[MS-CPREST-Preview]:

Control Plane REST API

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Preliminary Documentation. This particular Open Specifications document provides documentation for past and current releases and/or for the pre-release version of this technology. This document provides final documentation for past and current releases and preliminary documentation, as applicable and specifically noted in this document, for the pre-release version. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. Because this documentation might change between the pre-release version and the final

version of this technology, there are risks in relying on this preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Comments
4/24/2019	Released Preview document.

Table of Contents

1 Introduction

The Control Plane REST API protocol specifies an HTTP-based web service API that deploys data services and applications into a managed cluster environment, and then communicates to its management service APIs to manage high-value data stored in relational databases that has been integrated with high-volume data resources within the dedicated cluster.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

Apache Spark: A parallel processing framework that supports in-memory processing to boost the performance of big-data analytic applications.

Apache YARN: A resource manager and job scheduler used by **Hadoop**.

Apache ZooKeeper: A service for maintaining synchronization in highly available systems.

app proxy: A **pod** deployed in the **control plane** that exposes an endpoint that provides users the ability to interact with the apps deployed within the **big data cluster**.

application: A participant that is responsible for beginning, propagating, and completing an atomic transaction. An application communicates with a transaction manager in order to begin and complete transactions. An application communicates with a transaction manager in order to marshal transactions to and from other applications. An application also communicates in application-specific ways with a resource manager in order to submit requests for work on resources.

Basic: An authentication access type supported by HTTP as defined by [RFC2617].

Bearer: An authentication access type supported by HTTP as defined by [RFC6750].

big data cluster: A grouping of high-value relational data with high-volume big data that provides the computational power of a cluster to increase scalability and performance of applications.

compute pool: A group of one or more **pods** that provide computational resources. The automated creation and management of these pods is coordinated by the Kubernetes master.

control plane: A logical plane that provides management and security for a **Kubernetes cluster**. It contains the Kubernetes master, the **master instance** of the database management system, and other cluster-level services.

data pool: A grouping of **pods** that provides persistent storage for a cluster and is used to ingest data from queries.

docker: An open-source project for automating the deployment of applications as portable, self-sufficient containers that can run on the cloud or on-premises.

Hadoop: An open-source framework that provides distributed processing of large data sets across clusters of computers that use different programming paradigms and software libraries. For more information, see [HADOOP].

Hadoop Distributed File System (HDFS): A core component of Hadoop, consisting of a distributed storage and file system that allows storage of files of various formats across numerous machines or nodes.

JavaScript Object Notation (JSON): A text-based, data interchange format that is used to transmit structured data, typically in Asynchronous JavaScript + XML (AJAX) web applications, as described in [RFC7159]. The JSON format is based on the structure of ECMAScript (Jscript, JavaScript) objects.

JSON Web Token (JWT): A type of token that includes a set of claims encoded as a JSON object. For more information, see [RFC7519].

Kubernetes: An open-source container orchestrator that can scale container deployments according to need.

Kubernetes cluster: A set of computers in which each computer is called a node. A designated master node controls the cluster, and the remaining nodes in the cluster are the worker nodes. A Kubernetes cluster can contain a mixture of physical-machine and virtual-machine nodes.

Kubernetes namespace: Namespaces represent subdivisions within a cluster. A cluster can have multiple namespaces that act as their own independent virtual clusters.

master instance: A server instance that is running in a **big data cluster control plane**. The master instance provides various kinds of functionality in the cluster, such as for connectivity, scale-out query management, and metadata and user databases.

NameNode: A central service in **HDFS** in which clients request to perform operations on files stored in the file system.

persistent volume: A volume that can be mounted to **Kubernetes** to provide persistent storage to a cluster.

pod: A group of containers that are deployed in the same functional unit together.

storage class: A definition that specifies how storage volumes that are used for persistent storage are to be configured.

storage pool: A group of disks where all of the storage space on all of the disks is aggregated and managed as a single unit.

Uniform Resource Identifier (URI): A string that identifies a resource. The URI is an addressing mechanism defined in Internet Engineering Task Force (IETF) Uniform Resource Identifier (URI): Generic Syntax [RFC3986].

universally unique identifier (UUID): A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and RPC objects. UUIDs are highly likely to be unique. UUIDs are also known as globally unique identifiers (GUIDs) and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the UUID.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.rfc-editor.org/rfc/rfc2616.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.rfc-editor.org/rfc/rfc2818.txt

[RFC7230] Fielding, R., and Reschke, J., Eds., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014, http://www.rfc-editor.org/rfc/rfc7230.txt

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 8259, December 2017, https://www.rfc-editor.org/rfc/rfc8259.txt

[YAML1.2] Ben-Kiki, O., Evans, C., and dot NET, I., "YAML Ain't Markup Language (YAML) Version 1.2", 3rd edition, October 2009, https://yaml.org/spec/1.2/spec.html

1.2.2 Informative References

[Kubernetes] The Kubernetes Authors, "Kubernetes Documentation", version 1.14, https://kubernetes.io/docs/home/

1.3 Overview

The **Control Plane** acts as an abstraction layer that allows users to create and manage **big data clusters** without needing to communicate with **Kubernetes** or the services deployed within it directly. It provides convenient APIs to allow the user to manage the lifecycle of resources deployed in the cluster.

The protocol specifies an API for interacting with a Control Plane deployed in a **Kubernetes cluster**. A Control Plane manages big data cluster authentication and security and provides monitoring tools to observe the state of the big data cluster. The API additionally handles the management of the lifetime of resources within a cluster.

The protocol is constructed to be a RESTful API. All communications are initiated in **JavaScript Object Notation (JSON)** format by the client, and the server responds in JSON format to the client requests, as specified in [RFC8259].

The protocol uses **Basic** authentication for all requests unless specified otherwise.

The protocol allows users to do the following:

- Manage the lifecycle of a big data cluster.
- Manage the lifecycle of machine learning **applications** deployed in a big data cluster.
- Manage the lifecycle of **Hadoop Distributed File System (HDFS)** mounts mounted remotely.

The Control Plane is deployed on a Kubernetes [Kubernetes] cluster. It consists of a controller **replica set** and a **management proxy**, as well as various other **pod**s that provide log and metrics collection for pods in the cluster.

The Control Plane allows users to deploy a big data cluster in their Kubernetes cluster. Depending on the configuration sent to the Control Plane REST API, the user can customize the topography of the cluster.

All requests are initiated by the client. The server responds to client requests. Server responses are provided in JSON format, as illustrated in the following diagram.

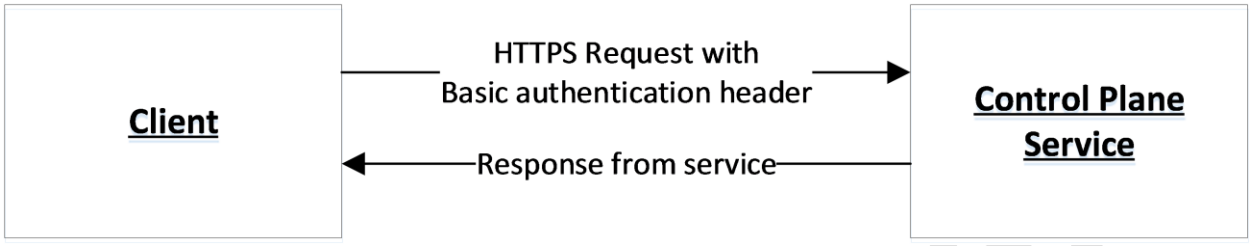

Figure 1: Communication flow

1.4 Relationship to Other Protocols

The Control Plane REST API protocol transmits messages by using HTTPS [RFC7230] [RFC2818].

The following diagram shows the protocol layering.

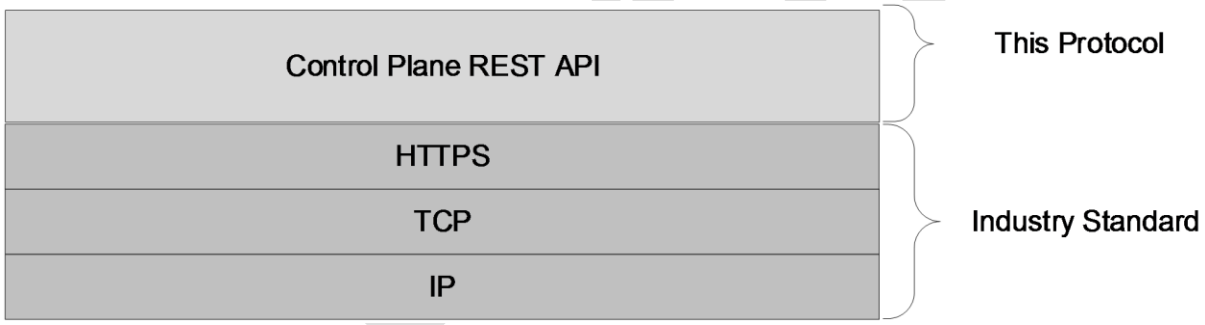

Figure 2: Protocol layering

1.5 Prerequisites/Preconditions

A controller replica set must be deployed in the Kubernetes cluster before the Control Plane REST API can be used. The controller is deployed by using Kubernetes APIs.

1.6 Applicability Statement

This protocol supports exchanging messages between a client and the Control Plane service.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The **Control Plane** protocol consists of a set of RESTful web services, and HTTPS over TCP/IP, as specified in [RFC2616]. All client messages MUST use HTTPS.

The management service is granted permission by the cluster administrator to manage all resources within the cluster, including but not limited to authentication. Implementers can configure their servers to use standard authentication, such as HTTP **Basic**.

The protocol does not require any specific HTTP ports, character sets, or transfer encoding.

2.2 Common Data Types

2.2.1 Namespaces

None.

2.2.2 HTTP Methods

This protocol uses HTTP methods GET, POST, and DELETE.

2.2.3 HTTP Headers

This protocol defines the following common HTTP headers in addition to the existing set of standard HTTP headers.

HTTP headers	Description
X-RequestID	An optional **UUID** that can be included to help map a request through the Control Plane service.

2.2.3.1 X-RequestId

A request to the Control Plane service can include an X-RequestID header that is included in all subsequent calls within the Controller Service. This can help with following a request through the Control Plane service logs.

2.2.4 URI Parameters

Every resource that supports CRUD operations uses common **JavaScript Object Notation (JSON)** elements in any request or response. The following table summarizes a set of common **URI** parameters defined by this specification.

This protocol defines the following common URI parameters.

URI parameters	Description
clusterIP	The IP address of a connectable node in the cluster.
controllerPort	The port that is exposed on the cluster for the controller. It is defined by the

URI parameters	Description
	user during Control Plane creation.
clusterName	The name of the **big data cluster** being manipulated.
managementProxyPort	The port that is exposed on the cluster for the management proxy. It is defined by the user during Control Plane creation.
appProxyPort	The port that is exposed on the cluster for the **app proxy**. It is defined by the user during Control Plane creation.

2.2.4.1 clusterIP

The *clusterIp* parameter contains the IP address of a node in the cluster that is accessible to the user. This is often the same address that tools, such as kubectl, use to connect to the cluster.

2.2.4.2 controllerPort

The *controllerPort* parameter is defined in the controller. This value is specified before controller deployment.

2.2.4.3 clusterName

The *clusterName* parameter provides the name of the deployed cluster. This name matches the **Kubernetes cluster** in which the cluster is to be deployed.

2.2.4.4 managementProxyPort

The *managementProxyPort* is the endpoint that is exposed by the Control Plane to provide access to the management proxy.

2.2.4.5 appProxyPort

The *appProxyport* is the endpoint that is exposed by the Control Plane to provide access to the **app proxy**.

2.2.5 JSON Elements

Data structures that are defined in this section flow through the protocol in JSON format and are defined in JSON schema.

This protocol defines the following common JSON schema elements. All elements are required unless specified otherwise in the Type field.

Element	Type	Description
metadata		Structured data that provides additional information about the JSON object.
metadata.kind		Structured data that describes the type of object that is to be created.
metadata.name		Structured data that provides the name of the component that is to be created.

Element	Type	Description
docker		Structured data that defines where to find a **docker** image.
docker.registry		Specifies the registry where a docker image is located.
docker.repository		Specifies the repository where a docker image is located.
docker.imageTag		Specifies the image tag for the image to be pulled.
docker.imagePullPolicy		Specifies the image pull policy for the docker image.
storage		Structured data that defines persistent storage to be used in the cluster.
storage.usePersistentVolume		A flag that specifies whether the deployment is to use **persistent volumes**.
storage.className		Specifies the **Kubernetes storage class** that is used to create persistent volumes.
storage.accessMode		Specifies the access mode for Kubernetes persistent volumes.
storage.size		Specifies the size of the persistent volume.
endpoints		Specifies an array of endpoints that is exposed for a component.
endpoint		Specifies an endpoint that is exposed outside of the cluster.
endpoint.name		Specifies the name of the endpoint that is exposed outside of the cluster.
endpoint.serviceType		Specifies the Kubernetes service type that exposes the endpoint port.
endpoint.Port		Specifies the port on which the service is exposed.
replicas		Specifies the number of Kubernetes **pods** to deploy for a component.
pool		The structured data that represents an organizational unit in the cluster.
pool.metadata		The statistical information that describes the pool.
pool.spec		Structured data that provides information on what the pool is to contain.
pool.spec.type		Specifies the type of pool that is being deployed. Valid values are the following:

Element	Type	Description
		▪ 0 ▪ 1 ▪ 2 ▪ 3 These values map to the following definitions: ▪ 0: Master Pool ▪ 1: **Compute Pool** ▪ 2: **Data Pool** ▪ 3: **Storage Pool**
pool.replicas		See replicas element definition.
pool.docker		See docker element definition.
pool.storage		See storage element definition.
pool.endpoints		See endpoints element definition.
pool.hadoop	Required if pool.spec.type is 0 or 3.	See **Hadoop** element definition.
pool.spark	Required if pool.spec.type is 0 or 3.	See **Apache Spark** element definition.
pool.namenode	Required if pool.spec.type is 3.	See **NameNode** element definition.
hadoop		Configuration settings for the Hadoop that is running in the cluster.
hadoop.imageName		Docker image bits that are necessary to run Hadoop.
hadoop.yarn		Structured data that describes the configuration for **Apache YARN** that is used by Hadoop.
hadoop.yarn.nodeManager		Structured data that describes settings for the node manager.
hadoop.yarn.nodeManager.memory		Maximum amount of memory available to the node manager (in MB).
hadoop.yarn.nodeManager.vcores		Specifies the number of virtual cores to allocate to the node manager.
hadoop.yarn.schedulerMax		Structured data that describes settings for the Apache YARN scheduler.
hadoop.yarn.schedulerMax.memory		Maximum number of MB of memory available to the Apache YARN scheduler.
hadoop.yarn.capacityScheduler		Structured data that describes settings for the Apache YARN scheduler's capacity.
hadoop.yarn.capacityScheduler.maxAmPercent		Specifies the maximum percentage of resources available to be used by the

Element	Type	Description
		Apache YARN Scheduler.
hadoop.spark		The collection of data that describe settings for Spark in the cluster.
hadoop.spark.driverMemory		Specifies the amount of memory that is available to the Spark Driver.
hadoop.spark.driverCores		Specifies the number of cores that are available to the Spark Driver.
hadoop.spark.executorInstances		Describes the number of executors usable by the Spark Driver.
hadoop.spark.executorMemory		Describes the maximum number of bytes available to each Spark Executor.
hadoop.spark.executorCores		Specifies the maximum number of cores that are available to each Spark Executor.

3 Protocol Details

3.1 Common Details

If an HTTP operation is unsuccessful, the server MUST return the error as JSON content in the response. The format of the **JavaScript Object Notation (JSON)** response is described in section 3.1.5.1.1.2.

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation can maintain to participate in this protocol. The organization is provided to help explain how the protocol works. This document does not require that implementations of the **Control Plane** REST API protocol adhere to this model, provided the external behavior of the implementation is consistent with that specified in this document.

The following resources are managed by using this protocol:

- cluster (section 3.1.5.1)
- storage (section 3.1.5.2)
- app (section 3.1.5.3)
- token (section 3.1.5.4)

3.1.2 Timers

None.

3.1.3 Initialization

For a client to utilize this protocol, the client MUST have a healthy Control Plane that is running in a **Kubernetes cluster**.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The following resources can be created and managed by using the Control Plane REST API.

Resource	Section	Description
cluster	3.1.5.1	Represents a **big data cluster** that is deployed in the Kubernetes cluster.
storage	3.1.5.2	The external mounts that are mounted in the big data cluster's **HDFS** instance.
app	3.1.5.3	The standalone Python or R scripts that are deployed in the big data cluster.
token	3.1.5.4	A token that can be included as a header for application calls.

The URL of the message that invokes the resource is formed by a concatenation of the following components:

- The absolute URI to the Controller Service.
- A string that represents the endpoint to be accessed.
- The remainder of the desired HTTP URL as described in this document.

Requests require a **Basic** authentication header to be attached to the request. For example, to retrieve the state of a currently deployed cluster that is named "test", the following request would be made:

```
curl -k -u admin:<adminPassword> --header "X-RequestID: 72b674f3-9288-42c6-a47b-
     948011f15010" https://<clusterIp>:<controllerPort>/clusters/test
```

adminPassword: The cluster admin password that was set up during Control Plane setup.

-k: The parameter that is required because the cluster currently uses self-signed certificates. For more information, see section 5.1.

--header: The parameter that adds the **X-RequestID** header to the request.

3.1.5.1 cluster

A **cluster** resource represents the highest-level object in a big data cluster and is deployed in a Kubernetes cluster in a **Kubernetes namespace** of the same name.

The **cluster** resource is invoked by using the following **URI**.

```
https://<cluster-ip>:<controller-port>/clusters
```

The following HTTP methods can be performed on this resource.

HTTP method	Section	Description
CREATE cluster	3.1.5.1.1	Create a **cluster**.
DELETE	3.1.5.1.2	Delete a **cluster**.
GET Logs	3.1.5.1.3	Retrieve logs from a **cluster**.
GET State	3.1.5.1.4	Retrieve the state of a **cluster**.

The following property elements are valid. All property elements are required.

Element name	Description
apiVersion	The Kubernetes API version that is being used. The value of this element MUST be "v1".
metadata	See section 2.2.5.
spec	Structured data that defines what to deploy in the big data cluster.
spec.controlPlane	Structured data that defines what to deploy in the Control Plane.
spec.controlPlane.secrets	Structured data that defines secrets to be mounted in the big data cluster.

Element name	Description
spec.controlPlane.secrets.mssqlSaPassword	Defines the sa password for the big data cluster **master instance**. This should be a base64 encoded string of the password defined during cluster creation.
spec.controlPlane.mssqlClusterAdminPassword	Defines the password for the cluster's admin user account. This should be a base64 encoded string of the password defined during cluster creation.
spec.controlPlane.spec.docker	See section 2.2.5.
spec.controlPlane.spec.endpoints	An array of **endpoints**. See section 2.2.5.
spec.controlPlane.spec.storage	See section 2.2.5.
spec.controlPlane.zookeeper	Defines the configuration settings for the **ZooKeeper** element.
spec.controlPlane.zookeeper.replicas	Defines the number of **ZooKeeper** replicas to deploy in the cluster.
spec.controlPlane.zookeeper.storage	Defines the storage setting to use for the **ZooKeeper** element.
spec.pools	Specifies an array of pools that are to be deployed in the cluster. See section 2.2.5.

3.1.5.1.1 CREATE

The CREATE method creates a **big data cluster** in the **Kubernetes cluster**.

This method is invoked by sending a POST operation to the following URI:

```
https://{clusterIp}:{controllerPort}/clusters
```

The response message for the CREATE method can result one or more of the following status codes.

Status code	Description
200	The Cluster spec was accepted, and cluster creation has been initiated.
400	The Control Plane services failed to parse the cluster spec.
400	A cluster with the provided name already exists.
500	An unexpected error occurred while parsing the cluster spec.
500	An internal error occurred while firing the create event for the cluster.
500	The operation failed to store the **data pool** node list in metadata storage.
500	The operation failed to store the **storage pool** node list in metadata storage.

The state of the cluster is retrieved by using the GET State method as specified in section 3.1.5.1.4.

3.1.5.1.1.1 Request Body

The request body is a JSON object in the following format.

```
{
  "apiVersion": "v1",
  "metadata": {
    "kind": "Cluster",
    "name": "cluster"
  },
  "spec": {
    "controlPlane": {
      "secrets": {
        "mssqlSaPassword": "Zm9vYmFy",
        "mssqlClusterAdminPassword": "Zm9vYmFy"
      },
      "spec": {
        "docker": {
          "registry": "private-repo.microsoft.com",
          "repository": "mssql-private-preview",
          "imageTag": "latest",
          "imagePullPolicy": "IfNotPresent"
        },
        "storage": {
          "usePersistentVolume": true,
          "className": "local-storage",
          "accessMode": "ReadWriteOnce",
          "size": "10Gi"
        },
        "endpoints": [
          {
            "name": "Controller",
            "serviceType": "LoadBalancer",
            "port": 30080
          },
          {
            "name": "ServiceProxy",
            "serviceType": "LoadBalancer",
            "port": 30777
          },
          {
            "name": "AppServiceProxy",
            "serviceType": "LoadBalancer",
            "port": 30778
          },
          {
            "name": "Knox",
            "serviceType": "LoadBalancer",
            "port": 30443
          }
        ]
      },
      "zookeeper": {
        "replicas": 0,
        "storage": {
          "usePersistentVolume": true,
          "className": "local-storage",
          "accessMode": "ReadWriteOnce",
          "size": "10Gi"
        }
      }
    },
    "pools": [
      {
        "metadata": {
          "kind": "Pool",
          "name": "default"
        },
        "spec": {
          "type": "Master",
          "replicas": 1,
          "docker": {
            "registry": "private-repo.microsoft.com",
            "repository": "mssql-private-preview",
```

```
        "imageTag": "latest",
        "imagePullPolicy": "IfNotPresent"
      },
      "storage": {
        "usePersistentVolume": true,
        "className": "local-storage",
        "accessMode": "ReadWriteOnce",
        "size": "10Gi"
      },
      "endpoints": [
        {
          "name": "Master",
          "serviceType": "LoadBalancer",
          "port": 31433
        }
      ]
    },
    "hadoop": {
      "imageName": "mssql-hadoop ",
      "yarn": {
        "nodeManager": {
          "memory": 18432,
          "vcores": 6
        },
        "schedulerMax": {
          "memory": 18432,
          "vcores": 6
        },
        "capacityScheduler": {
          "maxAmPercent": 0.3
        }
      },
      "spark": {
        "driverMemory": "2g",
        "driverCores": 1,
        "executorInstances": 3,
        "executorMemory": "1536m",
        "executorCores": 1
      }
    }
  },
  {
    "metadata": {
      "kind": "Pool",
      "name": "default"
    },
    "spec": {
      "type": "Compute",
      "replicas": 1,
      "docker": {
        "registry": "private-repo.microsoft.com",
        "repository": "mssql-private-preview",
        "imageTag": "latest",
        "imagePullPolicy": "IfNotPresent"
      },
      "storage": {
        "usePersistentVolume": true,
        "className": "local-storage",
        "accessMode": "ReadWriteOnce",
        "size": "10Gi"
      }
    }
  },
  {
    "metadata": {
      "kind": "Pool",
      "name": "default"
    },
    "spec": {
      "type": "Data",
```

```
      "replicas": 2,
      "docker": {
        "registry": "private-repo.microsoft.com",
        "repository": "mssql-private-preview",
        "imageTag": " latest ",
        "imagePullPolicy": "IfNotPresent"
      },
      "storage": {
        "usePersistentVolume": true,
        "className": "local-storage",
        "accessMode": "ReadWriteOnce",
        "size": "10Gi"
      }
    }
  },
  {
    "metadata": {
      "kind": "Pool",
      "name": "default"
    },
    "spec": {
      "type": "Storage",
      "replicas": 2,
      "docker": {
        "registry": "private-repo.microsoft.com",
        "repository": "mssql-private-preview ",
        "imageTag": "latest",
        "imagePullPolicy": "IfNotPresent"
      },
      "storage": {
        "usePersistentVolume": true,
        "className": "local-storage",
        "accessMode": "ReadWriteOnce",
        "size": "10Gi"
      }
    },
    "namenode": {
      "imageName": "mssql-hadoop",
      "replicas": 1,
      "spec": {
        "storage": {
          "usePersistentVolume": true,
          "className": "local-storage",
          "accessMode": "ReadWriteOnce",
          "size": "10Gi"
        }
      }
    },
    "hadoop": {
      "imageName": "mssql-hadoop",
      "yarn": {
        "nodeManager": {
          "memory": 18432,
          "vcores": 6
        },
        "schedulerMax": {
          "memory": 18432,
          "vcores": 6
        },
        "capacityScheduler": {
          "maxAmPercent": 0.3
        }
      },
      "spark": {
        "driverMemory": "2g",
        "driverCores": 1,
        "executorInstances": 3,
        "executorMemory": "1536m",
        "executorCores": 1
      }
```

```
            }
          }
        ]
      }
    }
```

The JSON schema for the Cluster method is presented in section 6.1.1.

3.1.5.1.1.2 Response Body

If the request is successful, there is no response body.

If the request fails, a JSON object of the following format is returned.

```
{
  "code": 500,
  "reason": "An unexpected exception occurred.",
  "data": "Null reference exception"
}
```

The JSON schema for the response body is presented in section 6.1.2.

3.1.5.1.1.3 Processing Details

This method creates a new **cluster** resource.

3.1.5.1.2 DELETE

This method deletes a **cluster** resource.

It is invoked by sending a DELETE operation to the following URI.

```
https://{clusterIp}:{controllerPort}/clusters/{clusterName}
```

The response message for this method can result in one of the following status codes.

Status code	Description
200	Cluster deletion was initiated.
500	Cluster deletion failed due to an internal error.

3.1.5.1.2.1 Request Body

The request body is empty. There are no parameters.

3.1.5.1.2.2 Response Body

The response body is empty.

3.1.5.1.2.3 Processing Details

This method deletes a **cluster** resource.

3.1.5.1.3 GET Logs

This method retrieves the logs from a **cluster**.

This method is invoked by sending a GET to the following URI.

```
https://{clusterIp}:{controllerPort}/clusters/{clusterName}/log?offset={offsetNumber}
```

offset: A parameter that allows a partial log to be returned. If the value of **offset** is 0, the whole log is returned. If the value of **offset** is non-zero, the log that is returned starts at the byte located at the offset value.

The response message for this method can result in the following status code.

Status code	Description
200	The logs are successfully returned.

3.1.5.1.3.1 Request Body

The request body is empty.

3.1.5.1.3.2 Response Body

The response body is the contents of the log file. The log starts with the offset value and continues to the end of the log.

3.1.5.1.3.3 Processing Details

The client is responsible for tracking the offset into the file when a partial log is retrieved. To do so, the client adds the previous offset to the length of the log returned. This value represents the new offset value.

3.1.5.1.4 GET State

This method retrieves the status of a **cluster** resource.

It is invoked by sending a GET operation to the following URI:

```
https://{clusterIp}:{controllerPort}/clusters/{clusterName}/status
```

The response message for this method can result in one of the following status codes.

Status code	Description
200	**Cluster** status returned successfully.
404	The **cluster** specified by **clusterName** does not exist.
500	The operation failed to get a state for the **cluster** that is specified by **clusterName**.

3.1.5.1.4.1 Request Body

The request body is empty.

3.1.5.1.4.2 Response Body

The response body is a JSON object of the following format:

```
{
    "state": "Ready",section
}
```

The JSON schema for this response can be found in section 6.1.3.

3.1.5.1.4.3 Processing Details

None.

3.1.5.2 storage

The **storage** resource specifies a remote file system that is mounted to a path in the cluster's local **HDFS**.

It is invoked by using the following **URI**.

```
https://{clusterIp}:{controllerPort}/storage/mounts
```

Resource	Section	Description
GET	3.1.5.2.1	Retrieve the status of a mount.
Get all mount statuses	3.1.5.2.2	Retrieve the status of all mounts in the cluster.
Create mount	3.1.5.2.3	Create a mount.
Delete mount	3.1.5.2.4	Delete a mount.

The following property elements are valid.

Property Name	Description
mount	The path of the HDFS mount.
remote	The HDFS mount point to attach the mount to.
state	The state of the HDFS mount deployment.
error	This field is populated only if the mount is unhealthy.

3.1.5.2.1 Get Mount Status

This method is used to retrieve the status of one or more HDFS mounts in the cluster.

It is invoked by sending a GET operation to the following URI:

```
https://{clusterIp}:{controllerPort}/storage/mounts/{mountPath}
```

mountPath: The directory of the mount.

The response message for this method can result in the following status codes.

Status code	Description
200	The mount is returned successfully.
404	The mount specified by mountName does not exist.

3.1.5.2.1.1 Request Body

The request body is empty.

3.1.5.2.1.2 Response Body

If a mountPath is specified, the response body contains a response in a JSON object of the following format.

```
{
    "mount": "/mnt/test",
    "remote": "abfs://foo.bar",
    "state": "Ready",
    "error": ""
}
```

The full JSON schema for the response can be found in section 6.2.1.

3.1.5.2.1.3 Processing Details

This method is used to retrieve the status of one or more HDFS **mounts** in the cluster.

3.1.5.2.2 Get All Mount Statuses

This method is used to retrieve the status of all HDFS **mounts** in the cluster.

It is invoked by sending a GET operation to the following URI:

```
https://{clusterIp}:{controllerPort}/storage/mounts/
```

The response message for this method can result in the following status codes.

Status code	Description
200	The mount is returned successfully.

3.1.5.2.2.1 Request Body

The request body is empty.

3.1.5.2.2.2 Response Body

The response body contains a response in JSON that contains an array of JSON objects in the format that is specified in section 3.1.5.2.1.2.

3.1.5.2.2.3 Processing Details

This method is used to retrieve the status of all HDFS **mounts** in the cluster.

3.1.5.2.3 Create Mount

This method creates an HDFS mount within the cluster.

It is invoked by sending a POST operation to the following URI.

```
https://{clusterIp}:{controllerPort}/storage/mounts?remote{remote}&mount{mount}
```

remote: The URI of the store to mount.

mount: the local HDFS path for the mount point.

The response message for this message can result in the following status codes.

Status code	Description
202	Mount creation was successfully initiated.
400	The specified mount already exists.
500	An internal error occurred while firing the create event for the specified mount.
500	An unexpected error occurred while processing the mount credentials.

3.1.5.2.3.1 Request Body

The request body should be a request in JSON where each element corresponds to an authentication property needed to access the remote file system. The authentication properties required vary from provider to provider.

3.1.5.2.3.2 Response Body

The response body is empty.

3.1.5.2.3.3 Processing Details

The client should use the GET method to monitor the creation of the mount.

3.1.5.2.4 Delete Mount

This method deletes a currently mounted HDFS mount.

It is invoked by sending a DELETE operation to the following URI:

```
https://{clusterIp}:{controllerPort}/storage/mounts?mount={mount}
```

mount: Mount point to delete.

The response message for this method can result in the following status codes.

Status code	Description
202	The delete request was accepted.
400	The delete request is invalid.
400	The specified mount does not exist.
500	The method failed to delete specified mount.

3.1.5.2.4.1 Request Body

The request body is empty.

3.1.5.2.4.2 Response Body

On an unsuccessful request, the response body contains a response in JSON of the type **Response**.

3.1.5.2.4.3 Processing Details

The client should use the Get Mount Status API to monitor the deletion of the mount.

3.1.5.3 App

The **app** resource specifies an R or Python script that can be deployed or is deployed in the **cluster**.

It is invoked through the following URIs.

```
https://{clusterIp}:{managementProxyPort}/api/app

https://{clusterIp}:{appProxyPort}/api/app
```

The following HTTP methods can be performed on this resource.

Resource	Section	Description
GET app	3.1.5.3.1	Retrieve the status of the application.
GET Application Versions	3.1.5.3.2	Retrieve the status of all deployed **applications**.
GET All Applications	3.1.5.3.3	Retrieve the status of one or more deployed applications.
CREATE app	3.1.5.3.4	Deploy a deployed application.
Update App	3.1.5.3.5	Update a deployed application.
Delete app	3.1.5.3.6	Delete a deployed application.
Run app	3.1.5.3.7	Send inputs to a deployed application.

The following property elements are valid.

Property Name	Description
name	Name of the application that is being deployed.
internal_name	Name for the application that is used internally within the cluster.
state	State of the application's deployment. Valid values are the following: ▪ "Initial" ▪ "Creating" ▪ "Updating" ▪ "WaitingForUpdate" ▪ "Ready" ▪ "Deleting" ▪ "WaitingForDelete" ▪ "Deleted" ▪ "Error"

Property Name	Description
version	Version of the app being deployed.
input_param_defs	Array of parameters that represent the inputs that can be passed to the application.
parameter	Structured data representing an app parameter. A parameter consists of a **name** and a **type**.
parameter.name	Name of the parameter.
parameter.type	Type of parameter. Valid values are the following: ▪ "str" ▪ "int" ▪ "dataframe" ▪ "data.frame" ▪ "float", ▪ "Matrix" ▪ "vector" ▪ "bool"
output_param_defs	Array of parameters that represent the outputs of the application.
links	Array of **links**.
link	Structured data that represents a URL that can be used to access the deployed application.
link.app	An endpoint to access to deployed application.
link.swagger	An endpoint to a Swagger editor. The editor can be used to directly send requests to the deployed application.
success	Describes whether an application method succeeded.
errorMessage	Describes the reason an appliction method failed.
outputParameters	List of output parameters that resulted from the method. See **output_parameter_defs**.
outputFiles	Array of file names that resulted from the application operation.
consoleOutput	Describes the text output that resulted from the application method.
changedFiles	Array of file names that were modified from the application operation.

3.1.5.3.1 GET app

This method returns a description of a deployed application with the given name and version.

This method is invoked by sending a GET operation to the following URI.

```
https://{clusterIp}:{serviceProxyPort}/api/app/{name}/{version}
```

name: The name of the deployed application.

version: The specific application version to get the status of.

The response message for this method can result in the following status codes.

Status code	Description
200	Success.
404	The application cannot be found.

3.1.5.3.1.1 Request Body

The request body is empty.

3.1.5.3.1.2 Response Body

```
{
  "name": "hello-py",
  "internal_name": "app1",
  "version": "v1",
  "input_param_defs": [
    {
      "name": "msg",
      "type": "str"
    },
    {
      "name": "foo",
      "type": "int"
    }
  ],
  "output_param_defs": [
    {
      "name": "out",
      "type": "str"
    }
  ],
  "state": "Ready",
  "links": {
    "app": "https://10.127.22.96:30777/api/app/hello-py/v1",
    "swagger": "https://10.127.22.96:30777/api/app/hello-py/v1/swagger.json"
  }
}
```

The JSON schema for the response can be found in section 6.3.1

3.1.5.3.1.3 Processing Details

This method returns a list of statuses for all **applications** of the specified name.

3.1.5.3.2 GET Application Versions

This method lists all versions of a deployed **app**.

This method is invoked by sending a GET operation to the following URI.

```
https://{clusterIp}:{serviceProxyPort}/api/app/{name}
```

The response message for this method can result in the following status codes.

Status code	Description
200	Success.

Status code	Description
404	The application cannot be found.

3.1.5.3.2.1 Request Body

The request body is empty.

3.1.5.3.2.2 Response Body

The response body contains a response in JSON format that contains an array of **app** descriptions as specified in section 3.1.5.3.1.1.

3.1.5.3.2.3 Processing Details

This method returns the status of all versions of a specific **app**.

3.1.5.3.3 GET All Applications

This method is invoked by calling a GET operation to the following URI.

```
https://{clusterIp}:{serviceProxyPort}/api/app
```

The response message for this method can result in the following status codes.

Status code	Description
200	The statuses of the applications was retrieved successfully.

3.1.5.3.3.1 Request Body

The request body is empty.

3.1.5.3.3.2 Response Body

The response body contains a response in JSON that contains an array of descriptions for all applications that are deployed in the cluster, as specified in section 3.1.5.3.1.2.

3.1.5.3.3.3 Processing Details

This method returns the description of all **apps** deployed in the **cluster**.

3.1.5.3.4 CREATE app

This method is used to create an **app** in the cluster.

This method is invoked by sending a POST operation to the following URI:

```
https://{clusterIp}:{serviceProxyPort}/api/app
```

The response message for this method can result in the following status codes.

Status code	Description
201	Created. App status available through location header link.
400	The request is invalid.
409	An app with the specified version already exists.

3.1.5.3.4.1 Request Body

The request body contains a ZIP file that has been stored. The ZIP file contains a specification with the filename "spec.yaml" that is written in YAML [YAML1.2] as well as the Python or R script that the user is deploying.

3.1.5.3.4.2 Response Body

The response body is empty.

3.1.5.3.4.3 Processing Details

This method is used to create an **app** in the cluster.

3.1.5.3.5 UPDATE

This method is used to update an already deployed **app**.

This method is invoked by sending a PATCH operation to the following URI.

```
https://{clusterIp}:{serviceProxyPort}/api/app
```

The response message for this method can result in the following status codes.

Status code	Description
201	The application was updated. The update status is available by using a GET operation.
400	The request is invalid.
404	The specified application cannot be found.

3.1.5.3.5.1 Request Body

The request body contains a ZIP file that has been stored. The ZIP file contains a specification with the filename "spec.yaml" that is written in YAML [YAML1.2] as well as the updated Python or R script that the user is deploying.

3.1.5.3.5.2 Response Body

The response body is empty.

3.1.5.3.5.3 Processing Details

This method is used to update an already deployed **app**.

3.1.5.3.6 DELETE

This method is used to delete an **app** resource in the **cluster**.

This method can be invoked by sending a DELETE operation to the following URI:

```
https://{clusterIp}:{serviceProxyPort}/api/app/{name}/{version}
```

The response message for this method can result in the following status codes.

Status code	Description
202	The request is accepted. The application will be deleted.
404	The specified application cannot be found.

3.1.5.3.6.1 Request Body

The request body is empty.

3.1.5.3.6.2 Response Body

The response body is empty.

3.1.5.3.6.3 Processing Details

This method is used to delete an **app** resource in the **cluster**.

3.1.5.3.7 RUN app

This method is used to send a request to a deployed **app**.

This method can be invoked by sending a RUN operation to the following URI.

```
https://{clusterIp}:{appProxyPort}/api/app/{name}/{version}/run
```

The response message for this method can result in the following status codes.

Status code	Description
202	Accepted. The **app** will be deleted.
404	The **app** cannot be found.

3.1.5.3.7.1 Request Header

The request MUST use **Bearer** authentication. This is done by including an **Authorization** HTTP header that contains a Bearer token. The header should look like the following.

```
'Authorization: Bearer <token>'
```

token: The token string that is returned when a token is retrieved. For more information, see section 3.1.5.4.1.

3.1.5.3.7.2 Request Body

The request body should contain a JSON of the following format.

```
{
  “x”:5,
  “y”: 37
}
```

The elements in this JSON object match the names and types that are described in **appModel.input_params_defs**.

3.1.5.3.7.3 Response Body

The response is a JSON of the following format.

```
{
  "success": true,
  "errorMessage": "",
  "outputParameters": {
    "result": 42
  },
    "outputFiles": {},
    "consoleOutput": "",
    "changedFiles": []
}
```

The full schema definition can be found in section 6.3.2.

3.1.5.3.7.4 Processing Details

This method is used to delete an **app** resource in the cluster.

3.1.5.4 token

A **token** resource is a **JWT** token that can be used as a form authentication to use app.

It can be invoked through the following **URI**.

```
https://{clusterIp}:{controllerPort}/token
```

The following HTTP methods can be performed on this resource.

HTTP method	Section	Description
Create token	3.1.5.4.1	Create and retrieve a **token**.

The following property elements are valid. All elements are required.

Element name	Description
token_type	Token type returned MUST be “Bearer”.
access_token	JWT token generated for the request.
expires_in	Number of seconds the token is valid for after being issued.
expires_on	The date on which the token expires. The date is based on the number of seconds since the Unix Epoch.
token_id	Unique ID that was generated for the token request.

3.1.5.4.1 CREATE

This method is used to create a **JWT** Bearer **token**.

This method can be invoked by sending a POST operation to the following URI.

```
https://{clusterIp}:{controllerPort}/token
```

The response message for this method can result in the following status codes.

Status code	Description
200	Token created.
400	The request is invalid.

3.1.5.4.1.1 Request Body

The request body is empty.

3.1.5.4.1.2 Response Body

The response will be a JSON of the following format.

```
{
    "token_type": "Bearer",
    "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjpbImFwcCIsImNvbnRyb2xsZXIiLCJtZXRhZGF0YSJdLCJ
uYmYiOjE1NTQ5MTM0MjIsImV4cCI6MTU1NDk0OTQyMSwiaWF0IjoxNTU0OTEzNDIyLCJpc3MiOiJtc3NxbC5taWNyb3Nv
ZnQuY29tIiwiYXVkIjoibXNzcWwubWljcm9zb2Z0LmNvbSJ9.qKTG4PsGxDDFbjnZnE__3NWxEqCS9X9kc9B9IpR_UTY"
,
    "expires_in": 36000,
    "expires_on": 1554949422,
    "token_id":
"YsaMFgi1Re72fyfd7dZz6twfgjCy7jb49h1IVKkHMZt0QpqO7noNte6Veu0x8h3PD7msPDiR9z9drWyJvZQ6MPWD0wNz
mRrvCQ+v7dNQV8+9e9N4gZ7iE5vDP6z9hBgrggh8w4FeVSwCYYZiOG67OTzF2cnCfhQ8Gs+AjJWso3ga5lHqIKv34JNgO
ONp5Vpbu5iHGffZepgZ4jaIDIVd3ByogHtq+/c5pjdwLwoxH47Xuik0wNLLwiqktAWOv1cxDXOivkaGbJ6FDtJR4tPuNg
RLjNuz9iAZ16osNDyJ7oKyecnt4Tbt+XerwlyYYrjDWcW92qtpHX+kWnDrnmRn1g=="
}
```

The full schema definition can be found in section 6.4.1.

3.1.5.4.1.3 Processing Details

This method is used to create a JWT Bearer **token**.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Cluster Admin Details

The client role of this protocol is simply a pass-through and requires no additional timers or other state. Calls made by the higher-layer protocol or application are passed directly to the transport, and

the results returned by the transport are passed directly back to the higher-layer protocol or application.

4 Protocol Examples

5 Security

5.1 Security Considerations for Implementers

Unless specified otherwise, all authentication is done by way of **Basic** authentication.

The Control Plane Rest API uses self-signed certificates. A user of this protocol needs to skip certificate verification when sending HTTP operations.

5.2 Index of Security Parameters

None.

6 Appendix A: Full JSON Schema

For ease of implementation, the following sections provide the full JSON schema for this protocol.

Schema name	Section
cluster	6.1
storage	6.2
app	6.3
token	6.4

6.1 cluster

6.1.1 cluster Spec Schema

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "title": "PUT Cluster schema",
  "definitions": {
    "storage": {
      "required": [
        "usePersistentVolume",
        "className",
        "accessMode",
        "size"
      ],
      "properties": {
        "usePersistentVolume": {
          "type": "boolean",
        },
        "className": {
          "type": "string",
        },
        "accessMode": {
          "enum": [
            "ReadWriteOnce",
            "ReadOnlyMany",
            "ReadWriteMany"
          ],
        },
        "size": {
          "type": "string",
          "example": "10Gi",
        }
      }
    },
    "docker": {
      "required": [
        "registry",
        "repository",
        "imageTag",
        "imagePullPolicy"
      ],
      "properties": {
        "registry": {
          "type": "string",
          "example": "repo.microsoft.com",
        },
        "repository": {
```

```
        "type": "string",
      },
      "imageTag": {
        "type": "string",
        "example": "latest",
      },
      "imagePullPolicy": {
        "enum": [
          "Always",
          "IfNotPresent"
        ],
      }
    }
  },
  "yarn": {
    "required": [
      "nodeManager",
      "schedulerMax",
      "capacityScheduler"
    ],
    "properties": {
      "nodeManager": {
        "required": [
          "memory",
          "vcores"
        ],
        "properties": {
          "memory": {
            "type": "integer",
          },
          "vcores": {
            "type": "integer",
          }
        }
      },
      "schedulerMax": {
        "required": [
          "memory",
          "vcores"
        ],
        "properties": {
          "memory": {
            "type": "integer",
          },
          "vcores": {
            "type": "integer",
          }
        }
      },
      "capacityScheduler": {
        "required": [
          "maxAmPercent"
        ],
        "properties": {
          "maxAmPercent": {
            "type": "number",
          }
        }
      }
    }
  },
  "imageName": {
    "type": "string",
  },
  "namenode": {
    "required": [
      "imageName",
      "replicas",
      "spec"
    ],
```

```
    "properties": {
      "imageName": {
        "$ref": "#/definitions/imageName"
      },
      "replicas": {
        "$ref": "#/definitions/replicas"
      },
      "spec": {
        "required": [
          "storage"
        ],
        "properties": {
          "storage": {
            "$ref": "#/definitions/storage"
          }
        }
      }
    }
  },
  "hadoop": {
    "required": [
      "imageName",
      "yarn",
      "spark"
    ],
    "properties": {
      "imageName": {
        "type": "string"
      },
      "yarn": {
        "$ref": "#/definitions/yarn"
      },
      "spark": {
        "$ref": "#/definitions/spark"
      }
    }
  },
  "spark": {
    "properties": {
      "driverMemory": {
        "type": "string",
        "example": "2g"
      },
      "driverCores": {
        "type": "integer"
      },
      "executorInstances": {
        "type": "integer"
      },
      "executorMemory": {
        "type": "string",
        "example": "1536m"
      },
      "executorCores": {
        "type": "integer"
      }
    }
  },
  "metadata": {
    "required": [
      "kind",
      "name"
    ],
    "properties": {
      "kind": {
        "type": "string"
      },
      "name": {
        "name": "string"
      }
```

```
        }
      },
      "endpoint": {
        "required": [
          "name",
          "serviceType",
          "port"
        ],
        "properties": {
          "name": {
            "type": "string",
          },
          "serviceType": {
            "enum": [
              "LoadBalancer",
              "NodePort"
            ],
          },
          "port": {
            "type": "integer",
          }
        }
      },
      "endpoints": {
        "type": "array",
        "title": "The Endpoints Schema",
        "items": {
          "$ref": "#/definitions/endpoint"
        },
      },
      "replicas": {
        "type": "integer",
      },
      "masterPool": {
ss      "required": [
          "metadata",
          "spec",
          "hadoop"
        ],
        "properties": {
          "metadata": {
            "$ref": "#/definitions/metadata"
          },
          "spec": {
            "required": [
              "type",
              "replicas",
              "docker",
              "storage",
              "endpoints"
            ],
            "properties": {
              "type": {
                "const": "Master"
              },
              "replicas": {
                "$ref": "#/definitions/replicas"
              },
              "docker": {
                "$ref": "#/definitions/docker"
              },
              "storage": {
                "$ref": "#/definitions/storage"
              },
              "endpoints": {
                "$ref": "#/definitions/endpoints"
              }
            }
          },
          "hadoop": {
```

```
        "$ref": "#/definitions/hadoop"
      }
    }
  },
  "storagePool": {
    "required": [
      "metadata",
      "spec",
      "namenode",
      "hadoop"
    ],
    "properties": {
      "metadata": {
        "$ref": "#/definitions/metadata"
      },
      "spec": {
        "required": [
          "type",
          "replicas",
          "docker",
          "storage"
        ],
        "properties": {
          "type": {
            "const": "Storage"
          },
          "replicas": {
            "$ref": "#/definitions/replicas"
          },
          "docker": {
            "$ref": "#/definitions/docker"
          },
          "storage": {
            "$ref": "#/definitions/storage"
          }
        }
      },
      "namenode": {
        "$ref": "#/definitions/namenode"
      },
      "hadoop": {
        "$ref": "#/definitions/hadoop"
      }
    }
  },
  "dataPool": {
    "required": [
      "metadata",
      "spec"
    ],
    "properties": {
      "metadata": {
        "$ref": "#/definitions/metadata"
      },
      "spec": {
        "required": [
          "type",
          "replicas",
          "docker",
          "storage"
        ],
        "properties": {
          "type": {
            "const": "Data"
          },
          "replicas": {
            "$ref": "#/definitions/replicas"
          },
          "docker": {
            "$ref": "#/definitions/docker"
```

```
            },
            "storage": {
              "$ref": "#/definitions/storage"
            }
          }
        }
      }
    },
    "computePool": {
      "required": [
        "metadata",
        "spec"
      ],
      "properties": {
        "metadata": {
          "$ref": "#/definitions/metadata"
        },
        "spec": {
          "required": [
            "type",
            "replicas",
            "docker",
            "storage"
          ],
          "properties": {
            "type": {
              "const": "Compute"
            },
            "replicas": {
              "$ref": "#/definitions/replicas"
            },
            "docker": {
              "$ref": "#/definitions/docker"
            },
            "storage": {
              "$ref": "#/definitions/storage"
            }
          }
        },
        "namenode": {
          "$ref": "#/definitions/namenode"
        },
        "hadoop": {
          "$ref": "#/definitions/hadoop"
        }
      }
    }
  },
  "required": [
    "apiVersion",
    "metadata",
    "spec"
  ],
  "properties": {
    "apiVersion": {
      "$id": "#/properties/apiVersion",
      "type": "string",
      "const": "v1"
    },
    "metadata": {
      "$ref": "#/definitions/metadata"
    },
    "spec": {
      "$id": "#/properties/spec",
      "type": "object",
      "title": "The Spec Schema",
      "required": [
        "controlPlane",
        "pools"
      ],
```

```
    "properties": {
      "controlPlane": {
        "$id": "#/properties/spec/properties/controlPlane",
        "type": "object",
        "required": [
          "spec",
          "zookeeper"
        ],
        "properties": {
          "spec": {
            "$id": "#/properties/spec/properties/controlPlane/properties/spec",
            "type": "object",
            "required": [
              "docker",
              "storage",
              "endpoints"
            ],
            "properties": {
              "docker": {
                "$ref": "#/definitions/docker"
              },
              "storage": {
                "$ref": "#/definitions/storage"
              },
              "endpoints": {
                "$ref": "#/definitions/endpoints"
              }
            }
          },
          "zookeeper": {
            "$id": "#/properties/spec/properties/controlPlane/properties/zookeeper",
            "type": "object",
            "required": [
              "replicas",
              "storage"
            ],
            "properties": {
              "replicas": {
                "$ref": "#/definitions/replicas"
              },
              "storage": {
                "$ref": "#/definitions/storage"
              }
            }
          }
        }
      },
      "pools": {
        "$id": "#/properties/spec/properties/pools",
        "type": "array",
        "items": {
          "anyOf": [
            {
              "$ref": "#/definitions/masterPool"
            },
            {
              "$ref": "#/definitions/computePool"
            },
            {
              "$ref": "#/definitions/storagePool"
            },
            {
              "$ref": "#/definitions/dataPool"
            }
          ]
        }
      }
    }
  }
```

6.1.2 cluster Error Response Schema

```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "code",
    "reason",
    "data"
  ],
  "properties": {
    "code": {
      "$id": "#/properties/code",
      "type": "integer",
      "title": "The Code Schema",
      "default": 0,
      "examples": [
        500
      ]
    },
    "reason": {
      "$id": "#/properties/reason",
      "type": "string",
      "default": "",
      "examples": [
        "An unexpected exception occurred."
      ]
    },
    "data": {
      "$id": "#/properties/data",
      "type": "string",
      "default": "",
      "examples": [
        "Null reference exception"
      ]
    }
  }
}
```

6.1.3 cluster State Schema

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "title": "Cluster State Response Schema",
  "required": [
    "state"
  ],
  "properties": {
    "state": {
      "$id": "#/properties/state",
      "type": "string",
      "enum": [
        "Initial",
        "Creating",
        "Waiting",
        "Ready",
        "Deleting",
        "Deleted",
        "Error"
      ]
    }
  }
}
```

6.2 storage

6.2.1 storage Response Schema

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "title": "Storage Response Schema",
  "required": [
    "mount",
    "remote",
    "state",
    "error"
  ],
  "properties": {
    "mount": {
      "$id": "#/properties/mount",
      "type": "string",
    },
    "remote": {
      "$id": "#/properties/remote",
      "type": "string",
    },
    "state": {
      "$id": "#/properties/state",
        "enum": [
          "Initial",
          "Creating",
          "WaitingForCreate",
          "Updating",
          "WaitingForUpdate",
          "Ready",
          "Deleting",
          "WaitingForDelete",
          "Deleted",
          "Error"
        ]
    },
    "error": {
      "$id": "#/properties/error",
      "type": "string",
    }
  }
}
```

6.3 app

6.3.1 app Description Schema

```
{
  "definitions": {
    "link": {
      "type": "object",
      "properties": {
        "^.*$": {
          "type": "string"
        }
      }
    },
    "parameter": {
      "required": [
        "name",
        "type"
      ],
      "properties": {
        "name": {
          "type": "string"
```

```
      },
      "type": {
        "enum": [
          "str",
          "int",
          "dataframe",
          "data.frame",
          "float",
          "matrix",
          "vector",
          "bool"
        ]
      }
    }
  }
},
"$schema": "http://json-schema.org/draft-07/schema#",
"type": "array",
"title": "App Result Schema",
"items": {
  "$id": "#/items",
  "type": "object",
  "required": [
    "name",
    "internal_name",
    "version",
    "input_param_defs",
    "output_param_defs",
    "state",
    "links"
  ],
  "properties": {
    "name": {
      "$id": "#/items/properties/name",
      "type": "string"
    },
    "internal_name": {
      "$id": "#/items/properties/internal_name",
      "type": "string"
    },
    "version": {
      "$id": "#/items/properties/version",
      "type": "string",
    },
    "input_param_defs": {
      "$id": "#/items/properties/input_param_defs",
      "type": "array",
      "description": "Array of input parameters for the deployed app",
      "items": {
        "$ref": "#/definitions/parameter"
      }
    },
    "output_param_defs": {
      "$id": "#/items/properties/output_param_defs",
      "type": "array",
      "items": {
        "$ref": "#/definitions/parameter"
      }
    },
    "state": {
      "$id": "#/items/properties/state",
      "enum": [
        "Initial",
        "Creating",
        "WaitingForCreate",
        "Updating",
        "WaitingForUpdate",
        "Ready",
        "Deleting",
        "WaitingForDelete",
```

```
            "Deleted",
            "Error"
          ]
        },
        "links": {
          "$id": "#/properties/links",
          "type": "object",
          "required": [
            "app",
            "swagger"
          ],
          "properties": {
            "app": {
              "$id": "#/properties/links/properties/app",
              "type": "string",
            },
            "swagger": {
              "$id": "#/properties/links/properties/swagger",
              "type": "string",
            }
          }
        }
      }
    }
```

6.3.2 app RUN Result Schema

```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "required": [
    "success",
    "errorMessage",
    "outputParameters",
    "outputFiles",
    "consoleOutput",
    "changedFiles"
  ],
  "properties": {
    "success": {
      "$id": "#/properties/success",
      "type": "boolean",
    },
    "errorMessage": {
      "$id": "#/properties/errorMessage",
      "type": "string",
    },
    "outputParameters": {
      "$id": "#/properties/outputParameters",
      "type": "object",
      "required": [
        "result"
      ],
      "properties": {
        "result": {
          "$id": "#/properties/outputParameters/properties/result",
          "type": "integer"
        }
      }
    },
    "outputFiles": {
      "$id": "#/properties/outputFiles",
      "type": "object",
    },
    "consoleOutput": {
      "$id": "#/properties/consoleOutput",
      "type": "string",
```

```
      },
      "changedFiles": {
        "$id": "#/properties/changedFiles",
        "type": "array",
      }
    }
  }
```

6.4 token

6.4.1 token Response Schema

```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "required": [
    "token_type",
    "access_token",
    "expires_in",
    "expires_on",
    "token_id"
  ],
  "properties": {
    "token_type": {
      "$id": "#/properties/token_type",
      "type": "string",
    },
    "access_token": {
      "$id": "#/properties/access_token",
      "type": "string",
    },
    "expires_in": {
      "$id": "#/properties/expires_in",
      "type": "integer",
    },
    "expires_on": {
      "$id": "#/properties/expires_on",
      "type": "integer",
    },
    "token_id": {
      "$id": "#/properties/token_id",
      "type": "string",
    }
  }
}
```

7 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- Microsoft SQL Server 2019 Community Technology Preview 2.5 (CTP2.5)

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.