

# Compute Service: A RESTful Approach to the SAS<sup>®</sup> Programming Environment

Jason Spruill and Joseph Henry, SAS Institute Inc., Cary, NC

## ABSTRACT

SAS<sup>®</sup> Viya<sup>®</sup> provides an open architecture that makes it easy to include SAS<sup>®</sup> analytics into your application development via REST APIs. The Compute Service API provides access directly into the SAS language programming environment. The Compute service not only allows you to execute SAS code, but also provides access to SAS Data sets, SAS Files, Output Delivery System (ODS) results, and more. This paper provides an introduction to the Compute Service API, providing examples for how to best use the many features this API provides. Learn how the Compute Service API will make a great addition to your application development pipeline.

## INTRODUCTION

REST APIs have become much more popular than traditional remote procedure calls for integrating with services. This is due to the fact that with REST, the client and server are very loosely coupled, leaving the server free to change without directly affecting the clients. In a REST API, the communication between client and server is over HTTP, which allows for any number of consumers to communicate with and use the API. The OAuth authentication protocol provides secure client access to these servers..

SAS is following suit in SAS Viya with a wide range of services that each have a REST API. The Compute service provides a SAS programming environment, like you would see in a SAS Workspace Server for SAS 9, controlled via an easy to use REST API.

Resources are returned from requests that are made to the service. These resources might represent a session, a job, a collection of other resources, file information, or data. Resources are generally represented in a JSON format, which many languages can process easily. The format of these resources is governed by defined media types. Most resources will contain an array of links that will allow you to navigate from one resource to another, or to create a new resource from an existing one.

There are two main resources that are used in the Compute service. The Compute context is a persisted grouping of content containing initialization and configuration information. The Compute session is a running instance based on a Compute context that provides the direct access to the SAS environment. These resources will be discussed in further detail in the following sections.

## COMPARISON TO SAS V9

In SAS V9, there is a concept of the SAS Application server, which contains information related to what data could be accessed by applications using that application server. It also contains definitions of computational servers that provide access to SAS functionality for applications to use. Applications would use the SAS Application server as their connection point to the environment's computational layer.

In the SAS Viya Compute service, the Compute context is analogous to the SAS Application server, in which the Compute context provides applications with the ability to access the SAS programming environment in SAS Viya.

In SAS 9, a workspace server is most often used when client applications need to access SAS functionality. The SAS Workspace Server provides the ability to execute SAS code, process results, and access SAS data sets.

In the SAS Viya Compute service, the Compute session becomes the computational component that is used to access the functionality provided by SAS. The session is very similar in functionality to a SAS Workspace Server for SAS 9. The ability to execute SAS code, process results, and access SAS data sets can all be achieved by using the session.

## COMMON RESOURCE MEMBERS

The following sections describe members that can be used in multiple resources or media types that are supported by the Compute service.

### Attributes

**Attributes** are objects that can be specified to control the behavior of the Compute session. Currently, there is only one member of the Attributes object, named **sessionInactiveTimeout**. This attribute controls the amount of time, in seconds, that a session can be inactive before it terminates. This attribute might be stored on the Compute context, or used as part of a session request as shown below. The attribute has a default value of fifteen minutes (900 seconds).

```
{
  ...
  "attributes": {
    "sessionInactiveTimeout": 30000
  },
  ...
}
```

### Environment

**Environment** is an object that can be specified to control certain aspects of the run-time environment in a Compute session. An environment contains two members:

- **options** - Allow the user to specify SAS options that will be processed by the session. These options might be any valid SAS command line option. These options behave similarly to options in a sasv9.cfg or sasv9\_usermods.cfg file in a SAS 9 environment.
- **autoExecLines** - Allow for adding specific lines of SAS code that are processed as SAS autoexec code. The **autoExecLines** member accepts an array of strings that contain SAS language code. This code is processed when a session initializes. These lines behave similarly to code in an autoexec.sas or autoexec\_usermods.sas file in a SAS 9 environment.

The environment can be stored on either a Compute context, or a session request as shown below. When set on the context, this environment is applied to every session that is created from that context. When an environment is set as part of a session request, the environment applies only to that session.

```
{
  ...
  "environment": {
    "options": [
      "-memsize 4g",
      "-validvarname any"
    ],
    "autoExecLines": [
      "filename salesdata '/opt/qtr1'",
      "filename custdata '/opt/newcust';"
    ]
  },
  ...
}
```

## COMPUTE CONTEXTS

The base resource used by the Compute service is a Compute context. The context contains basic configuration information to be used for creating a Compute session. Contexts can be created on a per-application basis, meaning that individual applications can define a context that is specific to their requirements.

For an application's usage, contexts are generally created for that application when the application is initially loaded. However, in cases where a new context needs to be created, SAS Environment Manager can be used to create a new context.

To programmatically create a new context, send a request to the Compute service that uses the media type [application/vnd.sas.compute.context.request](#). Here is an example request.

```
{
  "name": "myApplicationContext",
  "version": 1,
  "description": "My Application Context",
  "launchContext": {
    "contextName": "myLauncherContext"
  },
  "launchType": "service",
  "authorizedUsers": [
    "myuser1"
  ]
}
```

### Launcher Service

In SAS 9, an application called the Object Spawner launches SAS Workspace Server processes when needed. In SAS Viya, the Launcher service launches processes.

When a Compute context is created, a reference to an existing Launcher context is required. The Launcher context is a resource in the Launcher service that contains information related to how processes are launched in the deployment. The **launchContext** member in the [application/vnd.sas.compute.context.request](#) object must contain either the **contextName** or the **contextId** of the Launcher context to interface with.

### Security

There are three members of the [application/vnd.sas.compute.context.request](#) media type that control which users have permission to create Compute sessions based on this context. The **authorizedGroups** and **authorizedUsers** members work together to specify the groups and users that can create sessions. These groups and users are defined in the Identities service. When used, security rules are created to ensure that only specified users are allowed access. Other users receive errors when they attempt to create sessions.

If the **authorizeAllAuthenticatedUsers** field is set to true, then any user that can authenticate to the service is allowed to create sessions based on this Compute context.

## ACCESSING COMPUTE CONTEXTS

The Compute service maintains and stores Compute contexts that have been created. Contexts can be accessed by users or applications at any time that the Compute service is available as shown in Request 1. By default, an [application/vnd.sas.collection](#) is returned with all the available contexts. You can also request a context specifically by name as shown in Request 2 or by *id* as shown in Request 3. Both of these requests return an [application/vnd.sas.compute.context](#) response.

```
GET /compute/contexts HTTP/1.1
```

### Request 1. Retrieve All Compute Contexts

```
GET /compute/contexts?name=myContextName HTTP/1.1
```

### Request 2. Retrieve a Context by Name

```
GET /compute/contexts/myContextId HTTP/1.1
```

### Request 3. Retrieve a Context by Id

The context that is returned contains similar members to the [application/vnd.sas.compute.context.request](#) media type that is used to create the context. In addition, the links that can be used with this context are provided. These are important, as they allow you the ability to follow a tag or relation to perform a task within the context.

For the purposes of this paper, we focus on the link called **createSession**. If this link is available, it can be used to create a Compute session for this context. An example of this link is shown below.

```
{
  "method": "POST",
  "rel": "createSession",
  "uri": "/compute/contexts/48c6755b-50dd-4534-a23b-499faf7ff60b/sessions",
  "type": "application/vnd.sas.compute.session.request",
  "responseType": "application/vnd.sas.compute.session"
}
```

A link is based on the type [application/vnd.sas.link](#). The members of this link tell you how to make a call to perform a request. For this link, you create a request for a session similar to the one shown in Request 4.

```
POST /compute/contexts/{contextId}/sessions HTTP/1.1
Content-type: application/vnd.sas.compute.session.request+json
Accept: application/vnd.sas.compute.session+json
```

### Request 4. Create a Compute Session

## COMPUTE SESSIONS

The Compute session is the starting point for accessing the features that are available in the Compute service. When a session is created, a SAS run-time environment is initialized on a worker node that is referred to as a Compute server process. The session is your access point to the Compute server, providing the ability to execute SAS code and interact with data.

## CREATING A SESSION

A request to create a session might take a request body defined as an [application/vnd.sas.compute.session.request](#). All the members of this media type are optional, and are given default values if missing. To create a session, submit a POST request as shown in Request 5. The **contextId** that is used in the URL represents the unique identifier of the Compute context that you are referencing. The response is always the session that is created and is represented as an [application/vnd.sas.compute.session](#).

```
POST /compute/contexts/{contextId}/sessions HTTP/1.1
```

```
{
  "name" : "My Compute Session",
  "description" : "An example Compute Service Session.",
  "attributes" : {
    "sessionInactiveTimeout" : 300
  }
  "environment": {
    "options": [
      "memsize 8g"
    ]
  }
}
```

### Request 5. Create a Compute Session

## USING THE SESSION

To use a session and access other features and functionality available in the session, use the links that are provided in the Compute session resource. The links provide information about the method to use, as well as the URL to access, and any media types to use. The links that are available on a session are listed in the following table.

<b>self</b>	Returns the latest version of the session resource
<b>state</b>	Returns the current state of the session
<b>cancel</b>	Cancels the current work that is running in the session
<b>delete</b>	Destroys the session
<b>execute</b>	Executes a job in the session
<b>jobs</b>	Lists all current jobs in the session
<b>files</b>	Lists all SAS filerefs that are available to the session
<b>assign</b>	Assigns a new SAS fileref in the session
<b>librefs</b>	Lists all available SAS librefs in the session
<b>log</b>	Provides access to the full session log
<b>listing</b>	Provides access to the full session listing output
<b>results</b>	Provides access to any results that are created in the session
<b>variables</b>	Lists all macro variables that are defined in the current session
<b>engines</b>	Lists engines that are available for use in this session

## COMPUTE JOBS

The workhorse of the Compute service is the Jobs API. A job is an object that contains one or more lines of SAS code along with any options or variables that are needed to execute that code. A Compute job requires a Compute session to be executed in. Jobs have the following characteristics:

- All jobs are submitted to a session asynchronously. This means that the call returns immediately, and you need to query the job for its current state.
- Jobs are not autonomous. Each job that is submitted to the same session builds off any previously executed job or modified session attributes.

- Jobs do not run concurrently. If a new job is submitted before the currently running job has completed, it is queued up and begins execution after the current job has completed. Jobs are queued up in the order in which they are received.

## CREATING A JOB

To create a Compute job, follow the **execute** link on the Compute session with an [application/vnd.sas.compute.job.request](#) as shown in Request 6.

```
POST /compute/sessions/{sessionId}/jobs HTTP/1.1
Content-Type: application/vnd.sas.compute.job.request+json

{
  "environment": {
    "options": ["nodate",
              "linesize=72"]
  },
  "variables": {
    "NUMVAR": 903,
    "CHARVAR": "String Value"
  },
  "code": ["libname foo '~/work/foo';",
          "filename myods 'example.html';",
          "",
          "data foo.bar;",
          "x = 10;",
          "output;",
          "run;",
          "",
          "ods html body=myods;",
          "proc print data=foo.bar;run;",
          "ods html close;"]
}
```

### Request 6. Compute Job Request

When the job request is accepted, you get an HTTP 201 response along with an [application/vnd.sas.compute.job](#) response body.

## JOB STATE

All Compute jobs are submitted asynchronously, meaning that the call to create a job always returns immediately. To get that status of a job, that is, the job's state, follow the **state** link on the [application/vnd.sas.compute.job](#) object. Request 7 shows a sample state request:

```
GET /compute/sessions/{sessionId}/jobs/{jobId}/state HTTP/1.1
Accept: text/plain
```

### Request 7. Query Job State

The job state request returns plain text with the value of the job's state, which can be one of the following values:

<b>completed</b>	Job has finished running
<b>error</b>	Job has completed and the session is in an error state
<b>warning</b>	Job has completed and the session is in a warning state
<b>canceled</b>	Job has been canceled
<b>running</b>	Job is currently executing
<b>pending</b>	Job is in the run queue

## JOB OUTPUT

Every Compute job that is executed in the Compute service might create some form of output. Output can be in one of these forms: lines in the SAS log, lines in a listing output, ODS files, data sets, libraries, or filerefs. The following links in the job response are for retrieving output:

<b>log</b>	Provides access to the SAS log
<b>listing</b>	Provides access to the job listing output
<b>results</b>	Provides access to data sets, filerefs, librefs, and ODS output that is created by the job

These links are scoped to the job, which means only output that was created as a direct result of the job is present. If you want to see the output for the entire session, you can follow the same links from the session object.

### SAS LOG AND LISTING OUTPUT

The most basic output created from a Compute job is the SAS log. The log can be retrieved by following the **log** link on the Job as shown in Request 8.

```
GET /compute/sessions/{sessionId}/jobs/{jobId}/log HTTP/1.1
Accept: application/vnd.sas.collection+json
```

#### Request 8. Retrieve SAS Log

The response is an [application/vnd.sas.collection](#) that contains [application/vnd.sas.compute.log.line](#) items. The [application/vnd.sas.compute.log.line](#) contains the following members:

<b>line</b>	The text line that is part of the log												
<b>type</b>	The type of the line. Here are the possible line types: <table><tr><td>normal</td><td>highlighted</td><td>source</td><td>title</td></tr><tr><td>byline</td><td>footnote</td><td>error</td><td>warning</td></tr><tr><td>note</td><td>message</td><td></td><td></td></tr></table>	normal	highlighted	source	title	byline	footnote	error	warning	note	message		
normal	highlighted	source	title										
byline	footnote	error	warning										
note	message												

Output that is sent to the listing destination is available from the job by following the **listing** link. Like the log response, the listing response also returns a collection of [application/vnd.sas.compute.log.line](#) items.

### RESULTS

Apart from SAS log and listing output, you can retrieve the remaining output from a job by following the **results** link, as shown in Request 9.

```
GET /compute/sessions/{sessionId}/jobs/{JobId}/results HTTP/1.1
Accept: application/vnd.sas.collection
```

#### Request 9. Retrieve Listing Output

The response is an [application/vnd.sas.collection](#) that contains [application/vnd.sas.compute.result](#) items. Following the **results** link from the job request in Request 6 returns a response as shown in Response 1.

```

{
  "accept": "application/vnd.sas.compute.result",
  "count": 4,
  "items": [{
    "id": "example.html",
    "links": [{
      "method": "GET",
      "rel": "self",
      "type": "text/html",
      "uri": "/compute/sessions/{sessionId}/results/a7309add/example.html"
    }],
    "name": "example.html",
    "type": "ODS",
    "version": 1
  },
  {
    "id": "FOO",
    "links": [{
      "method": "GET",
      "rel": "self",
      "type": "application/vnd.sas.compute.library",
      "uri": "/compute/sessions/{sessionId}/data/FOO"
    }],
    "name": "FOO",
    "type": "LIBRARY",
    "version": 1
  },
  {
    "id": "BAR",
    "links": [{
      "method": "GET",
      "rel": "self",
      "type": "application/vnd.sas.compute.data.table",
      "uri": "/compute/sessions/{sessionId}/data/FOO/BAR"
    }],
    "name": "BAR",
    "type": "TABLE",
    "version": 1
  },
  {
    "id": "myods",
    "links": [{
      "method": "GET",
      "rel": "self",
      "type": "application/vnd.sas.compute.fileref",
      "uri": "/compute/sessions/{sessionId}/filerrefs/myods"
    }],
    "name": "myods",
    "type": "FILE",
    "version": 1
  }
  ],
  "limit": 10,
  "links": [...],
  "name": "Results",
  "start": 0,
  "version": 2
}

```

### Response 1. Job Results Response

If the result type is LIBRARY, TABLE, or FILE, the link is to their respective resources. Those resources are described in Accessing DATA and SAS Files, later in this paper.

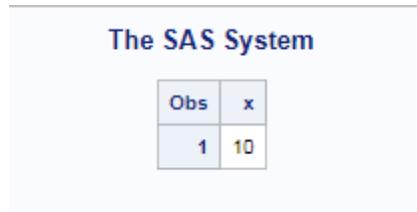
If the result type is ODS, that means that the Output Delivery System created output to a destination other than listing. In this example, there is one ODS item type that was created by the PRINT procedure in Request 6. The link to this item looks similar to this sample:

```

{
  "method": "GET",
  "rel": "self",
  "type": "text/html",
  "uri": "/compute/sessions/{sessionId}/results/a7309add/example.html"
}

```

The output resource is hosted by the Compute server, and can be retrieved just like a normal web resource. An HTTP GET on the link returns the HTML file, which renders in a web browser as shown in Image 1.



Obs	x
1	10

**Image 1. ODS HTML Output**

## ACCESSING DATA AND SAS FILES

The Compute service API provides direct access to SAS libraries, data sets, and filerefs. Filerefs can be accessed via the filerefs endpoint, whereas libraries and data sets are accessed via the data endpoint.

### FILEREFs

Any assigned fileref in the current Compute session can be accessed via the filerefs endpoint with the following URI scheme:

```
/compute/session/{sessionId}/filerefs/{filerefName}
```

You can see an example link in Response 1 that looks similar to this sample:

```

{
  "method": "GET",
  "rel": "self",
  "type": "application/vnd.sas.compute.fileref",
  "uri": "/compute/sessions/{sessionId}/filerefs/myods"
}

```

Following this link returns an [application/vnd.sas.compute.fileref](#) response type as shown in Response 2.

```

{
  "accessMethod": "DISK",
  "fileName": "example.html",
  "filePath":
"/r/ge.unx.sas.com/vol/vol1710/u71/johenr/projects/vb007/compute/comp_server2/exam
ple.html",
  "fileSize": 34991,
  "id": "myods",
  "isAggregation": false,
  "isDirectory": false,
  "links": [{
    "method": "GET",
    "rel": "self",
    "type": "application/vnd.sas.compute.fileref",
    "uri": "/compute/sessions/{sessionId}/filerefs/myods"
  },
  {
    "method": "DELETE",
    "rel": "deassign",
    "uri": "/compute/sessions/{sessionId}/filerefs/myods"
  },
  {
    "method": "GET",
    "rel": "content",
    "type": "text/html",
    "uri": "/compute/sessions/{sessionId}/filerefs/myods/content"
  },
  {
    "method": "PUT",
    "rel": "upload",
    "type": "text/html",
    "uri": "/compute/sessions/{sessionId}/filerefs/myods/content"
  },
  {
    "method": "POST",
    "rel": "append",
    "type": "text/html",
    "uri": "/compute/sessions/{sessionId}/filerefs/myods/content"
  },
  {
    "method": "DELETE",
    "rel": "delete",
    "uri": "/compute/sessions/{sessionId}/filerefs/myods/content"
  }
  ],
  "modifiedTimeStamp": "2018-01-29T14:34:35Z",
  "name": "myods",
  "version": 2
}

```

## Response 2. Fileref Response

All available filerefs that are assigned in the current session are accessed by following the **filerefs** link on the session resource. This link returns a collection of [application/vnd.sas.compute.fileref](#) items.

### DATA

The data endpoint provides access to libraries and data sets with the following URI scheme:

`/compute/sessions/{sessionId}/data/{library}/{dataset}`.

A key point here is that a data set must be referenced from a library. In other words, there is no default library. For example, if you were accessing data set Bar that is located in the Work library, the URI would be:

`/compute/sessions/{sessionId}/data/WORK/BAR`.

All assigned SAS librefs in the current session can be listed by following the **librefs** link on the session resource, as shown in Request 10.

```
GET /compute/data HTTP/1.1
Accept: application/vnd.sas.collection+json
```

### Request 10. Retrieve All Librefs in Session

Each item in the collection contains a link to retrieve the actual library as shown below:

```
{
  "method": "GET",
  "rel": "self",
  "type": "application/vnd.sas.compute.library",
  "uri": "/compute/sessions/{sessionId}/data/F00"
}
```

Following this link (shown in Request 11) returns an [application/vnd.sas.compute.library](#) as shown in Response 3. To list all the data sets that are contained in a libref, follow the tables link for the previous response, as shown in Request 12.

```
GET /sessions/0001-ses0000/data/F00 HTTP/1.1
Accept: application/vnd.sas.compute.library+json
```

### Request 11. Retrieve the Libref FOO

```
{
  "concatenationCount": 0,
  "engineName": "V9",
  "fileFormat": "7",
  "flags": 32,
  "id": "F00",
  "libref": "F00",
  "links": [{
    "method": "DELETE",
    "rel": "delete",
    "uri": "/compute/sessions/{sessionId}/data/F00"
  }],
  {
    "itemType": "application/vnd.sas.compute.data.table.summary",
    "method": "GET",
    "rel": "tables",
    "type": "application/vnd.sas.collection",
    "uri": "/compute/sessions/{sessionId}/data/F00"
  }],
  "name": "F00",
  "options": "",
  "physicalName": "/sas/work/foo",
  "version": 1
}
```

### Response 3. Libref Representation

```
GET /sessions/0001-ses0000/data/F00 HTTP/1.1
Accept: application/vnd.sas.collection+json
```

### Request 12. List All Data Sets in Libref FOO

As with the library collection, each item contains one link to the [application/vnd.sas.compute.data.table](#) resource, as shown below:

```
{
  "method": "GET",
  "rel": "self",
  "type": "application/vnd.sas.compute.data.table",
  "uri": "/compute/sessions/{sessionId}/data/F00/BAR"
}
```

Following this link (shown in Request 13) returns an [application/vnd.sas.compute.data.table](#), as shown in Response 4.

```
GET /sessions/0001-ses0000/data/FOO/BAR HTTP/1.1
Accept: application/vnd.sas.compute.data.table+json
```

### Request 13. Retrieve Data Set FOO.BAR

```
{
  "bookmarkLength": 12,
  "columnCount": 1,
  "compressionRoutine": "NO",
  "creationTimeStamp": "2018-01-30T18:11:14Z",
  "engine": "V9",
  "id": "BAR",
  "label": "",
  "libref": "FOO",
  "links": [{
    "itemType": "application/vnd.sas.compute.data.table.row",
    "method": "GET",
    "rel": "rows",
    "type": "application/vnd.sas.collection",
    "uri": "/compute/sessions/{sessionId}/data/FOO/BAR/rows"
  }],
  {
    "method": "GET",
    "rel": "rowSet",
    "type": "application/vnd.sas.compute.data.table.row.set",
    "uri": "/compute/sessions/{sessionId}/data/FOO/BAR/rowSet"
  },
  {
    "itemType": "application/vnd.sas.compute.data.table.column",
    "method": "GET",
    "rel": "columns",
    "type": "application/vnd.sas.collection",
    "uri": "/compute/sessions/{sessionId}/data/FOO/BAR/columns"
  }],
  "logicalRecordCount": 1,
  "modifiedTimeStamp": "2018-01-30T18:11:14Z",
  "name": "BAR",
  "physicalRecordCount": 1,
  "recordLength": 8,
  "rowCount": 1,
  "version": 1
}
```

### Response 4. Data Table

## SESSION CLEANUP

After you have completed the work for the Compute session, the session can be destroyed. Destroying the session can be accomplished by using the **delete** link given on the session resource. The request would look like the following.

```
DELETE /compute/sessions/{sessionId} HTTP/1.1
```

The Compute service processes this request and cleans up any resources that are associated with the session. This includes shutting down the Compute server process that was launched in your deployment to process the requests that are submitted to this session. As this process terminates, any temporary content that was created during the execution of this service is deleted. To continue using the Compute service, you must create a new session.

If you do not delete a session directly by using the link above, the session might still be terminated automatically. The **sessionInactiveTimeout** attribute controls how long the session remains inactive before terminating itself.

## CONCLUSION

REST APIs provide flexibility that allows developers to create the next generation of tools and applications. SAS Viya services use secure communication over HTTP combined with OAuth to give users secure communication between their applications and data.

The Compute service API provides access to the SAS programming environment for users of SAS Viya, allowing users to execute code, manage files, access data, and view results by submitting requests via HTTP. The Compute service API provides a flexible interface to allow developers and users to build the next generation of their tools and applications to access the power of SAS.

## MEDIA TYPES

### application/vnd.sas.compute.context.request

<b>name</b>	This is a required member that indicates the name of the context. The name of the context can be used to reference the context in the service and must be unique.
<b>description</b>	Optional description for the session.
<b>launchContext</b>	A required object that allows for linking with the Launcher service context.
<b>launchType</b>	A required member that indicates the type of launch to use for the context. For this release, the value should be "service".
<b>attributes</b>	Optional properties that can be used to control the behavior of the sessions that are created from this context.
<b>environment</b>	Object that contains subobjects that control the SAS environment that is used.
<b>authorizedUsers</b>	A list of users that are allowed to create sessions from this context.
<b>authorizedGroups</b>	A list of groups that are allowed to create sessions from this context.
<b>authorizeAllAuthenticatedUsers</b>	A Boolean value to indicate that any authenticated user can create sessions from this context.
<b>mediaTypeMap</b>	Provides overrides to media types based on file types returned from the Compute service.

### application/vnd.sas.compute.context

<b>name</b>	This is a required member that indicates the name of the context. The name of the context can be used to reference the context in the service and must be unique.
<b>description</b>	Optional description for the session.
<b>launchContext</b>	A required object that allows for linking with the Launcher service context.
<b>launchType</b>	A required member that indicates the type of launch to use for the context. For this release, the value should be "service".
<b>attributes</b>	Optional properties that can be used to control the behavior of the sessions that are created from this context.
<b>environment</b>	Object containing subobjects that control the SAS environment that is used.
<b>mediaTypeMap</b>	Provides overrides to media types based on file types that are returned from the Compute service.
<b>createdBy</b>	The user that created a context.

<b>modifiedBy</b>	The user that last updated a context.
<b>creationTimeStamp</b>	The timestamp indicating when a context was created.
<b>modifiedTimeStamp</b>	The timestamp indicating when a context was last updated.

### **application/vnd.sas.compute.session.request**

<b>name</b>	Name that is associated with this session. This is a human-readable string that describes the session.
<b>description</b>	Description for this session.
<b>attributes</b>	Properties that can be used to control the behavior of the session.
<b>environment</b>	Object containing subobjects that control the SAS environment that runs in the session.
<b>resources</b>	Resources that are defined in other services that can be referenced directly by your session.

### **application/vnd.sas.compute.session**

<b>id</b>	The identifier for this session, used in all access to a session.
<b>name</b>	Name that is associated with this session.
<b>description</b>	Description for this session.
<b>owner</b>	The user that requested and created a session.
<b>creationTimeStamp</b>	The timestamp that indicates when a session was created.
<b>serverId</b>	Identifier of the server that a session is a member of.
<b>state</b>	The current state of a session. A session that is ready for use has the state "idle".
<b>sessionConditionCode</b>	The condition code for a session. This is set to be the highest value of any jobConditionCode for jobs that are executed in the session.
<b>attributes</b>	Name-value attributes that are used in the creation of a session.
<b>environment</b>	The environment that is specified when creating a session.
<b>logInfo</b>	Information about the data that is available from the log endpoint on a session.
<b>listInfo</b>	Information about the data that is available from the listing endpoint on a session.
<b>statistics</b>	Statistics information about the resource usage in a session.
<b>stateElapsedTime</b>	The number of seconds since the state has been updated.
<b>failureMessages</b>	On session failure, a list of error messages.
<b>applicationName</b>	If available, the name of the application that requested a session.

### application/vnd.sas.compute.job.request

<b>name</b>	Optional name that is associated with a job. This is a human-readable string that describes a job.
<b>description</b>	Optional description for this job.
<b>environment</b>	An optional object containing subobjects that describe the environment that the job runs in. Currently, the only supported subobject is an array of SAS options.
<b>variables</b>	Optional set of name-value pairs that are used as macro variables in the executed code.
<b>code</b>	Strings that contain source code to be executed. Each string that is submitted is considered to be a single line submitted to the SAS language processor.

### application/vnd.sas.compute.job

<b>id</b>	The identifier for a job.
<b>name</b>	The optional name for a job. The name is specified when a job is created. If no name is given, this is empty.
<b>description</b>	The optional description for a job. The description is specified when a job is created. If no description is given, this is empty.
<b>creationTimeStamp</b>	The timestamp of when a job was created.
<b>completedTimeStamp</b>	The timestamp of when a job completed.
<b>sessionId</b>	The identifier of the session that this job is a member of.
<b>state</b>	The current state of a job. Here are the possible values: pending - The job has been submitted but is not running yet. running -The job is currently executing. completed -The job has completed execution. canceled - The job has been canceled. warning - The job completed with warnings. error -The job completed with errors.
<b>jobConditionCode</b>	The condition code for a job. This is set upon completion of a job and is the value of the SYSCC macro variable.
<b>logInfo</b>	Information about the data that is available from the log endpoint on the session. Here are the possible values: lineCount - The current number of lines that are available. modifiedTimeStamp - The timestamp of the last time that the log was updated.
<b>listInfo</b>	Information about the data that is available from the listing endpoint on the session. Here are the possible values: lineCount - The current number of lines available. modifiedTimeStamp -The timestamp for the last time that the listing was updated.

<b>statistics</b>	<p>Statistical information about resource usage by a job in its session. Here are the possible values:</p> <p>userCpuTime - The amount of user CPU time that is spent by a session since this job started. This value is of type double in units of seconds. If this job is complete, it is the amount of user CPU time that is spent by the session over the duration of the job.</p> <p>systemCpuTime - The amount of system CPU time that is spent by the session since this job started. This value is of type double in units of seconds. If this job is complete, it is the amount of system CPU time that is spent by the session over the duration of the job.</p>
<b>stateElapsedTime</b>	The number of seconds since a state has been updated.

#### application/vnd.sas.compute.log.line

<b>line</b>	The textual line that is part of the log.
<b>type</b>	<p>The type of the line. Here are the possible line types:</p> <p>normal  highlighted  source  title  byline  footnote  error  warning  note  message</p>

#### application/vnd.sas.compute.result

<b>name</b>	Name of the result.
<b>type</b>	<p>The type of result. Possible result types:</p> <p>ODS - The result is ODS generated output  LIBRARY - The result is a SAS libref  TABLE - The result is a SAS data set  FILE - The result is a SAS fileref</p>

### application/vnd.sas.compute.library

<b>name</b>	Name of a result
<b>concatenationCount</b>	Number of concatenations that make up a library
<b>engineName</b>	The name of the engine that is used to create a library
<b>fileFormat</b>	The identifier for the storage format
<b>libname</b>	The LIBNAME of a library
<b>options</b>	Options that were used to create a library
<b>physicalName</b>	The resolved location of a library

### application/vnd.sas.compute.fileref

<b>name</b>	Name of a result.
<b>dateModified</b>	The timestamp of the last modified date of the underlying entity.
<b>isDirectory</b>	If true, the fileref represents a directory or an aggregation of directories. If false, the fileref is not a simple file or an aggregation of flat files.
<b>isAggregation</b>	If true, the fileref represents an aggregation of files or directories.
<b>fileSize</b>	The size of the underlying entity.
<b>fileName</b>	The name of the underlying entity.

### application/vnd.sas.compute.data.table

<b>name</b>	Name of a table
<b>bookmarkLength</b>	The length of the bookmark in a data set
<b>compressionRoutine</b>	Indication of what type of compression is used for a data set
<b>dateCreated</b>	The time stamp for when this data set was created
<b>dateModified</b>	The time stamp for the last modified date for a data set
<b>engine</b>	The identifier for the engine that is accessing a data set
<b>fileName</b>	The name of the underlying entity
<b>label</b>	The label that is given to describe a data set
<b>libref</b>	The libref that a data set is being accessed through
<b>logicalRecordCount</b>	The number of logical records that are found in a data set
<b>columnCount</b>	The number of columns reported for a data set
<b>rowCount</b>	The number of rows reported for a data set
<b>physicalRecordCount</b>	The number of physical records that are found in a data set
<b>recordLength</b>	The length of each record in a data set

**application/vnd.sas.compute.data.table.column**

<b>name</b>	Name of a result.
<b>dateModified</b>	The timestamp of the last modified date of the underlying entity.
<b>isDirectory</b>	If true, a fileref represents a directory or an aggregation of directories. If false, the fileref is not a simple file or an aggregation of flat files.
<b>isAggregation</b>	If true, the fileref represents an aggregation of files or directories.
<b>fileSize</b>	The size of the underlying entity.
<b>fileName</b>	The name of the underlying entity.

**application/vnd.sas.compute.data.table.row**

<b>name</b>	The name of a column
<b>label</b>	The label given to a column
<b>Index</b>	Numerical index for a column
<b>length</b>	The length of the data in a column
<b>type</b>	Identifier for the type of data in a column (for example, FLOAT)
<b>format</b>	An object that describes the format specification
<b>informat</b>	An object that describes the informat specification

## RECOMMENDED READING

“application/vnd.sas.collection Media Type.” SAS for Developers website. Available at <https://developer.sas.com/reference/schema/application/vnd.sas.collection/v2/index.html>. Accessed in February, 2018.

“Links in REST APIs.” SAS for Developers website. Available at <https://developer.sas.com/guides/rest/links/>. Accessed in February, 2018.

SAS for Developers website. Available at <http://developer.sas.com>. Accessed in February, 2018.

Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and Mortimore, C. *OpenID Connect Basic Client Implementer's Guide 1.0 (Draft)*. Available at [https://openid.net/specs/openid-connect-basic-1\\_0.html](https://openid.net/specs/openid-connect-basic-1_0.html). Accessed in February, 2018.

Henry, Joseph. 2017. “Show Off Your OAuth”. *Proceedings of the SAS Global 2017 Conference*. Cary, NC: SAS Institute, Inc. Available at <http://support.sas.com/resources/papers/proceedings17>.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jason Spruill  
100 SAS Campus Drive  
Cary, NC 27513  
SAS Institute Inc.  
[jason.spruill@sas.com](mailto:jason.spruill@sas.com)  
<http://www.sas.com>

Joseph Henry  
100 SAS Campus Drive  
Cary, NC 27513  
SAS Institute Inc.  
[joseph.henry@sas.com](mailto:joseph.henry@sas.com)  
<http://www.sas.com>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.