.1.1 GET /database**

2387 Returns all databases

2388 Responses

Code	Description	Schema
200	List of databases	array[Database]
401	Not authorized	String
404	Named database not found	String

2389 **4.7.3.5.2 /database/{name}/schema**

2390 **4.7.3.5.2.1 GET /database/{name}/schema**

2391 Responses

Code	Description	Schema
200	successfully returned the schema	array[Schema]
401	Not authorized	String
404	Named database not found	String

2392 Parameters

Name	Located in	Description	Required	Schema
name	path	Name of the schema	True	String

2393 **4.7.3.5.2.2 PUT /database/{name}/schema**

2394 Responses

Code	Description	Schema
200	successfully returned the list	Schema
401	Not authorized	String
404	Named database not found	String

2395 Parameters

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

2396 **4.7.3.5.2.3 DELETE /database/{name}/schema**

2397 Deletes a database from the list of databases

2398 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	Named database not found	String

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

2399 Parameters

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

2400 **4.7.3.5.3 /database/{name}**

2401 **4.7.3.5.3.1 GET /database/{name}**

2402 Query the named database

2403 Responses

Code	Description	Schema
200	Successful query	array[Record]
401	Not authorized	String
404	Named database not found	String

2404 Parameters

Name	Located in	Description	Required	Schema
name	path	Name of the database	True	String
query	query	Database Query	True	Query

2405 **4.7.3.5.3.2 PUT /database/{name}**

2406 Responses

Code	Description	Schema
200	successfully uploaded	Record
401	Not authorized	String
404	Named database not found	String

2407 Parameters

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

2408 Request Body

Located in	Description	Required	Schema
Body	Record to be uploaded	True	Record

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

2409 **4.7.3.5.3.3 DELETE /database/{name}**

2410 Responses

Code	Description	Schema
200	Successful query	array[Record]
401	Not authorized	String
404	Named database not found	String

2411 Parameters

Name	Located in	Description	Required	Schema
name	path	Name of the database	True	String
query	query	Database Query	True	Query

2412 **4.7.3.6 database.yaml**

```

2413 openapi: "3.0.2"
2414 info:
2415   version: 3.2.0
2416   x-date: 17-06-2019
2417   x-status: defined
2418   title: Database
2419   description: |-
2420
2421   The database specification allows to register a database and perform
2422   elementary operations to use this database. We distinguish actions
2423   related to the registration, the adding of a schema, the insertion of
2424   data and the query of data. The data base is defined by a name an endpoint
2425   (e.g., host, port), and a protocol used (e.g., SQL, NoSQL, graph-based databases)
2426
2427   termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
2428   contact:
2429     name: NIST BDRA Interface Subgroup
2430     url: https://cloudmesh-community.github.io/nist/spec/
2431   license:
2432     name: Apache 2.0
2433     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
2434 servers:
2435   - url: /cloudmesh/v3
2436 paths:
2437   /database:
2438     get:
2439       tags:
2440       - "Database Registry"
2441       summary: Returns all databases
2442       description: Returns all databases
2443       operationId: cloudmesh.database.get
2444       responses:
2445         '200':
2446           description: List of databases
2447           content:
2448             application/json:
2449               schema:
2450                 type: array
2451                 items:
2452                   $ref: "#/components/schemas/Database"
2453         '401':
2454           description: Not authorized
2455         '404':
2456           description: Named database not found
2457   /database/{name}/schema:
2458
  
```

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

2459 get:
2460   tags:
2461     - "Database Schema"
2462   summary: Get the list of the schema
2463   description: ""
2464   operationId: "cloudmesh.database.get.schema"
2465   parameters:
2466     - name: name
2467       description: Name of the schema
2468       in: path
2469       required: true
2470       schema:
2471         type: string
2472   responses:
2473     '200':
2474       description: "successfully returned the schema"
2475       content:
2476         application/json:
2477           schema:
2478             type: array
2479             items:
2480               $ref: "#/components/schemas/Schema"
2481     '401':
2482       description: Not authorized
2483     '404':
2484       description: Named database not found
2485 put:
2486   tags:
2487     - "Database Schema"
2488   summary: "Upload a schema"
2489   description: ""
2490   operationId: "cloudmesh.database.put.schema"
2491   parameters:
2492     - name: name
2493       description: Name of the database
2494       in: path
2495       required: true
2496       schema:
2497         type: string
2498   responses:
2499     '200':
2500       description: "successfully returned the list"
2501       content:
2502         application/json:
2503           schema:
2504             $ref: "#/components/schemas/Schema"
2505     '401':
2506       description: Not authorized
2507     '404':
2508       description: Named database not found
2509 delete:
2510   tags:
2511     - "Database Registry"
2512   summary: Deletes a database from the list of databases
2513   description: Deletes a database from the list of databases
2514   operationId: cloudmesh.database.delete
2515   parameters:
2516     - name: name
2517       description: Name of the database
2518       in: path
2519       required: true
2520       schema:
2521         type: string
2522   responses:
2523     '200':
2524       description: Deletion successful
2525     '401':
2526       description: Not authorized
2527     '404':

```

```

2528         description: Named database not found
2529 /database/{name}:
2530   get:
2531     tags:
2532     - "Database Data"
2533     summary: Query the named database
2534     description: Query the named database
2535     operationId: "cloudmesh.database.data.get"
2536     parameters:
2537     - name: name
2538       description: Name of the database
2539       in: path
2540       required: true
2541       schema:
2542         type: string
2543     - in: query
2544       name: query
2545       description: Database Query
2546       required: true
2547       schema:
2548         $ref: '#/components/schemas/Query'
2549     responses:
2550       '200':
2551         description: Successful query
2552         content:
2553           application/json:
2554             schema:
2555               type: array
2556               items:
2557                 $ref: "#/components/schemas/Record"
2558       '401':
2559         description: Not authorized
2560       '404':
2561         description: Named database not found
2562   put:
2563     tags:
2564     - "Database Data"
2565     summary: "add data to the table or collection"
2566     description: ""
2567     operationId: "cloudmesh.database.data.put"
2568     parameters:
2569     - name: name
2570       description: Name of the database
2571       in: path
2572       required: true
2573       schema:
2574         type: string
2575     requestBody:
2576       description: Record to be uploaded
2577       required: true
2578       content:
2579         application/json:
2580           schema:
2581             $ref: "#/components/schemas/Record"
2582     responses:
2583       '200':
2584         description: "successfully uploaded"
2585         content:
2586           application/json:
2587             schema:
2588               $ref: "#/components/schemas/Record"
2589       '401':
2590         description: Not authorized
2591       '404':
2592         description: Named database not found
2593   delete:
2594     tags:
2595     - "Database Data"
2596     summary: "Delete the objects matching the query"

```

```

2597     description: ""
2598     operationId: "cloudmesh.database.data.delete"
2599     parameters:
2600       - name: name
2601         description: Name of the database
2602         in: path
2603         required: true
2604         schema:
2605           type: string
2606       - name: query
2607         description: Database Query
2608         in: query
2609         required: true
2610         schema:
2611           $ref: '#/components/schemas/Query'
2612     responses:
2613       '200':
2614         description: Successful query
2615         content:
2616           application/json:
2617             schema:
2618               type: array
2619               items:
2620                 $ref: "#/components/schemas/Record"
2621       '401':
2622         description: Not authorized
2623       '404':
2624         description: Named database not found
2625     components:
2626       schemas:
2627         Database:
2628           type: object
2629           description: Defines a database object as an entry
2630           properties:
2631             name:
2632               type: string
2633               description: Name of the database
2634             description:
2635               type: string
2636               description: Description of the database
2637             endpoint:
2638               type: string
2639               description: Endpoint of the database
2640             kind:
2641               type: string
2642               description: the kind of the database
2643         Schema:
2644           type: object
2645           description: Defines a database
2646           properties:
2647             name:
2648               type: string
2649               description: Name of the database
2650             description:
2651               type: string
2652               description: Description of the database
2653             kind:
2654               type: string
2655               description: The kind of the definition
2656             content:
2657               type: string
2658               description: The schema associated with the table or collection
2659         Record:
2660           type: object
2661           description: The result of a query
2662           properties:
2663             status:
2664               type: string
2665               description: The status of the return

```

```

2666         result:
2667           type: string
2668           description: The result of the query in json string format
2669     Query:
2670       type: object
2671       description: The query
2672       properties:
2673         status:
2674           type: string
2675           description: The query string
    
```

2676 **4.7.4 VIRTUAL DIRECTORY**

2677 A virtual directory is a collection of files, replicas, streams, or other virtual directories.

2678 **4.7.4.1 Schema Virtualdirectory**

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

2679 **4.7.4.2 Paths**

HTTP	Path	Summary
get	/virtualdirectory	Returns a list of virtual directories
get	/virtualdirectory/{name}	Returns the named virtual directory
put	/virtualdirectory/{name}	Uploads a virtual directory to the list of virtual directories
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

2680 **4.7.4.2.1 /virtualdirectory**

2681 **4.7.4.2.1.1 GET /virtualdirectory**

2682 Returns a list of all virtual directories

2683 Responses

Code	Description	Schema
200	The list of virtual directories	array[Virtualdirectory]
401	Not authorized	String

2684 **4.7.4.2.2 /virtualdirectory/{name}**

2685 **4.7.4.2.2.1 GET /virtualdirectory/{name}**

2686 Returns a virtual directory by name

2687 Responses

Code	Description	Schema
200	Returning the information of the virtual directory	Virtualdirectory
401	Not authorized	String
404	The named virtual directory could not be found	String

2688 Parameters

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

2689 **4.7.4.2.2.2 PUT /virtualdirectory/{name}**

2690 Uploads a virtual directory to the list of virtual directories

2691 Responses

Code	Description	Schema
200	Virtual directory updated or created	String
401	Not authorized	String
404	The named virtual directory could not be found	String

2692 Request Body

Located in	Description	Required	Schema
Body	The virtual directory to be uploaded	True	Virtualdirectory

2693 **4.7.4.2.2.3 DELETE /virtualdirectory/{name}**

2694 Deletes a virtual directory by name

2695 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named virtual directory could not be found	String

2696 Parameters

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

2697 **4.7.4.2.3 /virtualdirectory/{name}/{filename}**

2698 **4.7.4.2.3.1 GET /virtualdirectory/{name}/{filename}**

2699 Returns the specific file of that virtual directory

2700 Responses

Code	Description	Schema
200	upload successful	File
401	Not authorized	String
404	The named virtual directory or file could not be found	String

2701 Parameters

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

2702 **4.7.4.2.3.2 PUT /virtualdirectory/{name}/{filename}**

2703 Updates or adds a virtual file in the virtual directory

2704 Responses

Code	Description	Schema
200	User added successfully	String
401	Not authorized	String

2705 Parameters

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

2706 Request Body

Located in	Description	Required	Schema
Body	The user to be uploaded	True	File

2707 **4.7.4.2.3.3 DELETE /virtualdirectory/{name}/{filename}**

2708 Delete a user in the virtual directory

2709 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named virtual directory or file could not be found	String

2710 Parameters

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

2711 **4.7.4.3 virtualdirectory.yaml**

```

2712 openapi: "3.0.2"
2713 info:
2714   version: 3.2.0
2715   x-date: 17-06-2019
2716   x-status: defined
2717   title: Virtual Directory
2718   description: |-
2719     A virtual directory is a collection of files, replicas, streams or other
2720     virtual directories.
2721
2722   termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
2723   contact:
2724     name: NIST BDRA Interface Subgroup Service
2725     url: https://cloudmesh-community.github.io/nist/spec/
2726   license:
2727     name: Apache 2.0
2728     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
2729 servers:
2730   - url: /cloudmesh/v3
2731 paths:
2732   /virtualdirectory:
2733     get:
2734       tags:
2735         - Virtual directory
2736       summary: Returns a list of virtual directories
2737       description: Returns a list of all virtual directories
2738       operationId: cloudmesh.virtual.directory.list
2739       responses:
2740         '200':
2741           description: The list of virtual directories
2742           content:
2743             application/json:
2744               schema:
2745                 type: array
2746                 items:
2747                   $ref: '#/components/schemas/Virtualdirectory'
2748         '401':
2749           description: Not authorized
2750
2751   /virtualdirectory/{name}:
2752     get:
2753       tags:
2754         - Virtual directory
2755       summary: Returns the named virtual directory
2756       description: Returns an virtual directory by name
2757       operationId: cloudmesh.virtualdirectory.find_by_name
2758
2759

```

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

2760     parameters:
2761     - name: name
2762       in: path
2763       required: true
2764       schema:
2765         type: string
2766       description: The name of the virtual directory
2767     responses:
2768     '200':
2769       description: Returning the information of the virtual directory
2770       content:
2771         application/json:
2772           schema:
2773             $ref: '#/components/schemas/Virtualdirectory'
2774     '401':
2775       description: Not authorized
2776     '404':
2777       description: The named virtual directory could not be found
2778   put:
2779     tags:
2780     - Virtual directory
2781     summary: Uploads a virtual directory to the list of virtual directories
2782     description: Uploads a virtual directory to the list of virtual directories
2783     operationId: cloudmesh.virtual_directory.add
2784     requestBody:
2785       description: The virtual directory to be uploaded
2786       required: true
2787       content:
2788         application/json:
2789           schema:
2790             $ref: '#/components/schemas/Virtualdirectory'
2791     responses:
2792     '200':
2793       description: Virtual directory updated or created
2794     '401':
2795       description: Not authorized
2796     '404':
2797       description: The named virtual directory could not be found
2798   delete:
2799     tags:
2800     - Virtual directory
2801     summary: Deletes the named virtual directory
2802     description: Deletes an virtual directory by name
2803     operationId: cloudmesh.virtualdirectory.delete_by_name
2804     parameters:
2805     - name: name
2806       in: path
2807       required: true
2808       schema:
2809         type: string
2810       description: The name of the virtual directory
2811     responses:
2812     '200':
2813       description: Deletion successful
2814     '401':
2815       description: Not authorized
2816     '404':
2817       description: The named virtual directory could not be found
2818 /virtualdirectory/{name}/{filename}:
2819   get:
2820     tags:
2821     - Virtual directory
2822     summary: Returns the specific file of that virtual directory
2823     description: Returns the specific file of that virtual directory
2824     operationId: cloudmesh.virtualdirectory.file.get_by_name
2825     parameters:
2826     - name: name
2827       in: path
2828       required: true

```

```

2829     schema:
2830       type: string
2831     description: The name of the virtual directory
2832   - name: filename
2833     description: The filename
2834     in: path
2835     required: true
2836     schema:
2837       type: string
2838 responses:
2839   '200':
2840     description: upload successful
2841     content:
2842       application/json:
2843         schema:
2844           $ref: "filestore.yaml#/components/schemas/File"
2845   '401':
2846     description: Not authorized
2847   '404':
2848     description: The named virtual directory or file could not be found
2849 put:
2850   tags:
2851   - Virtual directory
2852   summary: Updates or adds a virtual file in the virtual directory
2853   description: Updates or adds a virtual file in the virtual directory
2854   operationId: cloudmesh.virtualdirectory.file.add
2855   parameters:
2856   - name: name
2857     in: path
2858     required: true
2859     schema:
2860       type: string
2861     description: The name of the virtual directory
2862   - name: filename
2863     description: The filename
2864     in: path
2865     required: true
2866     schema:
2867       type: string
2868   requestBody:
2869     description: The user to be uploaded
2870     required: true
2871     content:
2872       application/json:
2873         schema:
2874           $ref: "filestore.yaml#/components/schemas/File"
2875 responses:
2876   '200':
2877     description: User added successfully
2878   '401':
2879     description: Not authorized
2880   '404':
2881     description: The named virtual directory or file could not be found
2882 delete:
2883   tags:
2884   - Virtual directory
2885   summary: Delete an user in the virtual directory
2886   description: Delete an user in the virtual directory
2887   operationId: cloudmesh.virtualdirectory.file.delete
2888   parameters:
2889   - name: name
2890     in: path
2891     required: true
2892     schema:
2893       type: string
2894     description: The name of the virtual directory
2895   - name: filename
2896     description: The filename
2897     in: path

```

```

2898         required: true
2899         schema:
2900           type: string
2901       responses:
2902         '200':
2903           description: Deletion successful
2904         '401':
2905           description: Not authorized
2906         '404':
2907           description: The named virtual directory or file could not be found
2908 components:
2909   schemas:
2910     Virtualdirectory:
2911       type: object
2912       description: the virtual directory
2913       properties:
2914         name:
2915           description: The name of the virtual directory
2916           type: string
2917         description:
2918           description: Description of the virtual directory
2919           type: string
2920         host:
2921           description: Remote host of the virtual directory
2922           type: string
2923         location:
2924           description: Remote location, e.g., a directory with full path on a
2925             host
2926           type: string
2927         protocol:
2928           description: Access protocol (e.g. HTTP, FTP, SSH, etc.)
2929           type: string
2930         credential:
2931           description: Credential to access
2932           type: object
    
```

2933 4.8 COMPUTE MANAGEMENT: VIRTUAL CLUSTERS

2934 4.8.1 VIRTUAL CLUSTER

2935 A Virtual Cluster is modeled as manager node, and one or more compute nodes. The manager node
 2936 usually serves as a login node and can be accessed from outside via a public IP. The compute nodes are
 2937 connected to the manager node via a private, usually high performance (high throughput and low latency),
 2938 network and, thus, are accessible only from the manager node. To use the virtual cluster, login to the
 2939 manager node, from which one can login to any compute node, or submit a job to run on the compute
 2940 nodes.

2941 4.8.1.1 Schema Virtualcluster

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	Node	Manager node of the virtual cluster
nodes	array[Node]	List of nodes of the virtual cluster

2942 **4.8.1.2 Schema Node**

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
nic	array[NIC]	List of network interfaces of the node

2943 **4.8.1.3 Schema NIC**

Property	Type	Description
mac	string	Media Access Control (MAC) address of the node
ip	string	IP address of the node

2944 **4.8.1.4 Paths**

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

2945 **4.8.1.4.1 /virtualcluster**

2946 **4.8.1.4.1.1 GET /virtualcluster**

2947 Returns a list of all virtual clusters

2948 Responses

Code	Description	Schema
200	The list of virtual clusters	array[Virtualcluster]
401	Not authorized	String

2949 **4.8.1.4.2 /virtualcluster/{name}**

2950 **4.8.1.4.2.1 GET /virtualcluster/{name}**

2951 Returns a virtual cluster by name

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

2952 Responses

Code	Description	Schema
200	Returning the information of the virtual cluster	Virtualcluster
401	Not authorized	String
404	The named virtual cluster could not be found	String

2953 Parameters

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

2954 **4.8.1.4.2.2 PUT /virtualcluster/{name}**

2955 Uploads a virtual cluster to the list of virtual clusters

2956 Responses

Code	Description	Schema
200	Virtual cluster updated or created	String
401	Not authorized	String
404	The named virtual cluster could not be found	String

2957 Request Body

Located in	Description	Required	Schema
Body	The virtual cluster to be uploaded	True	Virtualcluster

2958 **4.8.1.4.2.3 DELETE /virtualcluster/{name}**

2959 Deletes a virtual cluster by name

2960 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named virtual cluster could not be found	String

2961 Parameters

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

2962 **4.8.1.4.3 /virtualcluster/{name}/{node}**

2963 **4.8.1.4.3.1 GET /virtualcluster/{name}/{node}**

2964 Returns the specific node of the named virtual cluster. If the node name is manager, the manager node is used. A compute node cannot be named manager.

2966 Responses

Code	Description	Schema
200	Node info	Node
401	Not authorized	String
404	The named virtual cluster or node could not be found	String

2967 Parameters

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

2968 **4.8.1.4.3.2 PUT /virtualcluster/{name}/{node}**

2969 Updates or adds a node to the virtual cluster. If the node name is manager, the manager node is uploaded.

2970 Responses

Code	Description	Schema
200	Node added successfully	String
401	Not authorized	String
404	The named virtual cluster or node could not be found	String

2971 Parameters

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

2972 Request Body

Located in	Description	Required	Schema
Body	The node to be uploaded	True	Node

2973 **4.8.1.4.3.3 DELETE /virtualcluster/{name}/{node}**

2974 Delete a node in the virtual cluster

2975 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named virtual cluster or node could not be found	String

2976 Parameters

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

2977 **4.8.1.5 virtualcluster.yaml**

2978
 2979 openapi: "3.0.2"
 2980 info:
 2981 version: 3.2.0
 2982 x-date: 17-06-2019
 2983 x-status: defined
 2984 title: Virtual Cluster
 2985 description: |-
 2986
 2987 A Virtual Cluster is modeled as manager node, and one or more
 2988 compute nodes. The manager node usually serves as a login node and
 2989 can be accessed from outside via a public IP. The compute nodes are
 2990 connected to the manager node via a private, usually high performance (high
 2991 throughput and low latency) network and thus accessible only from the
 2992 manager node. To use the virtual cluster, login to the manager node, and
 2993 from there one can login to any compute node, or submit a job to run on the
 2994 compute nodes.
 2995
 2996 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
 2997 contact:
 2998 name: NIST BDRA Interface Subgroup Service
 2999 url: https://cloudmesh-community.github.io/nist/spec/
 3000 license:
 3001 name: Apache 2.0
 3002 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
 3003 servers:
 3004 - url: /cloudmesh/v3
 3005 paths:
 3006 /virtualcluster:
 3007 get:
 3008 tags:
 3009 - Virtual cluster
 3010 summary: Returns a list of virtual clusters
 3011 description: Returns a list of all virtual clusters
 3012 operationId: cloudmesh.virtualcluster.list
 3013 responses:
 3014 '200':
 3015 description: The list of virtual clusters
 3016 content:
 3017 application/json:
 3018 schema:
 3019 type: array
 3020 items:
 3021 \$ref: '#/components/schemas/Virtualcluster'
 3022 '401':
 3023 description: Not authorized
 3024 /virtualcluster/{name}:
 3025 get:
 3026 tags:
 3027 - Virtual cluster
 3028 summary: Returns the named virtual cluster
 3029 description: Returns an virtual cluster by name
 3030 operationId: cloudmesh.virtualcluster.find_by_name
 3031 parameters:
 3032 - name: name
 3033 in: path
 3034 required: true
 3035 schema:
 3036 type: string
 3037 description: The name of the virtual cluster
 3038 responses:
 3039 '200':
 3040 description: Returning the information of the virtual cluster
 3041 content:
 3042 application/json:
 3043 schema:
 3044 \$ref: '#/components/schemas/Virtualcluster'


```

3045     '401':
3046         description: Not authorized
3047     '404':
3048         description: The named virtual cluster could not be found
3049 put:
3050     tags:
3051     - Virtual cluster
3052     summary: Uploads an virtual cluster to the list of virtual clusters
3053     description: Uploads an virtual cluster to the list of virtual clusters
3054     operationId: cloudmesh.virtualcluster.add
3055     requestBody:
3056         description: The virtual cluster to be uploaded
3057         required: true
3058         content:
3059             application/json:
3060                 schema:
3061                     $ref: '#/components/schemas/Virtualcluster'
3062     responses:
3063         '200':
3064             description: Virtual cluster updated or created
3065         '401':
3066             description: Not authorized
3067         '404':
3068             description: The named virtual cluster could not be found
3069 delete:
3070     tags:
3071     - Virtual cluster
3072     summary: Deletes the named virtual cluster
3073     description: Deletes an virtual cluster by name
3074     operationId: cloudmesh.virtualcluster.delete_by_name
3075     parameters:
3076     - name: name
3077       in: path
3078       required: true
3079       schema:
3080         type: string
3081       description: The name of the virtual cluster
3082     responses:
3083         '200':
3084             description: Deletion successful
3085         '401':
3086             description: Not authorized
3087         '404':
3088             description: The named virtual cluster could not be found
3089 /virtualcluster/{name}/{node}:
3090 get:
3091     tags:
3092     - Virtual cluster
3093     summary: Node of the named virtual cluster
3094     description: Returns the specific node of the named virtual cluster. If
3095                 the node name is manager, the manager node is used. A
3096                 compute node can not be named manager
3097     operationId: cloudmesh.virtualcluster.node.get_by_name
3098     parameters:
3099     - name: name
3100       in: path
3101       required: true
3102       schema:
3103         type: string
3104       description: The name of the virtual cluster
3105     - name: node
3106       description: The node name
3107       in: path
3108       required: true
3109       schema:
3110         type: string
3111     responses:
3112         '200':
3113             description: Node info

```

```

3114         content:
3115           application/json:
3116             schema:
3117               $ref: "#/components/schemas/Node"
3118         '401':
3119           description: Not authorized
3120         '404':
3121           description: The named virtual cluster or node could not be found
3122     put:
3123       tags:
3124         - Virtual cluster
3125       summary: Updates or adds a node to the virtual cluster
3126       description: Updates or adds a node to the virtual cluster. If
3127         the node name is manager, the manager node is uploaded.
3128       operationId: cloudmesh.virtualcluster.node.add
3129       parameters:
3130         - name: name
3131           in: path
3132           required: true
3133           schema:
3134             type: string
3135           description: The name of the virtual cluster
3136         - name: node
3137           description: The node name
3138           in: path
3139           required: true
3140           schema:
3141             type: string
3142       requestBody:
3143         description: The node to be uploaded
3144         required: true
3145         content:
3146           application/json:
3147             schema:
3148               $ref: '#/components/schemas/Node'
3149       responses:
3150         '200':
3151           description: Node added successfully
3152         '401':
3153           description: Not authorized
3154         '404':
3155           description: The named virtual cluster or node could not be found
3156     delete:
3157       tags:
3158         - Virtual cluster
3159       summary: Delete a node in the virtual cluster
3160       description: Delete a node in the virtual cluster
3161       operationId: cloudmesh.virtualcluster.node.delete
3162       parameters:
3163         - name: name
3164           in: path
3165           required: true
3166           schema:
3167             type: string
3168           description: The name of the virtual cluster
3169         - name: node
3170           description: The node name
3171           in: path
3172           required: true
3173           schema:
3174             type: string
3175       responses:
3176         '200':
3177           description: Deletion successful
3178         '401':
3179           description: Not authorized
3180         '404':
3181           description: The named virtual cluster or node could not be found
3182     components:

```

```

3183 schemas:
3184   Virtualcluster:
3185     type: object
3186     properties:
3187       name:
3188         description: The name of the virtual cluster
3189         type: string
3190       description:
3191         type: string
3192         description: A description of the virtual cluster
3193       owner:
3194         type: string
3195         description: Username of the owner of the virtual cluster
3196       manager:
3197         description: Manager node of the virtual cluster
3198         $ref: "#/components/schemas/Node"
3199       nodes:
3200         description: List of nodes of the virtual cluster
3201         type: array
3202         items:
3203           $ref: "#/components/schemas/Node"
3204   Node:
3205     type: object
3206     properties:
3207       name:
3208         type: string
3209         description: Name of the node
3210       state:
3211         type: string
3212         description: Power state of the node
3213       ncpu:
3214         type: integer
3215         description: Number of virtual CPUs of the node
3216       ram:
3217         type: string
3218         description: RAM size of the node
3219       disk:
3220         type: string
3221         description: Disk size of the node
3222       nics:
3223         type: array
3224         description: List of network interfaces of the node
3225         items:
3226           $ref: "#/components/schemas/NIC"
3227   NIC:
3228     type: object
3229     properties:
3230       mac:
3231         type: string
3232         description: MAC address of the node
3233       ip:
3234         type: string
3235         description: IP address of the node

```

3236 4.8.2 NETWORK OF NODES

3237 A Network of Nodes (NON) contains a number of compute nodes that are connected by a network and
3238 can be reached from the compute nodes. The concept is a generalization of the term Network of
3239 Workstations. In contrast to a Virtual Cluster, it does not have a dedicated manager node. Network of
3240 nodes can be real or virtual. The same security context can be used to authenticate to all nodes in the
3241 network of nodes. This is typically done with a public key store in which all keys are stored that allow
3242 access to the nodes.

3243 **4.8.2.1 Schema Non**

Property	Type	Description
name	string	The name of the network of nodes
description	string	A description of the network of nodes
nodes	array[Node]	List of nodes of the network of nodes

3244 **4.8.2.2 Paths**

HTTP	Path	Summary
get	/non	Returns a list of network of nodes
put	/non/{name}	Uploads a network of nodes to the list of network of nodes
get	/non/{name}/publickeystore	Returns the information of the key store
delete	/non/{name}/publickeystore	Deletes the key store
put	/non/{name}/publickeystore	Adds a key store
get	/non/{name}/node	Returns the named network of nodes
delete	/non/{name}/node	Deletes the named network of nodes
get	/non/{name}/node/{node}	Node of the named network of nodes
put	/non/{name}/node/{node}	Updates or adds a node to the network of nodes
delete	/non/{name}/node/{node}	Delete a node in the network of nodes

3245 **4.8.2.2.1 /non**

3246 **4.8.2.2.1.1 GET /non**

3247 Returns a list of all network of nodes

3248 Responses

Code	Description	Schema
200	The list of network of nodes	array[Non]
401	Not authorized	String

3249 **4.8.2.2.2 /non/{name}**

3250 **4.8.2.2.2.1 PUT /non/{name}**

3251 Uploads a network of nodes to the list of network of nodes

3252 Responses

Code	Description	Schema
200	Network of nodes updated or created	String
400	Error updating network of nodes	String
401	Not authorized	String

3253 Request Body

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

Located in	Description	Required	Schema
Body	The network of nodes to be uploaded	True	Non

3254 **4.8.2.2.3 /non/{name}/publickeystore**

3255 **4.8.2.2.3.1 GET /non/{name}/publickeystore**

3256 Returns a network of nodes by name

3257 Responses

Code	Description	Schema
200	Returning the information of the network of nodes	string
401	Not authorized	String

3258 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the network of nodes	True	String

3259 **4.8.2.2.3.2 DELETE /non/{name}/publickeystore**

3260 Deletes a network of nodes by name

3261 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String

3262 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the network of nodes	True	String

3263 **4.8.2.2.3.3 PUT /non/{name}/publickeystore**

3264 Updates or adds a node to the network of nodes.

3265 Responses

Code	Description	Schema
200	Node keystore added successfully	String
401	Not authorized	String

3266 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the network of nodes	True	String

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

3267 **4.8.2.2.4 /non/{name}/node**

3268 **4.8.2.2.4.1 GET /non/{name}/node**

3269 Returns a network of nodes by name

3270 Responses

Code	Description	Schema
200	Returning the information of the node	Non
401	Not authorized	String
404	The named network of nodes could not be found	String

3271 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the network of nodes	True	String

3272 **4.8.2.2.4.2 DELETE /non/{name}/node**

3273 Deletes a network of nodes by name

3274 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named network of nodes could not be found	String

3275 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the network of nodes	True	String

3276 **4.8.2.2.5 /non/{name}/node/{node}**

3277 **4.8.2.2.5.1 GET /non/{name}/node/{node}**

3278 Returns the specific node of the named network of nodes.

3279 Responses

Code	Description	Schema
200	Node info	Node
401	Not authorized	String

3280 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the network of nodes	True	String
node	path	The node name	True	String

3281 **4.8.2.2.5.2 PUT /non/{name}/node/{node}**

3282 Updates or adds a node to the network of nodes

3283 Responses

Code	Description	Schema
200	Node added successfully	String
401	Not authorized	String

3284 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the network of nodes	True	String
node	path	The node name	True	String

3285 Request Body

Located in	Description	Required	Schema
Body	The node to be uploaded	True	Node

3286 **4.8.2.2.5.3 DELETE /non/{name}/node/{node}**

3287 Delete a node in the network of nodes

3288 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String

3289 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the network of nodes	True	String
node	path	The node name	True	String

3290 **4.8.2.3 non.yaml**

3291 openapi: "3.0.2"

3292 info:

3293 version: 3.2.0

3294 x-date: 17-06-2019

3295 x-status: defined

3296 title: Network of Nodes

3297 description: |-

3298
 3299 A Network of Nodes (NON) contains a number of compute nodes that are
 3300 connected by a network and can be reached from each other. The concept is a
 3301 generalization of the term Network of Workstations. In contrast to a
 3302 Virtual Cluster it does not have a dedicated manager node. Network of
 3303 nodes can be real or virtual. The same security context can be used to
 3304 authenticate to all nodes in the network of nodes. This is typically done
 3305 with a public key store in which all keys are stored that allow access to
 3306 the nodes.

3307
 3308 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
 3309 contact:

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

3310     name: NIST BDRA Interface Subgroup Service
3311     url: https://cloudmesh-community.github.io/nist/spec/
3312   license:
3313     name: Apache 2.0
3314     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
3315   servers:
3316     - url: /cloudmesh/v3
3317   paths:
3318     /non:
3319     get:
3320       tags:
3321         - Network of nodes
3322       summary: Returns a list of network of nodes
3323       description: Returns a list of all network of nodes
3324       operationId: cloudmesh.non.list
3325       responses:
3326         '200':
3327           description: The list of network of nodes
3328           content:
3329             application/json:
3330               schema:
3331                 type: array
3332                 items:
3333                   $ref: '#/components/schemas/Non'
3334         '401':
3335           description: Not authorized
3336     /non/{name}:
3337     put:
3338       tags:
3339         - Network of nodes
3340       summary: Uploads a network of nodes to the list of network of nodes
3341       description: Uploads a network of nodes to the list of network of nodes
3342       operationId: cloudmesh.non.add
3343       requestBody:
3344         description: The network of nodes to be uploaded
3345         required: true
3346         content:
3347           application/json:
3348             schema:
3349               $ref: '#/components/schemas/Non'
3350       responses:
3351         '200':
3352           description: Network of nodes updated or created.
3353         '400':
3354           description: Error updating network of nodes
3355         '401':
3356           description: Not authorized
3357     /non/{name}/publickeystore:
3358     get:
3359       tags:
3360         - Non
3361       summary: Returns the information of the key store
3362       description: Returns a network of nodes by name
3363       operationId: cloudmesh.non.keystore.find_by_name
3364       parameters:
3365         - name: name
3366           in: path
3367           required: true
3368           schema:
3369             type: string
3370       description: The name of the network of nodes
3371       responses:
3372         '200':
3373           description: Returning the information of the network of nodes
3374           content:
3375             application/json:
3376               schema:
3377                 type: string
3378           description: the endpoint of the publickeystore

```



```

3379     '401':
3380       description: Not authorized
3381 delete:
3382   tags:
3383     - Network of nodes
3384   summary: Deletes the keystore
3385   description: Deletes a network of nodes by name
3386   operationId: cloudmesh.non.keystore.delete
3387   parameters:
3388     - name: name
3389       in: path
3390       required: true
3391       schema:
3392         type: string
3393       description: The name of the network of nodes
3394   responses:
3395     '200':
3396       description: Deletion successful
3397     '401':
3398       description: Not authorized
3399 put:
3400   tags:
3401     - Network of nodes
3402   summary: Adds a keystore
3403   description: Updates or adds a node to the network of nodes.
3404   operationId: cloudmesh.non.keystore.add
3405   parameters:
3406     - name: name
3407       in: path
3408       required: true
3409       schema:
3410         type: string
3411       description: The name of the network of nodes
3412   responses:
3413     '200':
3414       description: Node keystore added successfully
3415     '401':
3416       description: Not authorized
3417 /non/{name}/node:
3418 get:
3419   tags:
3420     - Non
3421   summary: Returns the named network of nodes
3422   description: Returns a network of nodes by name
3423   operationId: cloudmesh.non.find_by_name
3424   parameters:
3425     - name: name
3426       in: path
3427       required: true
3428       schema:
3429         type: string
3430       description: The name of the network of nodes
3431   responses:
3432     '200':
3433       description: Returning the information of the node
3434       content:
3435         application/json:
3436           schema:
3437             $ref: '#/components/schemas/Non'
3438     '401':
3439       description: Not authorized
3440     '404':
3441       description: The named network of nodes could not be found
3442 delete:
3443   tags:
3444     - Network of nodes
3445   summary: Deletes the named network of nodes
3446   description: Deletes a network of nodes by name
3447   operationId: cloudmesh.non.delete_by_name

```

```

3448     parameters:
3449       - name: name
3450         in: path
3451         required: true
3452         schema:
3453           type: string
3454         description: The name of the network of nodes
3455     responses:
3456       '200':
3457         description: Deletion successful
3458       '401':
3459         description: Not authorized
3460       '404':
3461         description: The named network of nodes could not be found
3462 /non/{name}/node/{node}:
3463 get:
3464   tags:
3465     - Network of nodes
3466   summary: Node of the named network of nodes
3467   description: Returns the specific node of the named network of nodes.
3468   operationId: cloudmesh.non.node.get_by_name
3469   parameters:
3470     - name: name
3471       in: path
3472       required: true
3473       schema:
3474         type: string
3475       description: The name of the network of nodes
3476     - name: node
3477       description: The node name
3478       in: path
3479       required: true
3480       schema:
3481         type: string
3482   responses:
3483     '200':
3484       description: Node info
3485       content:
3486         application/json:
3487           schema:
3488             $ref: "virtualcluster.yaml#/components/schemas/Node"
3489     '401':
3490       description: Not authorized
3491 put:
3492   tags:
3493     - Network of nodes
3494   summary: Updates or adds a node to the network of nodes
3495   description: Updates or adds a node to the network of nodes
3496   operationId: cloudmesh.non.node.add
3497   parameters:
3498     - name: name
3499       in: path
3500       required: true
3501       schema:
3502         type: string
3503       description: The name of the network of nodes
3504     - name: node
3505       description: The node name
3506       in: path
3507       required: true
3508       schema:
3509         type: string
3510   requestBody:
3511     description: The node to be uploaded
3512     required: true
3513     content:
3514       application/json:
3515         schema:
3516           $ref: "virtualcluster.yaml#/components/schemas/Node"

```

```

3517     responses:
3518       '200':
3519         description: Node added sucessfully
3520       '401':
3521         description: Not authorized
3522     delete:
3523       tags:
3524         - Network of nodes
3525       summary: Delete a node in the network of nodes
3526       description: Delete a node in the network of nodes
3527       operationId: cloudmesh.non.node.delete
3528       parameters:
3529         - name: name
3530           in: path
3531           required: true
3532           schema:
3533             type: string
3534           description: The name of the network of nodes
3535         - name: node
3536           description: The node name
3537           in: path
3538           required: true
3539           schema:
3540             type: string
3541     responses:
3542       '200':
3543         description: Deletion successful
3544       '401':
3545         description: Not authorized
3546     components:
3547       schemas:
3548         Non:
3549           type: object
3550           properties:
3551             name:
3552               description: The name of the network of nodes
3553               type: string
3554             description:
3555               type: string
3556               description: A description of the network of nodes
3557             nodes:
3558               description: List of nodes of the network of nodes
3559               type: array
3560             items:
3561               $ref: "virtualcluster.yaml#/components/schemas/Node"

```

3562 **4.8.3 SCHEDULER**

3563 A scheduler allows for control of the execution of tasks based on a policy. Schedulers may allow the
 3564 assignment of different policies to define the order of the tasks. A scheduler returns the next task to be
 3565 executed. Tasks can be added and deleted.

3566 **4.8.3.1 Schema Task**

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

3567 **4.8.3.2 Schema Policy**

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 behavior of the scheduler

3568 **4.8.3.3 Paths**

HTTP	Path	Summary
get	/task	Returns a list of tasks
get	/task/{name}	Returns the named task
put	/task/{name}	Uploads a task to the list of tasks
delete	/task/{name}	Deletes the named task
get	/policy	Returns the policy found
put	/policy	Uploads the policy

3569 **4.8.3.3.1 /task**

3570 **4.8.3.3.1.1 GET /task**

3571 Returns a list of all tasks

3572 Responses

Code	Description	Schema
200	The list of tasks	Array[Task]
401	Not authorized	String

3573 **4.8.3.3.2 /task/{name}**

3574 **4.8.3.3.2.1 GET /task/{name}**

3575 Returns a task by name

3576 Responses

Code	Description	Schema
200	Returning the information of the task	Task
401	Not authorized	String
404	The named task could not be found	String

3577 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the task	True	String
operation	query	Show the task but do not remove it from the queue	False	String

3578 **4.8.3.3.2.2 PUT /task/{name}**

3579 Uploads a task to the list of tasks

3580 Responses

Code	Description	Schema
200	Task updated	String
401	Not authorized	String

3581 Parameters

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

3582 Request Body

Located in	Description	Required	Schema
Body	The task to be uploaded	True	Task

3583 **4.8.3.3.2.3 DELETE /task/{name}**

3584 Deletes a task by name

3585 Responses

Code	Description	Schema
200	Deletion successful	String
400	Error to delete the task	String
401	Not authorized	String
404	The named task could not be found	String

3586 Parameters

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

3587 **4.8.3.3.3 /policy**

3588 **4.8.3.3.3.1 GET /policy**

3589 Returns the policy

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

3590 Responses

Code	Description	Schema
200	The policy	array[Policy]
401	Not authorized	String

3591 **4.8.3.3.3.2 PUT /policy**

3592 Uploads a task to the list of tasks

3593 Responses

Code	Description	Schema
200	Task updated	String
400	Error updating	String
401	Not authorized	String

3594 Request Body

Located in	Description	Required	Schema
Body	The policy to be uploaded	True	Policy

3595 **4.8.3.4 queue.yaml**

3596 openapi: "3.0.2"

3597 info:

3598 version: 3.2.0

3599 x-date: 17-06-2019

3600 x-status: defined

3601 title: Queue

3602 description: |-

3603

3604 A scheduler allows for control of the execution of tasks based on a

3605 policy. Schedulers may allow the assignment of different policies to

3606 define the order of the tasks. A scheduler returns the next task to be

3607 executed. Tasks can be added and deleted.

3608 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"

3609 contact:

3610 name: NIST BDRA Interface Subgroup

3611 url: https://cloudmesh-community.github.io/nist

3612 license:

3613 name: Apache 2.0

3614 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt

3615 servers:

3616 - url: /cloudmesh/v3/scheduler

3617 paths:

3618 /task:

3619 get:

3620 tags:

3621 - Scheduler

3622 summary: Returns a list of tasks

3623 description: Returns a list of all tasks

3624 operationId: cloudmesh.scheduler.task.list

3625 responses:

3626 '200':

3627 description: The list of tasks

3628 content:

3629 application/json:

3630 schema:

3631 type: array

3632 items:

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

3633         $ref: '#/components/schemas/Task'
3634     '401':
3635         description: Not authorized
3636 /task/{name}:
3637     get:
3638         tags:
3639             - Scheduler
3640         summary: Returns the named task
3641         description: Returns an task by name
3642         operationId: cloudmesh.scheduler.task.find_by_name
3643         parameters:
3644             - name: name
3645               in: path
3646               required: true
3647               schema:
3648                 type: string
3649               description: The name of the task
3650             - in: query
3651               name: operation
3652               description: Show the task but do not remove it from the queue
3653               schema:
3654                 type: string
3655                 enum:
3656                     - info
3657         responses:
3658             '200':
3659                 description: Returning the information of the task
3660                 content:
3661                     application/json:
3662                         schema:
3663                             $ref: '#/components/schemas/Task'
3664             '401':
3665                 description: Not authorized
3666             '404':
3667                 description: The named task could not be found
3668     put:
3669         tags:
3670             - Scheduler
3671         summary: Uploads a task to the list of tasks
3672         description: Uploads a task to the list of tasks
3673         operationId: cloudmesh.scheduler.task.add
3674         parameters:
3675             - name: name
3676               in: path
3677               required: true
3678               schema:
3679                 type: string
3680               description: The name of the task
3681         requestBody:
3682             description: The task to be uploaded
3683             required: true
3684             content:
3685                 application/json:
3686                     schema:
3687                         $ref: '#/components/schemas/Task'
3688         responses:
3689             '200':
3690                 description: Task updated
3691             '401':
3692                 description: Not authorized
3693     delete:
3694         tags:
3695             - Scheduler
3696         summary: Deletes the named task
3697         description: Deletes an task by name
3698         operationId: cloudmesh.scheduler.task.delete_by_name
3699         parameters:
3700             - name: name
3701               in: path

```

```

3702         required: true
3703         schema:
3704           type: string
3705         description: The name of the task
3706     responses:
3707       '200':
3708         description: Deletion successful
3709       '400':
3710         description: Error to delete the task
3711       '401':
3712         description: Not authorized
3713       '404':
3714         description: The named task could not be found
3715 /policy:
3716   get:
3717     tags:
3718       - Scheduler
3719     summary: Returns the policy found
3720     description: Returns the policy
3721     operationId: cloudmesh.scheduler.policy.list
3722     responses:
3723       '200':
3724         description: The policy
3725         content:
3726           application/json:
3727             schema:
3728               type: array
3729               items:
3730                 $ref: '#/components/schemas/Policy'
3731       '401':
3732         description: Not authorized
3733   put:
3734     tags:
3735       - Scheduler
3736     summary: Uploads the policy
3737     description: Uploads a task to the list of tasks
3738     operationId: cloudmesh.scheduler.policy.add
3739     requestBody:
3740       description: The policy to be uploaded
3741       required: true
3742       content:
3743         application/json:
3744           schema:
3745             $ref: '#/components/schemas/Policy'
3746     responses:
3747       '200':
3748         description: Task updated
3749       '400':
3750         description: Error updating
3751       '401':
3752         description: Not authorized
3753 components:
3754   schemas:
3755     Task:
3756       type: object
3757       description: the scheduler
3758       properties:
3759         name:
3760           type: string
3761           description: name of the scheduler
3762         user:
3763           type: string
3764           description: the username the task belongs to
3765         description:
3766           type: string
3767           description: The description of the task
3768         kind:
3769           type: string
3770           description: The kind of the task

```



```

3771 Policy:
3772   type: object
3773   description: The policy of the scheduler
3774   properties:
3775     name:
3776       type: string
3777       description: name of the scheduler policy
3778     description:
3779       type: string
3780       description: The description of the policy
3781     kind:
3782       type: string
3783       description: The kind of the policy
3784     parameters:
3785       type: string
3786       description: parameters to define the behavior of the scheduler
    
```

3787 **4.8.4 QUEUE**

3788 The queue is a special scheduler that allows tasks to be scheduled while doing queue policies, such as
 3789 last-in-first-out (LIFO), first-in-first-out (FIFO), and others. A queue returns the next task to be executed.
 3790 Tasks can be added and deleted.

3791 **4.8.4.1 Schema Task**

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
Content	String	The content of the task

3792 **4.8.4.2 Schema Policy**

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 behavior of the scheduler

3793 **4.8.4.3 Paths**

HTTP	Path	Summary
get	/task	Returns a list of tasks
get	/task/{name}	Returns the named task
put	/task/{name}	Uploads a task to the list of tasks
delete	/task/{name}	Deletes the named task
get	/policy	Returns the policy
put	/policy	Uploads the policy

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

3794 **4.8.4.3.1 /task**

3795 **4.8.4.3.1.1 GET /task**

3796 Returns a list of all tasks

3797 Responses

Code	Description	Schema
200	The list of tasks	Array[Task]
400	No tasks found	String
401	Not authorized	String

3798 **4.8.4.3.2 /task/{name}**

3799 **4.8.4.3.2.1 GET /task/{name}**

3800 Returns a task by name

3801 Responses

Code	Description	Schema
200	Returning the information of the task	Task
400	No task found	String
401	Not authorized	String
404	The named task could not be found	String

3802 Parameters

Name	Located in	Description	Required	Schema
Name	path	The name of the task	True	String
operation	query	ERROR: description missing	False	String

3803 **4.8.4.3.2.2 PUT /task/{name}**

3804 Uploads a task to the list of tasks

3805 Responses

Code	Description	Schema
200	Task updated	String
400	Error updating task.	String
401	Not authorized	String

3806 Parameters

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

3807 Request Body

Located in	Description	Required	Schema
Body	The task to be uploaded	True	Task

3808 **4.8.4.3.2.3 DELETE /task/{name}**

3809 Deletes a task by name

3810 Responses

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

3811 Parameters

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

3812 **4.8.4.3.3 /policy**

3813 **4.8.4.3.3.1 GET /policy**

3814 Returns the policy

3815 Responses

Code	Description	Schema
200	The policy	array[Policy]
400	No tasks found	String
401	Not authorized	String

3816 **4.8.4.3.3.2 PUT /policy**

3817 Uploads a task to the list of tasks

3818 Responses

Code	Description	Schema
200	Task updated	String
400	Error updating task	String
401	Not authorized	String

3819 Request Body

Located in	Description	Required	Schema
Body	The policy	True	Policy

3820 **4.8.4.4 scheduler.yaml**

```

3821 openapi: "3.0.2"
3822 info:
3823   version: 3.2.0
3824   x-date: 17-06-2019
3825   x-status: defined
3826   title: Scheduler
3827   description: |-
3828
3829     The queue is a special scheduler that allows tasks to be scheduled
3830     while doing queue policies, such as LIFO, FIFO, and so on.
3831     A queue returns the next task to be executed. Tasks can be added
3832     and
3833     deleted.
3834
3835   termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
3836   contact:
3837     name: NIST BDRA Interface Subgroup
3838     url: https://cloudmesh-community.github.io/nist
3839   license:
3840     name: Apache 2.0
3841     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
3842 servers:
3843   - url: /cloudmesh/v3/scheduler
3844 paths:
3845   /task:
3846     get:
3847       tags:
3848         - Task
3849       summary: Returns a list of tasks
3850       description: Returns a list of all tasks
3851       operationId: cloudmesh.task.list
3852       responses:
3853         '200':
3854           description: The list of tasks
3855           content:
3856             application/json:
3857               schema:
3858                 type: array
3859                 items:
3860                   $ref: '#/components/schemas/Task'
3861         '400':
3862           description: No tasks found
3863         '401':
3864           description: Not authorized
3865   /task/{name}:
3866     get:
3867       tags:
3868         - Task
3869       summary: Returns the named task
3870       description: Returns an task by name
3871       operationId: cloudmesh.task.find_by_name
3872       parameters:
3873         - name: name
3874           in: path
3875           required: true
3876           schema:
3877             type: string
3878           description: The name of the task
3879         - in: query
3880           name: operation
3881           schema:
3882             type: string
3883             enum:
3884               - info
3885               - pop
3886       responses:
3887         '200':

```

```

3888     description: Returning the information of the task
3889     content:
3890       application/json:
3891         schema:
3892           $ref: '#/components/schemas/Task'
3893     '400':
3894       description: No task found
3895     '401':
3896       description: Not authorized
3897     '404':
3898       description: The named task could not be found
3899   put:
3900     tags:
3901     - Task
3902     summary: Uploads a task to the list of tasks
3903     description: Uploads a task to the list of tasks
3904     operationId: cloudmesh.task.add
3905     parameters:
3906     - name: name
3907       in: path
3908       required: true
3909       schema:
3910         type: string
3911       description: The name of the task
3912     requestBody:
3913       description: The task to be uploaded
3914       required: true
3915       content:
3916         application/json:
3917           schema:
3918             $ref: '#/components/schemas/Task'
3919     responses:
3920     '200':
3921       description: Task updated
3922     '400':
3923       description: Error updating task.
3924     '401':
3925       description: Not authorized
3926   delete:
3927     tags:
3928     - Task
3929     summary: Deletes the named task
3930     description: Deletes an task by name
3931     operationId: cloudmesh.task.delete_by_name
3932     parameters:
3933     - name: name
3934       in: path
3935       required: true
3936       schema:
3937         type: string
3938       description: The name of the task
3939     responses:
3940     '200':
3941       description: Deletion successful
3942     '400':
3943       description: No task found
3944     '401':
3945       description: Not authorized
3946     '404':
3947       description: The named task could not be found
3948 /policy:
3949   get:
3950     tags:
3951     - Task
3952     summary: Returns the policy
3953     description: Returns the policy
3954     operationId: cloudmesh.task.policy.list
3955     responses:
3956     '200':

```

```

3957     description: The policy
3958     content:
3959       application/json:
3960         schema:
3961           type: array
3962           items:
3963             $ref: '#/components/schemas/Policy'
3964     '400':
3965       description: No tasks found
3966     '401':
3967       description: Not authorized
3968   put:
3969     tags:
3970     - Task
3971     summary: Uploads the policy
3972     description: Uploads a task to the list of tasks
3973     operationId: cloudmesh.task.policy.add
3974     requestBody:
3975       description: The Policy
3976       required: true
3977       content:
3978         application/json:
3979           schema:
3980             $ref: '#/components/schemas/Policy'
3981     responses:
3982       '200':
3983         description: Task updated
3984       '400':
3985         description: Error updating task
3986       '401':
3987         description: Not authorized
3988   components:
3989     schemas:
3990     Task:
3991       type: object
3992       description: The scheduler
3993       properties:
3994         name:
3995           type: string
3996           description: Name of the scheduler
3997         user:
3998           type: string
3999           description: The username the task belongs to
4000       description:
4001         type: string
4002         description: The description of the task
4003       kind:
4004         type: string
4005         description: The kind of the task
4006       content:
4007         type: string
4008         description: The content of the task
4009     Policy:
4010       type: object
4011       description: The policy of the scheduler
4012       properties:
4013         name:
4014           type: string
4015           description: Name of the scheduler policy
4016       description:
4017         type: string
4018         description: The description of the policy
4019       kind:
4020         type: string
4021         description: The kind of the policy
4022       parameters:
4023         type: string
4024         description: parameters to define the behavior of the scheduler

```

4025 **4.9 COMPUTE MANAGEMENT: VIRTUAL MACHINES**

4026 This section summarizes a basic interface specification of virtual machines.

4027 **4.9.1 IMAGE**

4028 To execute virtual machines, an image is needed that specifies the details of the operating system.

4029 **4.9.1.1 Schema Image**

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	Requirements	Minimum requirement to run the image

4030 **4.9.1.2 Schema Requirements**

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

4031 **4.9.1.3 Paths**

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

4032 **4.9.1.3.1 /image/{cloud}**

4033 **4.9.1.3.1.1 GET /image/{cloud}**

4034 Returns a list of all images

4035 Responses

Code	Description	Schema
200	The list of images	Array[Image]
401	Not authorized	String

4036 Parameters

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

4037 **4.9.1.3.2 /image/{cloud}/{name}**

4038 **4.9.1.3.2.1 GET /image/{cloud}/{name}**

4039 Returns an image by name

4040 Responses

Code	Description	Schema
200	Returning the information of the image	Image
401	Not authorized	String
404	The named image could not be found	String

4041 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

4042 **4.9.1.3.2.2 PUT /image/{cloud}/{name}**

4043 Sets the named image

4044 Responses

Code	Description	Schema
200	Image updated or created	String
401	Not authorized	String
404	The named image could not be found	String

4045 Parameters

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

4046 Request Body

Located in	Description	Required	Schema
Body	The image to add or modify	True	Image

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

4047 **4.9.1.3.2.3 DELETE /image/{cloud}/{name}**

4048 Deletes an image by name

4049 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named image could not be found	String

4050 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

4051 **4.9.1.4 image.yaml**

```

4052 openapi: "3.0.2"
4053 info:
4054   version: 3.2.0
4055   x-date: 17-06-2019
4056   x-status: defined
4057   title: Image
4058   description: |-
4059
4060     To execute virtual machines, we need an image that specifies the
4061     details of the operating system.
4062
4063   termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
4064   contact:
4065     name: NIST BDRA Interface Subgroup
4066     url: https://cloudmesh-community.github.io/nist/spec/
4067   license:
4068     name: Apache 2.0
4069     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
4070 servers:
4071   - url: /cloudmesh/v3
4072 paths:
4073   /image/{cloud}:
4074     get:
4075       tags:
4076         - Image
4077       summary: Returns a list of images for the cloud
4078       description: Returns a list of all images
4079       operationId: cloudmesh.image.list
4080       parameters:
4081         - name: cloud
4082           in: path
4083           required: true
4084           schema:
4085             type: string
4086           description: The name of the cloud
4087       responses:
4088         '200':
4089           description: The list of images
4090           content:
4091             application/json:
4092               schema:
4093                 type: array
4094                 items:
4095                   $ref: '#/components/schemas/Image'

```

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

```

4096     '401':
4097         description: Not authorized
4098 /image/{cloud}/{name}:
4099     get:
4100         tags:
4101             - Image
4102         summary: Returns the named image
4103         description: Returns a image by name
4104         operationId: cloudmesh.image.find_by_name
4105         parameters:
4106             - name: cloud
4107               in: path
4108               description: The name of the cloud
4109               required: true
4110               schema:
4111                 type: string
4112             - name: name
4113               in: path
4114               required: true
4115               schema:
4116                 type: string
4117               description: The name of the image
4118         responses:
4119             '200':
4120                 description: Returning the information of the image
4121                 content:
4122                     application/json:
4123                         schema:
4124                             $ref: '#/components/schemas/Image'
4125             '401':
4126                 description: Not authorized
4127             '404':
4128                 description: The named image could not be found
4129     put:
4130         tags:
4131             - Image
4132         summary: Add a image
4133         description: Sets the named image
4134         operationId: cloudmesh.image.add
4135         parameters:
4136             - name: cloud
4137               in: path
4138               required: true
4139               schema:
4140                 type: string
4141               description: The name of the cloud
4142         requestBody:
4143             description: The image to add or modify
4144             required: true
4145             content:
4146                 application/json:
4147                     schema:
4148                         $ref: '#/components/schemas/Image'
4149         responses:
4150             '200':
4151                 description: Image updated or created
4152             '401':
4153                 description: Not authorized
4154             '404':
4155                 description: The named image could not be found
4156     delete:
4157         tags:
4158             - Image
4159         summary: Deletes the named image
4160         description: Deletes a image by name
4161         operationId: cloudmesh.image.delete_by_name
4162         parameters:
4163             - name: cloud
4164               description: The name of the cloud

```

```

4165         in: path
4166         required: true
4167         schema:
4168           type: string
4169     - name: name
4170       in: path
4171       required: true
4172       schema:
4173         type: string
4174       description: The name of the image
4175     responses:
4176       '200':
4177         description: Deletion successful
4178       '401':
4179         description: Not authorized
4180       '404':
4181         description: The named image could not be found
4182 components:
4183   schemas:
4184     Image:
4185       type: object
4186       properties:
4187         name:
4188           type: string
4189           description: A unique name of the image
4190         cloud:
4191           type: string
4192           description: The name of the cloud
4193         label:
4194           type: string
4195           description: A label that can be defined by the user for the image
4196         description:
4197           type: string
4198           description: A description for the image
4199         osType:
4200           type: string
4201           description: The OS type of the image
4202         osVersion:
4203           type: string
4204           description: The OS version of the image
4205         status:
4206           type: string
4207           description: The status of the image
4208         progress:
4209           type: integer
4210           description: The loading progress percentage of the image
4211         visibility:
4212           type: string
4213           description: The visibility of the image
4214         requirement:
4215           $ref: "#/components/schemas/Requirements"
4216           description: Minimum requirement to run the image
4217     Requirements:
4218       type: object
4219       properties:
4220         size:
4221           type: integer
4222           description: Minimum disk size in bytes required for the image
4223         ram:
4224           type: integer
4225           description: Minimum ram size in bytes to run the image
4226         cpu:
4227           type: string
4228           description: CPU required to run the image
4229         cores:
4230           type: integer
4231           description: Minimum number of cores

```

4232 **4.9.2 FLAVOR**

4233 The flavor specifies elementary information about a virtual machine or compute node. This information
 4234 includes name, ID, label, ram size, swap size, disk space, availability of ephemeral disk, available
 4235 bandwidth, price value, and cloud name. Flavors and the corresponding information are essential to size a
 4236 virtual cluster appropriately.

4237 **4.9.2.1 Schema Flavor**

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

4238 **4.9.2.2 Paths**

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

4239 **4.9.2.2.1 /flavor/{cloud}**

4240 **4.9.2.2.1.1 GET /flavor/{cloud}**

4241 Returns a list of all flavors

4242 Responses

Code	Description	Schema
200	The list of flavors	Array[Flavor]
401	Not authorized	String

4243 Parameters

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

4244 **4.9.2.2.2 /flavor/{cloud}/{name}**

4245 **4.9.2.2.2.1 GET /flavor/{cloud}/{name}**

4246 Returns a flavor by name

4247 Responses

Code	Description	Schema
200	Returning the information of the flavor	Flavor
401	Not authorized	String
404	The named flavor could not be found	String

4248 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

4249 **4.9.2.2.2.2 PUT /flavor/{cloud}/{name}**

4250 Sets the named flavor

4251 Responses

Code	Description	Schema
200	Flavor updated	String
401	Not authorized	String
404	The named flavor could not be found	String

4252 Parameters

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

4253 Request Body

Located in	Description	Required	Schema
Body	The flavor to add or modify	True	Flavor

4254 **4.9.2.2.2.3 DELETE /flavor/{cloud}/{name}**

4255 Deletes a flavor by name

4256 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named flavor could not be found	String

4257 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

4258 **4.9.2.3 flavor.yaml**

```

4259 openapi: "3.0.2"
4260 info:
4261   version: 3.2.0
4262   x-date: 17-06-2019
4263   x-status: defined
4264   title: Flavor
4265   description: |-
4266
4267     The flavor specifies elementary information about a virtual machine
4268     or compute node. This information includes name, id, label, ram size,
4269     swap size, disk space, availability of ephemeral disk, available
4270     bandwidth, price value, cloud name. Flavors and the corresponding
4271     information are essential to size a
4272     virtual cluster appropriately.
4273
4274   termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
4275   contact:
4276     name: NIST BDRA Interface Subgroup
4277     url: https://cloudmesh-community.github.io/nist
4278   license:
4279     name: Apache 2.0
4280     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
4281 servers:
4282   - url: /cloudmesh/v3
4283 paths:
4284   /flavor/{cloud}:
4285     get:
4286       tags:
4287         - Flavor
4288       summary: Returns a list of flavors for the cloud
4289       description: Returns a list of all flavors
4290       operationId: cloudmesh.flavor.list
4291       parameters:
4292         - name: cloud
4293           in: path
4294           required: true
4295           schema:
4296             type: string
4297             description: The name of the cloud
4298       responses:
4299         '200':
4300           description: The list of flavors
4301           content:
4302             application/json:
4303               schema:
4304                 type: array
4305                 items:
4306                   $ref: '#/components/schemas/Flavor'
4307         '401':
4308           description: Not authorized
4309   /flavor/{cloud}/{name}:
4310     get:
4311       tags:
4312         - Flavor
4313       summary: Returns the named flavor
4314       description: Returns a flavor by name
4315       operationId: cloudmesh.flavor.find_by_name
4316       parameters:

```

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

4317     - name: cloud
4318       in: path
4319       description: The name of the cloud
4320       required: true
4321       schema:
4322         type: string
4323     - name: name
4324       in: path
4325       required: true
4326       schema:
4327         type: string
4328       description: The name of the flavor
4329     responses:
4330       '200':
4331         description: Returning the information of the flavor
4332         content:
4333           application/json:
4334             schema:
4335               $ref: '#/components/schemas/Flavor'
4336       '401':
4337         description: Not authorized
4338       '404':
4339         description: The named flavor could not be found
4340     put:
4341       tags:
4342         - Flavor
4343       summary: Add a flavor
4344       description: Sets the named flavor
4345       operationId: cloudmesh.flavor.add
4346       parameters:
4347         - name: cloud
4348           in: path
4349           required: true
4350           schema:
4351             type: string
4352             description: The name of the cloud
4353       requestBody:
4354         description: The flavor to add or modify
4355         required: true
4356         content:
4357           application/json:
4358             schema:
4359               $ref: '#/components/schemas/Flavor'
4360     responses:
4361       '200':
4362         description: Flavor updated
4363       '401':
4364         description: Not authorized
4365       '404':
4366         description: The named flavor could not be found
4367     delete:
4368       tags:
4369         - Flavor
4370       summary: Deletes the named flavor
4371       description: Deletes a flavor by name
4372       operationId: cloudmesh.flavor.delete_by_name
4373       parameters:
4374         - name: cloud
4375           description: The name of the cloud
4376           in: path
4377           required: true
4378           schema:
4379             type: string
4380         - name: name
4381           in: path
4382           required: true
4383           schema:
4384             type: string
4385           description: The name of the flavor

```

```

4386     responses:
4387         '200':
4388             description: Deletion successful
4389         '401':
4390             description: Not authorized
4391         '404':
4392             description: The named flavor could not be found
4393 components:
4394     schemas:
4395         Flavor:
4396             type: object
4397             description: The flavor
4398             properties:
4399                 name:
4400                     type: string
4401                     description: Name of the flavor
4402                 id:
4403                     type: string
4404                     description: The id of the flavor for the named cloud
4405                 label:
4406                     type: string
4407                     description: A label that a user can set for this flavor
4408             description:
4409                 type: string
4410                 description: A description for the flavor
4411             ram:
4412                 type: integer
4413                 description: Number of bytes used for the image in RAM
4414             swap:
4415                 type: integer
4416                 description: Number of bytes used for the image in SWAP
4417             disk:
4418                 type: integer
4419                 description: Number of bytes used for the disk
4420             ephemeral_disk:
4421                 type: boolean
4422                 description: Specifies whether the flavor features an ephemeral disk
4423             bandwidth:
4424                 type: integer
4425                 description: Bandwidth of the node
4426             price:
4427                 type: number
4428                 description: Price for the flavor
4429             cloud:
4430                 type: string
4431                 description: Name of the cloud this flavor is used in

```

4432 **4.9.3 VIRTUAL MACHINE**

4433 VM is used to manage virtual machines.

4434 **4.9.3.1 Schema Vm**

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

4435 **4.9.3.2 Paths**

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

4436 **4.9.3.2.1 /vm/{cloud}**

4437 **4.9.3.2.1.1 GET /vm/{cloud}**

4438 Returns a list of all virtual machines

4439 Responses

Code	Description	Schema
200	The list of virtual machines	array[Vm]
400	No Vm found	String
401	Not authorized	String

4440 Parameters

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

4441 **4.9.3.2.2 /vm/{cloud}/{name}**

4442 **4.9.3.2.2.1 GET /vm/{cloud}/{name}**

4443 Returns a virtual machine by name

4444 Responses

Code	Description	Schema
200	Returning the information of the virtual machine	Vm
400	Error updating virtual machine	String
401	Not authorized	String
404	The named virtual machine or cloud could not be found	String

4445

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

4446 **4.9.3.2.2.2 PUT /vm/{cloud}/{name}**

4447 Sets the named virtual machine

4448 Responses

Code	Description	Schema
200	Vm updated	String
400	Error updating virtual machine	String
401	Not authorized	String
404	The named virtual machine or cloud could not be found	String

4449 Parameters

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

4450 Request Body

Located in	Description	Required	Schema
Body	The virtual machine to add or modify	True	Vm

4451 **4.9.3.2.2.3 DELETE /vm/{cloud}/{name}**

4452 Deletes a virtual machine by name

4453 Responses

Code	Description	Schema
200	Deletion successful	String
400	Error updating virtual machine	String
401	Not authorized	String
404	The named virtual machine or cloud could not be found	String

4454 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

4455 **4.9.3.3 vm.yaml**

```

4456 openapi: "3.0.2"
4457 info:
4458   version: 3.2.0
4459   x-date: 17-06-2019
4460   x-status: defined
4461   title: Virtual Machine
4462   description: |-
4463
4464     Vm is used to manage virtual machines.
4465
4466   termsOfService: https://github.com/cloudmesh-community/nist/blob/master/LICENSE.txt
4467   contact:
4468     name: NIST BDRA Interface Subgroup Service
4469     url: https://cloudmesh-community.github.io/nist/spec/
4470   license:
4471     name: Apache 2.0
4472     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
4473   servers:
4474     - url: /cloudmesh/v3
4475   paths:
4476     /vm/{cloud}:
4477       get:
4478         tags:
4479           - Vm
4480         summary: Returns a list of virtual machines for the cloud
4481         description: Returns a list of all virtual machines
4482         operationId: cloudmesh.vm.list
4483         parameters:
4484           - name: cloud
4485             in: path
4486             required: true
4487             schema:
4488               type: string
4489             description: The name of the cloud
4490         responses:
4491           '200':
4492             description: The list of virtual machines
4493             content:
4494               application/json:
4495                 schema:
4496                   type: array
4497                   items:
4498                     $ref: '#/components/schemas/Vm'
4499           '400':
4500             description: No Vm found
4501           '401':
4502             description: Not authorized
4503     /vm/{cloud}/{name}:
4504       get:
4505         tags:
4506           - Vm
4507         summary: Returns the named virtual machine
4508         description: Returns a virtual machine by name
4509         operationId: cloudmesh.vm.find_by_name
4510         parameters:
4511           - name: cloud
4512             in: path
4513             description: The name of the cloud

```

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

4514         required: true
4515         schema:
4516           type: string
4517     - name: name
4518       in: path
4519       description: The name of the virtual machine
4520       required: true
4521       schema:
4522         type: string
4523     responses:
4524       '200':
4525         description: Returning the information of the virtual machine
4526         content:
4527           application/json:
4528             schema:
4529               $ref: '#/components/schemas/Vm'
4530       '400':
4531         description: Error updating virtual machine
4532       '401':
4533         description: Not authorized
4534       '404':
4535         description: The named virtual machine or cloud could not be found
4536     put:
4537       tags:
4538         - Vm
4539       summary: Add a virtual machine
4540       description: Sets the named virtual machine
4541       operationId: cloudmesh.vm.add
4542       parameters:
4543         - name: cloud
4544           in: path
4545           required: true
4546           schema:
4547             type: string
4548           description: The name of the cloud
4549       requestBody:
4550         description: The virtual machine to add or modify
4551         required: true
4552         content:
4553           application/json:
4554             schema:
4555               $ref: '#/components/schemas/Vm'
4556       responses:
4557         '200':
4558           description: Vm updated
4559         '400':
4560           description: Error updating virtual machine
4561         '401':
4562           description: Not authorized
4563         '404':
4564           description: The named virtual machine or cloud could not be found
4565     delete:
4566       tags:
4567         - Vm
4568       summary: Deletes the named virtual machine
4569       description: Deletes a virtual machine by name
4570       operationId: cloudmesh.vm.delete_by_name
4571       parameters:
4572         - name: cloud
4573           description: The name of the cloud
4574           in: path
4575           required: true
4576           schema:
4577             type: string
4578         - name: name
4579           in: path
4580           description: The name of the virtual machine
4581           required: true
4582           schema:

```

```

4583         type: string
4584     responses:
4585         '200':
4586             description: Deletion successful
4587         '400':
4588             description: Error updating virtual machine
4589         '401':
4590             description: Not authorized
4591         '404':
4592             description: The named virtual machine or cloud could not be found
4593 components:
4594     schemas:
4595         Vm:
4596             type: object
4597             properties:
4598                 provider:
4599                     type: string
4600                     description: Name of the provider
4601                 name:
4602                     type: string
4603                     description: the unique name of the virtual machine
4604                 image:
4605                     type: string
4606                     description: the image name for the virtual machine
4607                 flavor:
4608                     type: string
4609                     description: the flavor name for the virtual machine
4610                 region:
4611                     type: string
4612                     description: an optional region
4613                 state:
4614                     type: string
4615                     description: The state of the virtual machine
4616                 private_ips:
4617                     type: string
4618                     description: The private IPs
4619                 public_ips:
4620                     type: string
4621                     description: The public IPS
4622                 metadata:
4623                     type: string
4624                     description: The meta data passed along to the virtual machine
4625

```

4626 **4.9.4 SECGROUP**

4627 A security group defines the incoming and outgoing security rules which can then be assigned to a node.
 4628 The connection to and from the node will be determined by the security group rules, in addition to any
 4629 other possible rules applied on network devices or from the instance’s firewall settings. A security group
 4630 may have one or multiple rules and a node may be associated with one or more security groups.

4631 **4.9.4.1 Schema Secgroup**

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

4632 **4.9.4.2 Schema SecGroupRule**

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 Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP)
from_port	integer	Port range starting port
to_port	integer	Port range ending port
cidr	String	The source or destination network in Classless Inter-Domain Routing (CIDR) notation

4633 **4.9.4.3 Paths**

HTTP	Path	Summary
get	/secgroup	Returns all security groups
get	/secgroup/{name}	Return the security group by name
post	/secgroup/{name}	Create the named security group
get	/secgroup/{name}/rule/{rule}	Get an existing rule from the specified security group
put	/secgroup/{name}/rule/{rule}	Create or update specified security group
delete	/secgroup/{name}/rule/{rule}	Delete an existing rule from the specified security group

4634 **4.9.4.3.1 /secgroup**

4635 **4.9.4.3.1.1 GET /secgroup**

4636 Returns all security groups

4637 Responses

Code	Description	Schema
200	security group information	array[Secgroup]
401	Not authorized	String

4638 **4.9.4.3.2 /secgroup/{name}**

4639 **4.9.4.3.2.1 GET /secgroup/{name}**

4640 Return the security group by name

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

4641 Responses

Code	Description	Schema
200	Security group information	Secgroup
401	Not authorized	String
404	The named security group could not be found	String

4642 Parameters

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

4643 **4.9.4.3.2.2 POST /secgroup/{name}**

4644 Create a new named security group

4645 Responses

Code	Description	Schema
201	Created	String
400	The group could not be created	String
401	Not authorized	String

4646 Parameters

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

4647 **4.9.4.3.3 /secgroup/{name}/rule/{rule}**

4648 **4.9.4.3.3.1 GET /secgroup/{name}/rule/{rule}**

4649 Create a new rule in security group

4650 Responses

Code	Description	Schema
200	The security group rule definition info	Secgrouprule
401	Not authorized	String
404	The named security group or role could not be found	String

4651 Parameters

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

4652 **4.9.4.3.3.2 PUT /secgroup/{name}/rule/{rule}**

4653 Create a new rule in security group

4654 Responses

Code	Description	Schema
200	Created	String
401	Not authorized	String
404	The named security group or role could not be found	String

4655 Parameters

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

4656 Request Body

Located in	Description	Required	Schema
Body	The new security group rule to create	True	SecGroupRule

4657 **4.9.4.3.3.3 DELETE /secgroup/{name}/rule/{rule}**

4658 Create a new rule in security group

4659 Responses

Code	Description	Schema
200	Deleted successfully	String
401	Not authorized	String
404	The named security group or role could not be found	String

4660 Parameters

Name	Located in	Description	Required	Schema
name	path	The named secgroup	True	String
rule	path	The secgroup rule	True	String

4661 **4.9.4.4 secgroup.yaml**

4662 openapi: "3.0.2"

4663 info:

4664 version: 3.2.0
 4665 x-date: 17-06-2019
 4666 x-status: defined
 4667 title: Secgroup
 4668 description: |-
 4669

4670 A security group defines the incoming and outgoing security rules
 4671 which can then be assigned to a node. The connection to and from the node
 4672 will be determined by the security group rules, in addition to any other
 4673 possible rules applied on network devices or from the instance's firewall
 4674 settings. A security group may have one or multiple rules and a node may be
 4675 associated with one or more security groups.
 4676

4677 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
 4678 contact:

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1


```

4679     name: NIST BDRA Interface Subgroup
4680     url: https://cloudmesh-community.github.io/nist
4681   license:
4682     name: Apache 2.0
4683     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
4684   servers:
4685     - url: /cloudmesh/v3
4686   paths:
4687     /secgroup:
4688       get:
4689         tags:
4690         - Security group
4691         summary: Returns all security groups
4692         description: Returns all security groups
4693         operationId: cloudmesh.secgroup.get
4694         responses:
4695           '200':
4696             description: security group information
4697             content:
4698               application/json:
4699                 schema:
4700                   type: array
4701                   items:
4702                     $ref: "#/components/schemas/Secgroup"
4703           '401':
4704             description: Not authorized
4705     /secgroup/{name}:
4706       get:
4707         tags:
4708         - Security group
4709         summary: Return the security group by name
4710         description: Return the security group by name
4711         operationId: cloudmesh.secgroup.get_by_name
4712         parameters:
4713         - name: name
4714           description: name of the security group
4715           in: path
4716           required: true
4717           schema:
4718             type: string
4719         responses:
4720           '200':
4721             description: security group information
4722             content:
4723               application/json:
4724                 schema:
4725                   $ref: "#/components/schemas/Secgroup"
4726           '401':
4727             description: Not authorized
4728           '404':
4729             description: The named security group could not be found
4730       post:
4731         tags:
4732         - Security group
4733         summary: Create the named security group
4734         description: Create a new named security group
4735         operationId: cloudmesh.secgroup.create
4736         parameters:
4737         - in: path
4738           name: name
4739           required: true
4740           description: The name of the security group to create
4741           schema:
4742             type: string
4743         responses:
4744           '201':
4745             description: Created
4746           '400':
4747             description: The group could not be created

```

```

4748     '401':
4749         description: Not authorized
4750 /secgroup/{name}/rule/{rule}:
4751 get:
4752     tags:
4753     - Security group
4754     summary: Get an existing rule from the specified security group
4755     description: Create a new rule in security group
4756     operationId: cloudmesh.secgroup.get_rule
4757     parameters:
4758     - in: path
4759       name: name
4760       required: true
4761       description: The name of the security group from which the rule will be deleted
4762       schema:
4763         type: string
4764     - in: path
4765       name: rule
4766       required: true
4767       description: The rule to be added
4768       schema:
4769         type: string
4770     responses:
4771     '200':
4772       description: The security group rule definition info
4773       content:
4774         application/json:
4775           schema:
4776             $ref: "#/components/schemas/SecGroupRule"
4777     '401':
4778       description: Not authorized
4779     '404':
4780       description: The named security group or role could not be found
4781 put:
4782     tags:
4783     - Security group
4784     summary: Create or update specified security group
4785     description: Create a new rule in security group
4786     operationId: cloudmesh.secgroup.add_rule
4787     parameters:
4788     - in: path
4789       name: name
4790       required: true
4791       description: The name of the new security group to create
4792       schema:
4793         type: string
4794     requestBody:
4795       description: The new security group rule to create
4796       required: true
4797       content:
4798         application/json:
4799           schema:
4800             $ref: '#/components/schemas/SecGroupRule'
4801     responses:
4802     '200':
4803       description: Created
4804     '401':
4805       description: Not authorized
4806     '404':
4807       description: The named security group or role could not be found
4808 delete:
4809     tags:
4810     - Security group
4811     summary: Delete an existing rule from the specified security group
4812     description: Create a new rule in security group
4813     operationId: cloudmesh.secgroup.delete_rule
4814     parameters:
4815     - in: path
4816       name: name

```

```

4817         required: true
4818         description: The named secgroup
4819         schema:
4820           type: string
4821         - in: path
4822           name: rule
4823           required: true
4824           description: The secgroup rule
4825           schema:
4826             type: string
4827         responses:
4828           '200':
4829             description: Deleted successfully
4830           '401':
4831             description: Not authorized
4832           '404':
4833             description: The named security group or role could not be found
4834 components:
4835   schemas:
4836     Secgroup:
4837       type: object
4838       description: the security group object
4839       properties:
4840         name:
4841           type: string
4842           description: Name of the security group
4843         description:
4844           type: string
4845           description: Describes what the security group is for
4846         rules:
4847           type: array
4848           description: List of Security group rules
4849           items:
4850             $ref: "#/components/schemas/SecGroupRule"
4851     SecGroupRule:
4852       type: object
4853       description: security group rule
4854       properties:
4855         name:
4856           type: string
4857           description: Unique name of the rule
4858         ingress:
4859           type: boolean
4860           description: The defined security group rule is for ingress if True
4861         egress:
4862           type: boolean
4863           description: The defined security group rule is for egress if True
4864         remote_group:
4865           type: string
4866           description: Name of the group if the rule is defined by group
4867                       instead of IP range
4868         protocol:
4869           type: string
4870           description: The protocol used such as TCP, UDP, ICMP
4871           example: TCP
4872         from_port:
4873           type: integer
4874           description: Port range starting port
4875         to_port:
4876           type: integer
4877           description: Port range ending port
4878         cidr:
4879           type: string
4880           description: The source or destination network in CIDR notation,
4881           example: 129.79.0.0/16

```

4882 **4.9.5 Nic**

4883 A resource store Network Interface Controller (NIC) information.

4884 **4.9.5.1 Schema Nic**

Property	Type	Description
name	string	Name of the network interface controller
kind	string	Kind of the network interface controller (e.g., Wi-Fi, 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 maximum transmission unit (MTU) of the NIC
bandwidth	integer	The bandwidth in bps

4885 **4.9.5.2 Paths**

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

4886 **4.9.5.2.1 /nic**

4887 **4.9.5.2.1.1 GET /nic**

4888 Returns a list of all network interface controllers

4889 Responses

Code	Description	Schema
200	The list of network interface controllers	array[Nic]
401	Not authorized	String

4890 **4.9.5.2.2 /nic/{name}**

4891 **4.9.5.2.2.1 GET /nic/{name}**

4892 Returns a network interface controller by name

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

4893 Responses

Code	Description	Schema
200	Returning the information of the network interface controller	Nic
401	Not authorized	String
404	The named network interface controller could not be found	String

4894 Parameters

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

4895 **4.9.5.2.2.2 PUT /nic/{name}**

4896 Sets the named network interface controller

4897 Responses

Code	Description	Schema
200	NIC updated	String
401	Not authorized	String

4898 Request Body

Located in	Description	Required	Schema
Body	The new NIC to create or update	True	Nic

4899 **4.9.5.2.2.3 DELETE /nic/{name}**

4900 Deletes a network interface controller by name

4901 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named network interface controller could not be found	String

4902 Parameters

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

4903 **4.9.5.3 nic.yaml**

4904 openapi: "3.0.2"

4905 info:

4906 version: 3.2.0

4907 x-date: 17-06-2019

4908 x-status: defined

4909 title: Nic

4910 description: |-

4911 A resource store Network Interface Controller (NIC) information.

4912

4913

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

```

4914 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
4915 contact:
4916   name: NIST BDRA Interface Subgroup
4917   url: https://cloudmesh-community.github.io/nist
4918 license:
4919   name: Apache 2.0
4920   url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
4921 servers:
4922   - url: /cloudmesh/v3
4923 paths:
4924   /nic:
4925     get:
4926       tags:
4927         - Nic
4928       summary: Returns a list of network interface controllers
4929       description: Returns a list of all network interface controllers
4930       operationId: cloudmesh.nic.list
4931       responses:
4932         '200':
4933           description: The list of network interface controllers
4934           content:
4935             application/json:
4936               schema:
4937                 type: array
4938                 items:
4939                   $ref: '#/components/schemas/Nic'
4940         '401':
4941           description: Not authorized
4942   /nic/{name}:
4943     get:
4944       tags:
4945         - Nic
4946       summary: Returns the named network interface controller
4947       description: Returns a network interface controller by name
4948       operationId: cloudmesh.nic.find_by_name
4949       parameters:
4950         - name: name
4951           in: path
4952           required: true
4953           schema:
4954             type: string
4955           description: The name of the network interface controller
4956       responses:
4957         '200':
4958           description: Returning the information of the network interface controller
4959           content:
4960             application/json:
4961               schema:
4962                 $ref: '#/components/schemas/Nic'
4963         '401':
4964           description: Not authorized
4965         '404':
4966           description: The named network interface controller could not be found
4967     put:
4968       tags:
4969         - Nic
4970       summary: Set a network interface controller
4971       description: Sets the named network interface controller
4972       operationId: cloudmesh.nic.add
4973       requestBody:
4974         description: The new nic to create or update
4975         required: true
4976         content:
4977           application/json:
4978             schema:
4979               $ref: '#/components/schemas/Nic'
4980       responses:
4981         '200':
4982           description: Nic updated

```

```

4983         '401':
4984             description: Not authorized
4985     delete:
4986         tags:
4987             - Nic
4988         summary: Deletes the named network interface controller
4989         description: Deletes a network interface controller by name
4990         operationId: cloudmesh.nic.delete_by_name
4991         parameters:
4992             - name: name
4993               in: path
4994               required: true
4995               schema:
4996                 type: string
4997               description: The name of the network interface controller
4998         responses:
4999             '200':
5000                 description: Deletion successful
5001             '401':
5002                 description: Not authorized
5003             '404':
5004                 description: The named network interface controller could not be found
5005     components:
5006         schemas:
5007             Nic:
5008                 type: object
5009                 description: The network interface controller
5010                 properties:
5011                     name:
5012                         type: string
5013                         description: Name of the network interface controller
5014                     kind:
5015                         type: string
5016                         description: Kind of the network interface controller (wifi, WAN, ...)
5017                     mac:
5018                         type: string
5019                         description: The mac address
5020                     ip:
5021                         type: string
5022                         description: The IP address
5023                     mask:
5024                         type: string
5025                         description: The network mask
5026                     broadcast:
5027                         type: string
5028                         description: The broadcast address
5029                     gateway:
5030                         type: string
5031                         description: The gateway address
5032                     mtu:
5033                         type: integer
5034                         description: The MTU of the NIC
5035                     bandwidth:
5036                         type: integer
5037                         description: The bandwidth in bps

```

4.10 COMPUTE MANAGEMENT: CONTAINERS

4.10.1 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.

5043 **4.10.1.1 Schema Container**

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

5044 **4.10.1.2 Paths**

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

5045 **4.10.1.2.1 /container**

5046 **4.10.1.2.1.1 GET /container**

5047 Returns a list of all containers

5048 Responses

Code	Description	Schema
200	The list of containers	array[Container]
401	Not authorized	String

5049 **4.10.1.2.2 /container/{name}**

5050 **4.10.1.2.2.1 GET /container/{name}**

5051 Returns a container by name

5052 Responses

Code	Description	Schema
200	Returning the information of the container	Container
400	No Container found	String
401	Not authorized	String
404	The named container could not be found	String

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

5053 Parameters

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

5054 **4.10.1.2.2.2 PUT /container/{name}**

5055 Sets the named container

5056 Responses

Code	Description	Schema
200	Container updated	String
401	Not authorized	String
400	Error updating container	String

5057 Request Body

Located in	Description	Required	Schema
Body	The new container to create	True	Container

5058 **4.10.1.2.2.3 DELETE /container/{name}**

5059 Deletes a container by name

5060 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named container could not be found	String

5061 Parameters

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

5062 **4.10.1.3 containers.yaml**

5063 openapi: "3.0.2"

5064 info:

5065 version: 3.2.0

5066 x-date: 17-06-2019

5067 x-status: defined

5068 title: Containers

5069 description: |-

5070
5071 Numerous different containers are likely to be created and handling them
5072 becomes more and more time consuming as their number increases. This service
5073 helps to solve that issue by storing containers and their corresponding
5074 information.

5075
5076 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"

5077 contact:

5078 name: NIST BDRA Interface Subgroup

5079 url: https://cloudmesh-community.github.io/nist

5080 license:

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

5081     name: Apache 2.0
5082     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
5083 servers:
5084   - url: /cloudmesh/v3
5085 paths:
5086   /container:
5087     get:
5088       tags:
5089         - Container
5090       summary: Returns a list of containers
5091       description: Returns a list of all containers
5092       operationId: cloudmesh.container.list
5093       responses:
5094         '200':
5095           description: The list of containerses
5096           content:
5097             application/json:
5098               schema:
5099                 type: array
5100                 items:
5101                   $ref: '#/components/schemas/Container'
5102         '401':
5103           description: Not authorized
5104   /container/{name}:
5105     get:
5106       tags:
5107         - Container
5108       summary: Returns the named container
5109       description: Returns an container by name
5110       operationId: cloudmesh.container.find_by_name
5111       parameters:
5112         - name: name
5113           in: path
5114           required: true
5115           schema:
5116             type: string
5117           description: The name of the container
5118       responses:
5119         '200':
5120           description: Returning the information of the container
5121           content:
5122             application/json:
5123               schema:
5124                 $ref: '#/components/schemas/Container'
5125         '400':
5126           description: No Container found
5127         '401':
5128           description: Not authorized
5129         '404':
5130           description: The named container could not be found
5131     put:
5132       tags:
5133         - Container
5134       summary: Set an container
5135       description: Sets the named container
5136       operationId: cloudmesh.container.add
5137       requestBody:
5138         description: The new container to create
5139         required: true
5140         content:
5141           application/json:
5142             schema:
5143               $ref: '#/components/schemas/Container'
5144       responses:
5145         '200':
5146           description: Container updated
5147         '401':
5148           description: Not authorized
5149         '400':

```

```

5150         description: Error updating container
5151 delete:
5152   tags:
5153     - Container
5154   summary: Deletes the named container
5155   description: Deletes an container by name
5156   operationId: cloudmesh.container.delete_by_name
5157   parameters:
5158     - name: name
5159       in: path
5160       required: true
5161       schema:
5162         type: string
5163         description: The name of the container
5164   responses:
5165     '200':
5166       description: Deletion successful
5167     '401':
5168       description: Not authorized
5169     '404':
5170       description: The named container could not be found
5171 components:
5172   schemas:
5173     Container:
5174       type: object
5175       description: A record representing a container
5176       properties:
5177         name:
5178           type: string
5179           description: Name of the container
5180         version:
5181           type: string
5182           description: Version of the container
5183         label:
5184           type: string
5185           description: Label of the container
5186         type:
5187           type: string
5188           description: Type of the container
5189         definition:
5190           type: string
5191           description: Definition or manifest of the container
5192         imgURI:
5193           type: string
5194           description: URI of the container
5195         tags:
5196           type: array
5197           description: Tags of the container
5198         items:
5199           type: string

```

5200 **4.11 COMPUTE MANAGEMENT: MAPREDUCE**

5201 **4.11.1 *MAPREDUCE***

5202 A service to store the information of a MapReduce deployment definition. All of the attributes are stored
 5203 as Strings.

5204 **4.11.1.1 Schema Map**

Property	Type	Description
name	string	The name of the map function
kind	string	The kind in which the specification is provided
content	string	The kind in which the specification is provided

5205 **4.11.1.2 Schema Reduce**

Property	Type	Description
name	string	The name of the reduce function
kind	string	The kind in which the specification is provided
content	string	The kind in which the specification is provided

5206 **4.11.1.3 Schema Data**

Property	Type	Description
name	string	The name of the data
content	string	The content of the data

5207 **4.11.1.4 Paths**

HTTP	Path	Summary
get	/mapreduce	Returns the data identified by the MapReduce resource
get	/mapreduce/{name}	Returns the data identified by the map and function
put	/mapreduce/map/{name}	Create or update the map function
get	/mapreduce/map/{name}	Returns the data identified by the map function
put	/mapreduce/reduce/{name}	Create or update the reduce function
get	/mapreduce/reduce/{name}	Returns the data identified by the reduce function

5208 **4.11.1.4.1 /mapreduce**

5209 **4.11.1.4.1.1 GET /mapreduce**

5210 Returns the data identified by the MapReduce resource

5211 Responses

Code	Description	Schema
200	MapReduce names	array[String]
401	Not authorized	String

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

5212 **4.11.1.4.2/mapreduce/{name}**

5213 **4.11.1.4.2.1 GET /mapreduce/{name}**

5214 Returns the data identified by the reduce function.

5215 Responses

Code	Description	Schema
200	The data identified by reduce	array[Data]
401	Not authorized	String
404	The resource could not be found	String

5216 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the function	True	String

5217 **4.11.1.4.3/mapreduce/map/{name}**

5218 **4.11.1.4.3.1 PUT /mapreduce/map/{name}**

5219 Create or update the map function

5220 Responses

Code	Description	Schema
200	The map function was created or updated	String
401	Not authorized	String
404	The resource could not be found	String

5221 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the function	True	String

5222 Request Body

Located in	Description	Required	Schema
Body	The new default to create	True	Map

5223 **4.11.1.4.3.2 GET /mapreduce/map/{name}**

5224 Returns the data identified by the map function

5225 Responses

Code	Description	Schema
200	The data identified by map	array[Data]
401	Not authorized	String
404	The resource could not be found	String

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r-1>

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

5226 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the function	True	String

5227 **4.11.1.4.4/mapreduce/reduce/{name}**

5228 **4.11.1.4.4.1 PUT /mapreduce/reduce/{name}**

5229 Create or update the reduce function

5230 Responses

Code	Description	Schema
200	The reduce function was created or updated	String
401	Not authorized	String
404	The resource could not be found	String

5231 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the function	True	String

5232 Request Body

Located in	Description	Required	Schema
Body	The new default to create	True	Reduce

5233 **4.11.1.4.4.2 GET /mapreduce/reduce/{name}**

5234 Returns the data identified by the reduce function

5235 Responses

Code	Description	Schema
200	The data identified by reduce	array[Data]
401	Not authorized	String
404	The resource could not be found	String

5236 Parameters

Name	Located in	Description	Required	Schema
name	path	The name of the function	True	String

5237 **4.11.1.5 mapreduce.yaml**

5238 openapi: "3.0.2"

5239 info:

5240 version: 3.2.0

5241 x-date: 17-06-2019

5242 x-status: defined

5243 title: MapReduce

5244 description: |-

5245

5246 A service to store the information of a mapreduce deployment definition.

```

5247     All of the attributes are stored as Strings.
5248
5249 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
5250 contact:
5251   name: NIST BDRA Interface Subgroup
5252   url: https://cloudmesh-community.github.io/nist
5253 license:
5254   name: Apache 2.0
5255   url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
5256 servers:
5257   - url: /cloudmesh/v3
5258 paths:
5259   /mapreduce:
5260     get:
5261       tags:
5262         - mapreduce
5263       summary: Returns the data identified by the mapreduce resource
5264       description: Returns the data identified by the mapreduce resource
5265       operationId: cloudmesh.mapreduce.list
5266       responses:
5267         '200':
5268           description: mapreduce names
5269           content:
5270             application/json:
5271               schema:
5272                 type: array
5273                 items:
5274                   type: string
5275         '401':
5276           description: Not authorized
5277   /mapreduce/{name}:
5278     get:
5279       tags:
5280         - mapreduce
5281       summary: Returns the data identified by the map and function
5282       description: Returns the data identified by the reduce function.
5283       operationId: cloudmesh.mapreduce.get
5284       parameters:
5285         - name: name
5286           in: path
5287           required: true
5288           schema:
5289             type: string
5290           description: The name of the function
5291       responses:
5292         '200':
5293           description: The data identified by reduce
5294           content:
5295             application/json:
5296               schema:
5297                 type: array
5298                 items:
5299                   $ref: '#/components/schemas/Data'
5300         '401':
5301           description: Not authorized
5302         '404':
5303           description: The resource could not be found
5304   /mapreduce/map/{name}:
5305     put:
5306       tags:
5307         - mapreduce
5308       summary: Create or update the map function
5309       description: Create or update the map function
5310       operationId: cloudmesh.mapreduce.map.put
5311       parameters:
5312         - name: name
5313           in: path
5314           required: true
5315           schema:

```

```

5316         type: string
5317         description: The name of the function
5318     requestBody:
5319         description: The new default to create
5320         required: true
5321         content:
5322             application/json:
5323                 schema:
5324                     $ref: '#/components/schemas/Map'
5325     responses:
5326         '200':
5327             description: The map function was created or updated
5328         '401':
5329             description: Not authorized
5330         '404':
5331             description: The resource could not be found
5332     get:
5333         tags:
5334             - mapreduce
5335         summary: Returns the data identified by the map function
5336         description: Returns the data identified by the map function
5337         operationId: cloudmesh.mapreduce.map.get
5338         parameters:
5339             - name: name
5340               in: path
5341               required: true
5342               schema:
5343                 type: string
5344                 description: The name of the function
5345         responses:
5346             '200':
5347                 description: The data identified by map
5348                 content:
5349                     application/json:
5350                         schema:
5351                             type: array
5352                             items:
5353                                 $ref: '#/components/schemas/Data'
5354             '401':
5355                 description: Not authorized
5356             '404':
5357                 description: The resource could not be found
5358 /mapreduce/reduce/{name}:
5359     put:
5360         tags:
5361             - mapreduce
5362         summary: Create or update the reduce function
5363         description: Create or update the reduce function
5364         operationId: cloudmesh.mapreduce.reduce.put
5365         parameters:
5366             - name: name
5367               in: path
5368               required: true
5369               schema:
5370                 type: string
5371                 description: The name of the function
5372         requestBody:
5373         description: The new default to create
5374         required: true
5375         content:
5376             application/json:
5377                 schema:
5378                     $ref: '#/components/schemas/Reduce'
5379     responses:
5380         '200':
5381             description: The reduce function was created or updated
5382         '401':
5383             description: Not authorized
5384         '404':

```



```

5385     description: The resource could not be found
5386 get:
5387   tags:
5388     - mapreduce
5389   summary: Returns the data identified by the reduce function
5390   description: Returns the data identified by the reduce function
5391   operationId: cloudmesh.mapreduce.reduce.get
5392   parameters:
5393     - name: name
5394       in: path
5395       required: true
5396       schema:
5397         type: string
5398       description: The name of the function
5399   responses:
5400     '200':
5401       description: The data identified by reduce
5402       content:
5403         application/json:
5404           schema:
5405             type: array
5406             items:
5407               $ref: '#/components/schemas/Data'
5408     '401':
5409       description: Not authorized
5410     '404':
5411       description: The resource could not be found
5412 components:
5413   schemas:
5414     Map:
5415       type: object
5416       description: The specification of the map function
5417       properties:
5418         name:
5419           type: string
5420           description: The name of the map function
5421         kind:
5422           type: string
5423           description: The kind in which the specification is provided
5424         content:
5425           type: string
5426           description: The kind in which the specification is provided
5427     Reduce:
5428       type: object
5429       description: The specification of the reduce function
5430       properties:
5431         name:
5432           type: string
5433           description: The name of the reduce function
5434         kind:
5435           type: string
5436           description: The kind in which the specification is provided
5437         content:
5438           type: string
5439           description: The kind in which the specification is provided
5440     Data:
5441       type: object
5442       description: The specification of the function
5443       properties:
5444         name:
5445           type: string
5446           description: The name of the data
5447         content:
5448           type: string
5449           description: The content of the data

```

5450 **4.12 COMPUTE MANAGEMENT: FUNCTIONS**

5451 **4.12.1 MICROSERVICE**

5452 As part of microservices, a function with parameters that can be invoked has been defined.

5453 **4.12.1.1 Schema Microservice**

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

5454 **4.12.1.2 Paths**

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

5455 **4.12.1.2.1 /microservice**

5456 **4.12.1.2.1.1 GET /microservice**

5457 Returns a list of all microservices

5458 Responses

Code	Description	Schema
200	The list of microservicees	array[Microservice]
401	Not authorized	String

5459 **4.12.1.2.2 /microservice/{name}**

5460 **4.12.1.2.2.1 GET /microservice/{name}**

5461 Returns the named microservice

5462 Responses

Code	Description	Schema
200	Returns the microservice	Microservice
401	Not authorized	String
404	The named microservice could not be found	String

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

5463 Parameters

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

5464 **4.12.1.2.2.2 PUT /microservice/{name}**

5465 Sets the named microservice

5466 Responses

Code	Description	Schema
200	Microservice updated	String
400	Error updating microservice	String
401	Not authorized	String

5467 Request Body

Located in	Description	Required	Schema
Body	The new microservice to create	True	Microservice

5468 **4.12.1.2.2.3 DELETE /microservice/{name}**

5469 Deletes an microservice by name

5470 Responses

Code	Description	Schema
200	Deletion successful	String
400	Error deleting microservice	String
401	Not authorized	String
404	The named microservice could not be found	String

5471 Parameters

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

5472 **4.12.1.3 microservice.yaml**

5473 openapi: "3.0.2"

5474 info:

5475 version: 3.2.0

5476 x-date: 17-06-2019

5477 x-status: defined

5478 title: Microservice

5479 description: |-

5480

5481 As part of microservices, a function with parameters that can be

5482 invoked has been defined.

5483

5484 termsOfService: "https://github.com/cloudmesh-nist/blob/master/LICENSE.txt"

5485 contact:

5486 name: NIST BDRA Interface Subgroup

5487 url: https://cloudmesh-community.github.io/nist

5488 license:

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

5489     name: Apache 2.0
5490     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
5491 servers:
5492   - url: /cloudmesh/v3
5493 paths:
5494   /microservice:
5495     get:
5496       tags:
5497         - Microservice
5498       summary: Returns a list of microservices
5499       description: Returns a list of all microservices
5500       operationId: cloudmesh.microservice.list
5501       responses:
5502         '200':
5503           description: The list of microservices
5504           content:
5505             application/json:
5506               schema:
5507                 type: array
5508                 items:
5509                   $ref: '#/components/schemas/Microservice'
5510         '401':
5511           description: Not authorized
5512   /microservice/{name}:
5513     get:
5514       tags:
5515         - Microservice
5516       summary: Returns the named microservice
5517       description: Returns the named microservice
5518       operationId: cloudmesh.microservice.find_by_name
5519       parameters:
5520         - name: name
5521           in: path
5522           required: true
5523           schema:
5524             type: string
5525           description: The name of the microservice
5526       responses:
5527         '200':
5528           description: Returns the microservice
5529           content:
5530             application/json:
5531               schema:
5532                 $ref: '#/components/schemas/Microservice'
5533         '401':
5534           description: Not authorized
5535         '404':
5536           description: The named microservice could not be found
5537     put:
5538       tags:
5539         - Microservice
5540       summary: Set an microservice
5541       description: Sets the named microservice
5542       operationId: cloudmesh.microservice.add
5543       requestBody:
5544         description: The new microservice to create
5545         required: true
5546         content:
5547           application/json:
5548             schema:
5549               $ref: '#/components/schemas/Microservice'
5550       responses:
5551         '200':
5552           description: Microservice updated
5553         '400':
5554           description: Error updating microservice
5555         '401':
5556           description: Not authorized
5557     delete:

```

```

5558     tags:
5559       - Microservice
5560     summary: Deletes the named microservice
5561     description: Deletes an microservice by name
5562     operationId: cloudmesh.microservice.delete_by_name
5563     parameters:
5564       - name: name
5565         in: path
5566         required: true
5567         schema:
5568           type: string
5569     description: The name of the microservice
5570     responses:
5571       '200':
5572         description: Deletion successful
5573       '400':
5574         description: Error deleting microservice
5575       '401':
5576         description: Not authorized
5577       '404':
5578         description: The named microservice could not be found
5579     components:
5580       schemas:
5581         Microservice:
5582           type: object
5583           description: The microservice
5584           properties:
5585             name:
5586               type: string
5587               description: Name of the microservice
5588             endpoint:
5589               type: string
5590               description: The end point of the microservice
5591             function:
5592               type: string
5593               description: The function the microservice represents
5594           description:
5595             type string:
5596               description: The description of the microservice
    
```

5597 4.13 RESERVATION

5598 4.13.1 RESERVATION

5599 Some services may consume a considerable amount of resources, necessitating the reservation of
 5600 resources.

5601 4.13.1.1 Schema Reservation

Property	Type	Description
name	string	Name of the reservation
service	string	The name of the service for which the reservation is applied
description	string	The description of the reservation
start	string(date)	The start time and date
end	string(date)	The end time and date

5602 **4.13.1.2 Paths**

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

5603 **4.13.1.2.1/reservation**

5604 **4.13.1.2.1.1 GET /reservation**

5605 Returns a list of all reservations

5606 Responses

Code	Description	Schema
200	The list of reservations	array[Reservation]
400	No Reservations found	String

5607 **4.13.1.2.2/reservation/{name}**

5608 **4.13.1.2.2.1 GET /reservation/{name}**

5609 Returns reservation by name

5610 Responses

Code	Description	Schema
200	Returning the information of the reservation	Reservation
400	No reservation found	String
401	Not authorized	String
404	The named reservation could not be found	String

5611 Parameters

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

5612 **4.13.1.2.2.2 PUT /reservation/{name}**

5613 Uploads a reservation to the list of reservations

5614 Responses

Code	Description	Schema
200	Reservation updated	String
400	Error updating reservation	String

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

5615	Request Body			
	Located in	Description	Required	Schema
	Body	The reservation to be uploaded	True	Reservation

5616 **4.13.1.2.2.3 DELETE /reservation/{name}**

5617 Deletes a reservation by name

5618 Responses

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

5619 Parameters

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

5620 **4.13.1.3 reservation.yaml**

```

5621 openapi: '3.0.2'
5622 info:
5623   version: 3.2.0
5624   x-date: 17-06-2019
5625   x-status: defined
5626   title: Reservation
5627   description: |-
5628
5629     Some services may consume a considerable amount of resources,
5630     necessitating the reservation of resources.
5631
5632   termsOfService: 'https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt'
5633   contact:
5634     name: NIST BDRA Interface Subgroup
5635     url: https://cloudmesh-community.github.io/nist
5636   license:
5637     name: Apache 2.0
5638     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
5639 servers:
5640   - url: /cloudmesh/v3
5641 paths:
5642   /reservation:
5643     get:
5644       tags:
5645       - Reservation
5646       summary: Returns a list of reservations
5647       description: Returns a list of all reservations
5648       operationId: cloudmesh.reservation.list
5649       responses:
5650         '200':
5651           description: The list of reservations
5652           content:
5653             application/json:
5654               schema:
5655                 type: array
5656                 items:
5657                   $ref: '#/components/schemas/Reservation'

```

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

```

5658         '400':
5659             description: No Reservations found
5660 /reservation/{name}:
5661     get:
5662         tags:
5663             - Reservation
5664         summary: Returns the named reservation
5665         description: Returns an reservation by name
5666         operationId: cloudmesh.reservation.find_by_name
5667         parameters:
5668             - name: name
5669               in: path
5670               required: true
5671               schema:
5672                 type: string
5673             description: The name of the reservation
5674         responses:
5675             '200':
5676                 description: Returning the information of the reservation
5677                 content:
5678                     application/json:
5679                         schema:
5680                             $ref: '#/components/schemas/Reservation'
5681             '400':
5682                 description: No reservation found
5683             '401':
5684                 description: Not authorized
5685             '404':
5686                 description: The named reservation could not be found
5687     put:
5688         tags:
5689             - Reservation
5690         summary: Uploads a reservation to the list of reservations
5691         description: Uploads a reservation to the list of reservations
5692         operationId: cloudmesh.reservation.add
5693         requestBody:
5694             description: The reservation to be uploaded
5695             required: true
5696             content:
5697                 application/json:
5698                     schema:
5699                         $ref: '#/components/schemas/Reservation'
5700         responses:
5701             '200':
5702                 description: Reservation updated
5703             '400':
5704                 description: Error updating reservation
5705     delete:
5706         tags:
5707             - Reservation
5708         summary: Deletes the named reservation
5709         description: Deletes an reservation by name
5710         operationId: cloudmesh.reservation.delete_by_name
5711         parameters:
5712             - name: name
5713               in: path
5714               required: true
5715               schema:
5716                 type: string
5717             description: The name of the reservation
5718         responses:
5719             '200':
5720                 description: Deletion successful
5721             '400':
5722                 description: No reservation found
5723             '401':
5724                 description: Not authorized
5725             '404':
5726                 description: The named reservation could not be found

```



```

5727 components:
5728   schemas:
5729     Reservation:
5730       type: object
5731       description: The reservation
5732       properties:
5733         name:
5734           type: string
5735           description: Name of the reservation
5736         service:
5737           type: string
5738           description: The name of the service for which the reservation is
5739           applied
5740         description:
5741           type: string
5742           description: The description of the reservation
5743         start:
5744           type: string
5745           format: date
5746           description: The start time and date
5747         end:
5748           type: string
5749           format: date
5750           description: The end time and date
    
```

5751 4.14 DATA STREAMS

5752 4.14.1 *STREAM*

5753 The stream object describes a data flow, providing information about the rate and number of items
 5754 exchanged while issuing requests to the stream. A stream may return data items in a specific format that is
 5755 defined by the stream.

5756 4.14.1.1 *Schema Stream*

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
endpoint	string	The endpoint of the stream
protocol	string	DThe definition of the protocol used

5757 4.14.1.2 *Paths*

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

5758 **4.14.1.2.1/stream**

5759 **4.14.1.2.1.1 GET /stream**

5760 Returns a list of all streams

5761 Responses

Code	Description	Schema
200	The list of streams	array[Stream]
400	No Stream found	String
401	Not authorized	String

5762 **4.14.1.2.2/stream/{name}**

5763 **4.14.1.2.2.1 GET /stream/{name}**

5764 Returns a stream by name

5765 Responses

Code	Description	Schema
200	Returning the information of the stream	Stream
401	Not authorized	String
404	The named stream could not be found	String

5766 Parameters

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

5767 **4.14.1.2.2.2 PUT /stream/{name}**

5768 Sets the named stream

5769 Responses

Code	Description	Schema
200	Stream updated	String
401	Not authorized	String

5770 Request Body

Located in	Description	Required	Schema
Body	The new stream to create	True	Stream

5771 **4.14.1.2.2.3 DELETE /stream/{name}**

5772 Deletes a stream by name

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

5773 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named stream could not be found	String

5774 Parameters

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

5775 **4.14.1.3 stream.yaml**

```

5776 openapi: "3.0.2"
5777 info:
5778   version: 3.2.0
5779   x-date: 17-06-2019
5780   x-status: defined
5781   title: Stream
5782   description: |-
5783
5784     The stream object describes a data flow, providing information
5785     about the rate and number of items exchanged while issuing requests
5786     to the stream. A stream may return data items in a specific format
5787     that is defined by the stream.
5788
5789   termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
5790   contact:
5791     name: NIST BDRA Interface Subgroup
5792     url: https://cloudmesh-community.github.io/nist
5793   license:
5794     name: Apache 2.0
5795     url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
5796   servers:
5797     - url: /cloudmesh/v3
5798   paths:
5799     /stream:
5800       get:
5801         tags:
5802           - Stream
5803         summary: Returns a list of streams
5804         description: Returns a list of all streams
5805         operationId: cloudmesh.stream.list
5806         responses:
5807           '200':
5808             description: The list of streams
5809             content:
5810               application/json:
5811                 schema:
5812                   type: array
5813                   items:
5814                     $ref: '#/components/schemas/Stream'
5815           '400':
5816             description: No Stream found
5817           '401':
5818             description: Not authorized
5819     /stream/{name}:
5820       get:
5821         tags:
5822           - Stream
5823         summary: Returns the named stream
5824         description: Returns an stream by name
5825         operationId: cloudmesh.stream.find_by_name
    
```

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

```

5826     parameters:
5827     - name: name
5828       in: path
5829       required: true
5830       schema:
5831         type: string
5832       description: The name of the stream
5833     responses:
5834     '200':
5835       description: Returning the information of the stream
5836       content:
5837         application/json:
5838           schema:
5839             $ref: '#/components/schemas/Stream'
5840     '401':
5841       description: Not authorized
5842     '404':
5843       description: The named stream could not be found
5844   put:
5845     tags:
5846     - Stream
5847     summary: Set an stream
5848     description: Sets the named stream
5849     operationId: cloudmesh.stream.add
5850     requestBody:
5851       description: The new stream to create
5852       required: true
5853       content:
5854         application/json:
5855           schema:
5856             $ref: '#/components/schemas/Stream'
5857     responses:
5858     '200':
5859       description: Stream updated
5860     '401':
5861       description: Not authorized
5862   delete:
5863     tags:
5864     - Stream
5865     summary: Deletes the named stream
5866     description: Deletes an stream by name
5867     operationId: cloudmesh.stream.delete_by_name
5868     parameters:
5869     - name: name
5870       in: path
5871       required: true
5872       schema:
5873         type: string
5874       description: The name of the stream
5875     responses:
5876     '200':
5877       description: Deletion successful
5878     '401':
5879       description: Not authorized
5880     '404':
5881       description: The named stream could not be found
5882   components:
5883     schemas:
5884     Stream:
5885       type: object
5886       description: The stream
5887     properties:
5888     name:
5889       type: string
5890       description: Name of the stream
5891     format:
5892       type: string
5893       description: Format of the stream
5894     rate:

```

5895 type: integer
 5896 description: The rate of messages
 5897 limit:
 5898 type: integer
 5899 description: The limit of items send
 5900 endpoint:
 5901 type: string
 5902 description: The endpoint of the stream
 5903 protocol:
 5904 type: string
 5905 description: DThe definition of the protocol used

5906 **4.14.2 FILTER**

5907 Filters can operate on a variety of objects and reduce the information received based on a search criterion.

5908 **4.14.2.1 Schema Filter**

Property	Type	Description
name	string	Name of the filter
function	string	The function used to filter the data in the stream
kind	string	The filter kind or type

5909 **4.14.2.2 Paths**

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

5910 **4.14.2.2.1 /filter**

5911 **4.14.2.2.1.1 GET /filter**

5912 Returns a list of all filters

5913 Responses

Code	Description	Schema
200	The list of filters	array[Filter]
401	Not authorized	String

5914 **4.14.2.2.2 /filter/{name}**

5915 **4.14.2.2.2.1 GET /filter/{name}**

5916 Returns a filter by name

This publication is available free of charge from: https://doi.org/10.6028/NIST.SP.1500-9r1

5917 Responses

Code	Description	Schema
200	Returning the information of the filter	Filter
401	Not authorized	String
404	The named filter could not be found	String

5918 Parameters

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

5919 **4.14.2.2.2 PUT /filter/{name}**

5920 Sets the named filter

5921 Responses

Code	Description	Schema
200	Filter updated	String
401	Not authorized	String
400	Error updating filter	String

5922 Request Body

Located in	Description	Required	Schema
Body	The new filter to create	True	Filter

5923 **4.14.2.2.3 DELETE /filter/{name}**

5924 Deletes a filter by name

5925 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named filter could not be found	String

5926 Parameters

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

5927 **4.14.2.3 filter.yaml**

5928 openapi: "3.0.2"
 5929 info:
 5930 version: 3.2.0
 5931 x-date: 17-06-2019
 5932 x-status: defined
 5933 title: Filter
 5934 description: |-
 5935

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

```

5936     Filters can operate on a variety of objects and reduce the
5937     information received based on a search criterion.
5938
5939     termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"
5940     contact:
5941       name: NIST BDRA Interface Subgroup
5942       url: https://cloudmesh-community.github.io/nist
5943     license:
5944       name: Apache 2.0
5945       url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt
5946     servers:
5947       - url: /cloudmesh/v3
5948     paths:
5949       /filter:
5950         get:
5951           tags:
5952             - Filter
5953           summary: Returns a list of filters
5954           description: Returns a list of all filters
5955           operationId: cloudmesh.filter.list
5956           responses:
5957             '200':
5958               description: The list of filters
5959               content:
5960                 application/json:
5961                   schema:
5962                     type: array
5963                     items:
5964                       $ref: '#/components/schemas/Filter'
5965             '401':
5966               description: Not authorized
5967       /filter/{name}:
5968         get:
5969           tags:
5970             - Filter
5971           summary: Returns the named filter
5972           description: Returns an filter by name
5973           operationId: cloudmesh.filter.find_by_name
5974           parameters:
5975             - name: name
5976               in: path
5977               required: true
5978               schema:
5979                 type: string
5980               description: The name of the filter
5981           responses:
5982             '200':
5983               description: Returning the information of the filter
5984               content:
5985                 application/json:
5986                   schema:
5987                     $ref: '#/components/schemas/Filter'
5988             '401':
5989               description: Not authorized
5990             '404':
5991               description: The named filter could not be found
5992         put:
5993           tags:
5994             - Filter
5995           summary: Set an filter
5996           description: Sets the named filter
5997           operationId: cloudmesh.filter.add
5998           requestBody:
5999             description: The new filter to create
6000             required: true
6001             content:
6002               application/json:
6003                 schema:
6004                   $ref: '#/components/schemas/Filter'

```

```

6005     responses:
6006       '200':
6007         description: Filter updated
6008       '401':
6009         description: Not authorized
6010       '400':
6011         description: Error updating filter
6012     delete:
6013       tags:
6014         - Filter
6015       summary: Deletes the named filter
6016       description: Deletes an filter by name
6017       operationId: cloudmesh.filter.delete_by_name
6018       parameters:
6019         - name: name
6020           in: path
6021           required: true
6022           schema:
6023             type: string
6024           description: The name of the filter
6025       responses:
6026         '200':
6027           description: Deletion successful
6028         '401':
6029           description: Not authorized
6030         '404':
6031           description: The named filter could not be found
6032     components:
6033       schemas:
6034         Filter:
6035           type: object
6036           description: The filter
6037           properties:
6038             name:
6039               type: string
6040               description: Name of the filter
6041             function:
6042               type: string
6043               description: The function used to filter the data
6044                 in the stream
6045             kind:
6046               type: string
6047               description: The filter kind or type

```

6048 4.15 DEPLOYMENT

6049 4.15.1 DEPLOYMENT

6050 A resource to store software deployments. The deployment is formulated in a specification file. To
6051 distinguish the format of the specification file, a string is used that defines the kind of the deployment. In
6052 case the specification uses an external service, an endpoint to the service can be used and the name of the
6053 specification is used to identify the deployment.

6054 **4.15.1.1 Schema Deployment**

Property	Type	Description
name	string	The name of the deployment
kind	string	The kind of the deployment
specification	string	The specification of the deployment
endpoint	string	The location of the deployment service
endpointname	string	In case an endpoint is used, the endpointname is used to uniquely identify the deployment within the endpoint defined service

6055 **4.15.1.2 Paths**

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

6056 **4.15.1.2.1 /deployment**

6057 **4.15.1.2.1.1 GET /deployment**

6058 Returns a list of all deployments

6059 Responses

Code	Description	Schema
200	The list of deployments	array[Deployment]
401	Not authorized	String

6060 **4.15.1.2.2 /deployment/{name}**

6061 **4.15.1.2.2.1 GET /deployment/{name}**

6062 Returns a deployment by name

6063 Responses

Code	Description	Schema
200	Returning the information of the deployment	Deployment
401	Not authorized	String
404	The named deployment could not be found	String

6064 Parameters

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

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

6065 **4.15.1.2.2.2 PUT /deployment/{name}**

6066 Sets the named deployment

6067 Responses

Code	Description	Schema
200	Deployment updated	String
401	Not authorized	String

6068 Request Body

Located in	Description	Required	Schema
Body	The new deployment to create	True	Deployment

6069 **4.15.1.2.2.3 DELETE /deployment/{name}**

6070 Deletes a deployment by name

6071 Responses

Code	Description	Schema
200	Deletion successful	String
401	Not authorized	String
404	The named deployment could not be found	String

6072 Parameters

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

6073 **4.15.1.3 deployment.yaml**

6074 openapi: "3.0.2"

6075 info:

6076 version: 3.2.0

6077 x-date: 17-06-2019

6078 x-status: defined

6079 title: Deployment

6080 description: |-

6081
6082 A resource to store software deployments. The deployment is formulated in
6083 a specification file. To distinguish the format of the specification file,
6084 a string is used that defines the kind of the deployment. In case the
6085 specification uses an external service, an
6086 endpoint to the service can be used and the name of the specification is
6087 used to identify the deployment.

6088
6089 termsOfService: "https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt"

6090 contact:

6091 name: NIST BDRA Interface Subgroup

6092 url: https://cloudmesh-community.github.io/nist

6093 license:

6094 name: Apache 2.0

6095 url: https://github.com/cloudmesh/cloudmesh-nist/blob/master/LICENSE.txt

6096 servers:

6097 - url: /cloudmesh/v3

6098 paths:

6099 /deployment:

6100 get:

```

6101     tags:
6102     - Deployment
6103     summary: Returns a list of deployments
6104     description: Returns a list of all deployments
6105     operationId: cloudmesh.deployment.list
6106     responses:
6107       '200':
6108         description: The list of deployments
6109         content:
6110           application/json:
6111             schema:
6112               type: array
6113               items:
6114                 $ref: '#/components/schemas/Deployment'
6115       '401':
6116         description: Not authorized
6117 /deployment/{name}:
6118   get:
6119     tags:
6120     - Deployment
6121     summary: Returns the named deployment
6122     description: Returns an deployment by name
6123     operationId: cloudmesh.deployment.find_by_name
6124     parameters:
6125     - name: name
6126       in: path
6127       required: true
6128       schema:
6129         type: string
6130     description: The name of the deployment
6131     responses:
6132       '200':
6133         description: Returning the information of the deployment
6134         content:
6135           application/json:
6136             schema:
6137               $ref: '#/components/schemas/Deployment'
6138       '401':
6139         description: Not authorized
6140       '404':
6141         description: The named deployment could not be found
6142   put:
6143     tags:
6144     - Deployment
6145     summary: Set an deployment
6146     description: Sets the named deployment
6147     operationId: cloudmesh.deployment.add
6148     requestBody:
6149       description: The new deployment to create
6150       required: true
6151       content:
6152         application/json:
6153           schema:
6154             $ref: '#/components/schemas/Deployment'
6155     responses:
6156       '200':
6157         description: Deployment updated
6158       '401':
6159         description: Not authorized
6160   delete:
6161     tags:
6162     - Deployment
6163     summary: Deletes the named deployment
6164     description: Deletes an deployment by name
6165     operationId: cloudmesh.deployment.delete_by_name
6166     parameters:
6167     - name: name
6168       in: path
6169       required: true

```

```

6170     schema:
6171         type: string
6172     description: The name of the deployment
6173     responses:
6174         '200':
6175             description: Deletion successful
6176         '401':
6177             description: Not authorized
6178         '404':
6179             description: The named deployment could not be found
6180 components:
6181     schemas:
6182         Deployment:
6183             type: object
6184             description: the deployment
6185             properties:
6186                 name:
6187                     type: string
6188                     description: The name of the deployment
6189                 kind:
6190                     type: string
6191                     description: The kind of the deployment
6192                 specification:
6193                     type: string
6194                     description: The specification of the deployment
6195                 endpoint:
6196                     type: string
6197                     description: The location of the deployment service
6198                 endpointname:
6199                     type: string
6200                 description: in case an endpoint is used, the endpointname is used
6201                             to uniquely identify the deployment within the
6202                             endpoint defined service
6203

```

6204 **Appendix A: Acronyms and Terms**

6205	ACID	Atomicity, Consistency, Isolation, Durability
6206	API	Application Programming Interface
6207	ASCII	American Standard Code for Information Interchange
6208	BASE	Basically Available, Soft state, Eventual consistency
6209	CIDR	Classless Inter-Domain Routing
6210	Container	See http://csrc.nist.gov/publications/drafts/800-180/sp800-180_draft.pdf
6211	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 .
6212		
6213		
6214	CPU	Central Processing Unit
6215	DevOps	A clipped compound of software DEvelopment and information technology OPerationS
6216		
6217	Deployment	The action of installing software on resources
6218	DSA	Digital Signature Algorithm
6219	ETL	Extract, Transform, Load
6220	ELT	Extract, Load, Transform
6221	FIFO	first-in-first-out
6222	FPGAs	field-programmable gate arrays
6223	GPUs	graphics processing units
6224	HTTP	HyperText Transfer Protocol
6225	HTTPS	HTTP Secure
6226	Hybrid Cloud	See http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf .
6227		
6228	ICMP	Internet Control Message Protocol
6229	IP	Internet Protocol
6230	ITL	Information Technology Laboratory
6231	LIFO	last-in-first-out
6232	MAC	Media Access Control
6233	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.
6234		
6235		
6236	MTU	maximum transmission unit
6237	NBDIF	NIST Big Data Interoperability Framework

6238	NBD-PWG	NIST Big Data Public Working Group
6239	NBDRA	NIST Big Data Reference Architecture
6240	NBDRAI	NIST Big Data Reference Architecture Interface
6241	NIC	Network Interface Controller
6242	NIST	National Institute of Standards and Technology
6243	NON	Network of Nodes
6244	OS	Operating System
6245	RAM	Random Access Memory
6246	REST	REpresentational State Transfer
6247	Replica	A duplicate of a file on another resource to avoid costly transfer costs in case of
6248		frequent access.
6249	RSA	Rivest–Shamir–Adleman
6250	SSH	Secure Shell
6251	Serverless Computing	Serverless computing specifies the paradigm of function as a service (FaaS). It is
6252		a cloud computing code execution model in which a cloud provider manages the
6253		function deployment and utilization while clients can utilize them. The charge
6254		model is based on execution of the function rather than the cost to manage and
6255		host the VM or container.
6256	Software Stack	A set of programs and services that are installed on a resource to support
6257		applications.
6258	TCP	Transmission Control Protocol
6259	UDP	User Datagram Protocol
6260	URI	Uniform Resource Identifier
6261	Virtual File System	An abstraction layer on top of a distributed physical file system to allow easy
6262		access to the files by the user or application.
6263	VM	Virtual Machine. A virtual machine is a software computer that, like a physical
6264		computer, runs an operating system and applications. The VM is composed of a
6265		set of specification and configuration files and is backed by the physical
6266		resources of a host.
6267	Virtual Cluster	A virtual cluster is a software cluster that integrate either VMs, containers, or
6268		physical resources into an agglomeration of compute resources. A virtual cluster
6269		allows users to authenticate and authorize to the virtual compute nodes to utilize
6270		them for calculations. Optional high-level services that can be deployed on a
6271		virtual cluster may simplify interaction with the virtual cluster or provide higher-
6272		level services.
6273	Workflow	The sequence of processes or tasks
6274	WWW	World Wide Web
6275		
6276		

This publication is available free of charge from: <https://doi.org/10.6028/NIST.SP.1500-9r1>

Appendix B: Bibliography

- [1] W. Chang, “NIST Big Data Public Working Group (NBD-PWG),” 2018. [Online]. Available: <http://bigdatawg.nist.gov/home.php>. [Accessed: 03-Apr-2018]
- [2] W. L. Chang (Co-Chair), N. Grady (Subgroup Co-chair), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 1, Big Data Definitions (NIST SP 1500-1 VERSION 3),” Gaithersburg MD, Sep. 2019 [Online]. Available: <https://doi.org/10.6028/NIST.SP.1500-1r2>
- [3] W. L. Chang (Co-Chair), N. Grady (Subgroup Co-chair), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 2, Big Data Taxonomies (NIST SP 1500-2 VERSION 3),” Gaithersburg, MD, Sep. 2019 [Online]. Available: <https://doi.org/10.6028/NIST.SP.1500-2r2>
- [4] W. L. Chang (Co-Chair), G. Fox (Subgroup Co-chair), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 3, Big Data Use Cases and General Requirements (NIST SP 1500-3 VERSION 3),” Gaithersburg, MD, Sep. 2019 [Online]. Available: <https://doi.org/10.6028/NIST.SP.1500-3r2>
- [5] W. L. Chang (Co-Chair), A. Roy (Subgroup Co-chair), M. Underwood (Subgroup Co-chair), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 4, Big Data Security and Privacy (NIST SP 1500-4 VERSION 3),” Gaithersburg, MD, Sep. 2019 [Online]. Available: <https://doi.org/10.6028/NIST.SP.1500-4r2>
- [6] W. L. Chang (Co-Chair), S. Mishra (Editor), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 5, Big Data Architectures White Paper Survey (NIST SP 1500-5 VERSION 1),” Sep. 2015.
- [7] W. L. Chang (Co-Chair), D. Boyd (Subgroup Co-chair), O. Levin (Version 1 Subgroup Co-Chair), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 6, Big Data Reference Architecture (NIST SP 1500-6 VERSION 3),” Gaithersburg MD, Sep. 2019 [Online]. Available: <https://doi.org/10.6028/NIST.SP.1500-6r2>
- [8] 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, “NIST Big Data Interoperability Framework: Volume 7, Big Data Standards Roadmap (NIST SP 1500-7 VERSION 3),” Gaithersburg, MD, Sep. 2019 [Online]. Available: <https://doi.org/10.6028/NIST.SP.1500-7r2>
- [9] W. L. Chang (Co-Chair), R. Reinsch (Subgroup Co-chair), C. Austin (Editor), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 9, Adoption and Modernization (NIST SP 1500-10 VERSION 2),” Gaithersburg, MD, Sep. 2019 [Online]. Available: <https://doi.org/10.6028/NIST.SP.1500-10r1>

- [10] T. White House Office of Science and Technology Policy, “Big Data is a Big Deal,” *OSTP Blog*, 2012. [Online]. Available: <http://www.whitehouse.gov/blog/2012/03/29/big-data-big-deal>. [Accessed: 21-Feb-2014]
- [11] W. L. Chang (Co-Chair), N. Grady (Subgroup Co-chair), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 1, Big Data Definitions (NIST SP 1500-1 VERSION 2),” Jun. 2018 [Online]. Available: <https://doi.org/10.6028/NIST.SP.1500-1r1>
- [12] W. L. Chang (Co-Chair) and G. Fox (Subgroup Co-chair), “NIST Big Data Interoperability Framework: Volume 3, Big Data Use Cases and General Requirements (NIST SP 1500-3 VERSION 2),” Jun. 2018.
- [13] W. L. Chang (Co-Chair), A. Roy (Subgroup Co-chair), M. Underwood (Subgroup Co-chair), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 4, Big Data Security and Privacy (NIST SP 1500-4 VERSION 2),” Jun. 2018.
- [14] W. L. Chang (Co-Chair), D. Boyd (Subgroup Co-chair), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 6, Big Data Reference Architecture (NIST SP 1500-6 VERSION 2),” Jun. 2018.
- [15] W. L. Chang (Co-Chair), R. Reinsch (Subgroup Co-chair), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 7, Big Data Standards Roadmap (NIST SP 1500-7 VERSION 2),” Jun. 2018.
- [16] W. L. Chang (Co-Chair), G. von Laszewski (Editor), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 8, Big Data Reference Architecture Interfaces (NIST SP 1500-9 VERSION 1),” Jun. 2018.
- [17] W. L. Chang (Co-Chair), R. Reinsch (Subgroup Co-chair), and NIST Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 9, Adoption and Modernization (NIST SP 1500-10 VERSION 1),” Jun. 2018.
- [18] Department of Defense, “The DoDAF Architecture Framework Version 2.02,” Apr. 2010 [Online]. Available: <https://dodcio.defense.gov/library/dod-architecture-framework/>
- [19] R. Fielding (Editor) and J. Reschke (Editor), “Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content,” *Internet Engineering Task Force (IETF)*, Jun-2014. [Online]. Available: <https://tools.ietf.org/html/rfc7231>
- [20] G. Klyne (Editor) and C. Newman, “Date and Time on the Internet: Timestamps,” *Internet Engineering Task Force, Network Working Group*, Jul-2002. [Online]. Available: <https://xml2rfc.tools.ietf.org/public/rfc/html/rfc3339.html#anchor14>
- [21] Internet2 Middleware Architecture Committee for Education Directory Working Group (MACE-Dir), “eduPerson Object Class Specification (201602),” 2016. [Online]. Available: <http://software.internet2.edu/eduperson/internet2-mace-dir-eduperson-201602.html>. [Accessed: 15-Nov-2017]