SAS® Intelligent Decisioning: Decision Management REST API Examples

2020.1 -2021.1.2*

* This document might apply to additional versions of the software. Open this document in SAS Help Center and click on the version in the banner to see all available versions.

SAS® Documentation
May 18, 2023

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2020. *SAS® Intelligent Decisioning: Decision Management REST API Examples*. Cary, NC: SAS Institute Inc.

SAS® Intelligent Decisioning: Decision Management REST API Examples

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

May 2023

v_001-P1:edmresttut

Contents

1

About the Examples

Introduction

SAS Viya REST APIs are based on resource-oriented URLs, use HTTP authentication and HTTP verbs, and return HTTP response codes. With SAS Viya REST APIs, you can create and access SAS resources by using any client technology. These APIs make it easy to integrate the capabilities of SAS Viya into your business processes or to extend and customize SAS Viya to meet specific requirements.

The Decision Management REST APIs provide access to machine scoring, business rules, models, decisions, treatments, and more. The examples in this document show how to use these APIs to perform some of the more common decision management tasks that you might want to do through the APIs.

Note: The examples in this document were written in Python 3.7 and run in Jupyter Notebook, which was launched from Anaconda Navigator.

Client ID

Before you use these APIs, your administrator must register a client identifier. You must specify this ID to obtain an access token to use with REST API requests. For more information, see SAS Viya REST APIs: Authentication to SAS Viya.

Authorization

The REST APIs require authentication for all operations. Authentication is a means of verifying the identity of the user or agent that is making the request. Authentication is handled with an OAuth2-based service in the SAS Logon Manager. You must obtain an access token, and use that token on all requests. The example code shown in Chapter 2, “Define Basic Methods and Get an Access Token,” on page 5 provides a function for obtaining an access token.

Code Files for These Examples

The ZIP file that contains the code for the examples in this document is named REST_API_Examples.zip and is available at https://support.sas.com/en/software/intelligent-decisioning-support.html#documentation. To extract these files to your computer:

1. Save the ZIP file into a folder on your machine.
2. In Microsoft Windows Explorer, right-click **REST_API_Examples.zip**, and select **WinZip** ⇨ **Extract to here**.

   Windows creates a folder named REST_API_Examples, which contains the example files.

Additional Documentation and Examples

Complete reference information for the Decision Management API is available on the web at https://developer.sas.com/apis/rest/DecisionManagement/.

Additional examples of using the Decision Management API are available on GitHub at https://github.com/sassoftware/devsascom-rest-api-samples/tree/master/DecisionManagement.

2

Define Basic Methods and Get an Access Token

Introduction

The code shown in "Example Code" defines the get(), post(), put(), and getAccessToken() functions for use with the examples in this document. The code calls the getAccessToken() function to request an access token and returns the token in the variable `accessToken1`. You must specify this access token with all subsequent requests to SAS Viya APIs.

Important: This code is required by all of the other examples in this document. Prepend this code to each example before you run it.

Updating Objects with the put() Function

When you submit PUT requests that update existing resources, you must specify a header name and value that identify the specific state of the resource that you are

updating. This name-value pair prevents a request from updating a resource with outdated content.

For example, you can submit a GET request to retrieve the content of a decision, modify it, and submit a PUT request to update the decision. The PUT request must include the `If-Match` header name and the value of the ETag header from the response to the initial GET request. If someone else modifies the decision before you submit your PUT request, then the ETag value is no longer valid and your request fails.

The header name that you need to specify in a PUT request is documented in the REST API documentation for the request that you are submitting. Depending on the request, the header name is either `If-Match` or `If-Unmodified-Since`. Specify the header name and its value with the `conditionalPutKey` and `conditionalPutValue` parameters for the put() function.

Table 2.1 *Conditional PUT Keys and Values*

conditionalPutKey	conditionalPutValue
`If-Match`	`ETag` value from the response header for the last GET request for the resource. The examples Chapter 5, “Activate Treatment Groups,” on page 25 and Chapter 8, “Update Subject Contact History,” on page 43 both submit PUT requests that include the `If-Match` header.
`If-Unmodified-Since`	`Last-Modified` value from the response header for the last GET request for the resource. The GET request returns this value in Coordinated Universal Time (UTC) format. However, when you specify this value in a PUT request, you must specify the value in the following format: `day-name, day month year hour:minutes:seconds GMT` For example: `Mon, 24 Aug 2020 20:58:15 GMT` For more information, see Last-Modified in the MDN (Mozilla Developer Network) Web Docs.

Example Code

IMPORTANT Replace the client ID, client secret (if applicable), host URL, user ID, and password with the appropriate values for your environment.

```
import sys, json
import urllib.parse as u1
import urllib.request as urllib2
```

```
import urllib.error as urllib3
import base64
import pprint
import requests
import re as regexp
import datetime
from time import sleep

# The GETACCESSTOKEN function requests an access token using the SAS
Logon
# OAuth API. The response contains a field named access_token that
contains
# the value of the token that you use for subsequent API requests.

def getAccessToken(baseUrl1):
    url1 = baseUrl1 + '/SASLogon/oauth/token'

    # Replace client-ID and client-secret
    # with values appropriate for your environment.
    s = "client-ID:client-secret"

    # Encode the value of the client ID.
    tokenCredentials =
base64.b64encode(s.encode('ascii')).decode('ascii')
    headers = {"Accept": "application/json",
               "Authorization": "Basic " + tokenCredentials,
               "Content-Type": "application/x-www-form-urlencoded" }
    values = { "grant_type": "password",
               "username": userid,
               "password": password }

    # Convert the values dictionary into a string.
    dV = u1.urlencode(values)
    dV = dV.encode('ascii')

    # Request an access token.
    req = urllib2.Request(url1, data=dV, headers=headers)
    try:
        # Open the response object, and convert it into a JSON object.
        responseLogon = urllib2.urlopen(req)
        body = responseLogon.read()
        responseBodyJson = json.loads(body)

        # Extract the access token from the response.
        accessToken1 = responseBodyJson['access_token']
    except urllib3.URLError as e:
        if hasattr(e, 'reason'):
            print ('Failed to reach a server.')
            print ('Error: ', e.reason)
            print (e)
        elif hasattr(e, 'code'):
            print ('The server could not fulfill the request.')
            print ('Error: ', e.reason)
    except urllib3.HTTPError as e:
        print ('Error: ', e.reason)
    return accessToken1;
```

```
# Define the GET function. This function defines request headers,
# submits the request, and returns both the response body and
# the response header.

def get(url1, accessToken1, accept):
    headers = {"Accept": accept,
               "Authorization": "bearer " + accessToken1}
    try:
        # Submit the request.
        req = urllib2.Request(url1, headers=headers)

        # Open the response, and convert it to a string.
        domainsResponse = urllib2.urlopen(req)
        body = domainsResponse.read()

        # Return the response body and the response headers.
        respHeaders = domainsResponse.headers
        return body, respHeaders
    except urllib3.URLError as e:
        if hasattr(e, 'reason'):
            print ('Failed to reach a server.')
            print ('Error: ', e.read())
        elif hasattr(e, 'code'):
            print ('The server could not fulfill the request.')
            print ('Error: ', e.read())
    except urllib3.HTTPError as e:
        print ('Error: ', e.read())

# Define the POST function. This function converts the request body
into
# a JSON object, defines the request headers, posts the request, and
# returns the response.

def post(url1, contentType, accept, accessToken, body):
    headers = {"Accept": accept,
               "Authorization": "bearer " + accessToken,
               "Content-Type": contentType }
    # Convert the request body to a JSON object.
    reqBody = json.loads(body)
    # Post the request.
    req = sess.post(url1, json=reqBody, headers=headers)
    return req;

# Define the PUT function. This function converts the request body into
# a JSON object, defines the request headers, and submits the request.
# The conditionalPutKey and conditionalPutValue fields are used to
# identify a specific state of the resource. See "Updating Objects with
# The put() Function".

def put(url1, contentType, accept, accessToken, \
        conditionalPutKey, conditionalPutValue, body):
    headers = {"Accept": accept,
               "Authorization": "bearer " + accessToken,
               "Content-Type": contentType,
               conditionalPutKey : conditionalPutValue }
```

```
    if (contentType != "text/plain"):
        reqBody = json.loads(body)
        res = sess.put(url1, json=reqBody, headers=headers)
    else:
        res = sess.put(url1, body, headers=headers)
    return res

# Specify the URL, user ID, and password required to access your
server.
baseUrl1 = 'host-URL'
userid = 'user-ID'
password = 'password'

# Get an access token.
accessToken1 = getAccessToken(baseUrl1);

# Create a connection session.
sess = requests.Session()

# Add statements to submit your API requests here.
```

See Also

- SAS Viya REST APIs: Authentication to SAS Viya
- SAS Viya REST APIs: Authorization
- The Client ID and Secret

3

Define and Execute a Decision Test

Introduction

This example defines and executes a decision test. To test a decision by using the API, you create a score definition and a score execution.

How To

1. Get an access token by submitting a GET request to the `/SASLogon/oauth/token` resource. The example described in Chapter 2, “Define Basic Methods and Get an Access Token” includes the code required to get an access token. Prepend that code to the code for this example.
2. Specify the ID of the decision that you want to test.
3. Define the URL to the decision object.
4. Define the request body for the score definition. Define the following parameters:

name
specifies a name for the test.

objectDescriptor
specifies the name, object type, and URI for the decision that you are testing.

inputData
specifies the input data for the test. You can specify an existing input table or define the input data set in the request body. To specify an existing input table, specify the input type `CASTable`, specify the name of the table, and the library and server where the table resides.

To define an input data set in the request body, specify `Inline` as the input type, and enter the DATA step code to create the input data in the `code` property. For an example of using inline data, see Example 2: Specifying the type of inputData as Inline in the Score Definitions API examples on GitHub.

mappings
maps each of the decision's input-only and input-output variables to a column in the input table or to a static value.

variableName
specifies the name of a decision variable.

mappingType
specifies where the decision variable's value comes from. For variables whose value comes from the input table, specify `datasource`. For variables to which you want to assign a fixed value, specify `static`.

mappingValue
specifies the input table column or the static value to map to the decision variable that is specified by the `variableName` property. For variables whose mapping type is `datasource`, specify the column in the input table from which the decision variable gets its value.

properties
specifies the server and library where you want the test to write the output tables. You can specify a base name for the output tables with the `tableBaseName` property. If you do not specify a base name, the API uses *`test-name_decision-name`* for the base name.

Note: These parameters are described in the schema for the Score Definitions API at https://developer.sas.com/apis/rest/DecisionManagement/#schemas-7.

5 Create the score definition by submitting a POST request to `/scoreDefinitions/definitions`.

6 Use the `json()` method to convert the response to a JSON object, and retrieve the ID of the score definition.

7 Create the request body for the score execution. Specify the following parameters:

name
specifies a name for the score execution request. This name is displayed when you view the log of API calls on your network.

scoreDefinitionId
specifies the score definition ID that was returned by the score definition request.

hints
specifies the library and table name for the input table, and specifies the URI to the decision object.

8 Execute the score code by submitting a POST request to `/scoreExecution/executions`.

Example Code

```
<< Include the code in "Define Basic Methods and Get an Access Token".
>>
<< That code is required to successfully execute this example.
>>

# Specify the ID of the decision that you want to test, for example,
# "e289b21b-4be1-4739-9313-639b9629cb42".
decision_ID = "decision-ID"

# Define the URI to the decision object.
objectUri = "/decisions/flows/" + decision_ID

# Define the request body for the score definition.
# Modify these values to match the data for your test.

scoreDefinitionBody = '''
   {
      "name": "test_name",
      "objectDescriptor":
         {
         "name":"decision_name",
         "type": "decision",
         "uri": "%s"
         },
      "inputData":
         {
            "type": "CASTable",
            "tableName": "HMEQ_TRAIN",
            "libraryName": "Public",
            "serverName": "cas-shared-default"
          },
      "mappings": [
         {
            "variableName": "value",
            "mappingType": "datasource",
            "mappingValue": "value"
         },
         {
            "variableName": "debtinc",
            "mappingType": "datasource",
            "mappingValue": "debtinc"
         },
```

```
            {
               "variableName": "derog",
               "mappingType": "datasource",
               "mappingValue": "derog"
            },
            {
               "variableName": "delinq",
               "mappingType": "datasource",
               "mappingValue": "delinq"
            }
         ],
         "properties": {
            "outputServerName": "cas-shared-default",
            "outputLibraryName": "Public",
            "tableBaseName": "decisionTest_results"
         }
}
''' %(objectUri)

# Create the score definition.
requestUrl= baseUrl1 + "/scoreDefinitions/definitions"
scoreCreationResponse = post(requestUrl,
                        "application/vnd.sas.score.definition+json",
                        "application/vnd.sas.score.definition+json",
                        accessToken1, scoreDefinitionBody);
print ("creation response =", scoreCreationResponse, end='\n\n')
print ("creation response content =", "\n",
       json.dumps(json.loads(scoreCreationResponse.content),
       indent=4), end='\n\n')

# Retrieve the ID of the score definition from the response.
responseObject=scoreCreationResponse.json()
scoreDefinitionID = responseObject['id']

# Create the request body for executing the score defintion.
# The request name is a descriptive name that is displayed
# when you view the log of API calls on your network.
scoreExecutionBody = '''
   {
      "name": "request-name",
      "scoreDefinitionId": "%s",
      "hints": {
         "inputLibraryName": "Public",
         "inputTableName": "HMEQ_TRAIN",
         "objectURI": "%s"
      }
}
''' %(scoreDefinitionID, objectUri)

# Execute the score code. (Create a score execution.)
requestUrl= baseUrl1 + "/scoreExecution/executions"
scoreExecutionResponse = post(requestUrl,
                        "application/vnd.sas.score.execution.request
+json",
                        "application/vnd.sas.score.execution+json",
                        accessToken1, scoreExecutionBody);
```

```
print("execute response = ", scoreExecutionResponse, end="\n\n")

# Print the response body in readable format.
printable = json.loads(scoreExecutionResponse.content)
print("scoreExecutionResponse body = ", "\n",
    json.dumps(printable, indent=4), end="\n\n")
```

Responses

The response bodies that are returned by the POST requests in this example are byte objects. To make the responses readable, they are converted to dictionaries.

The POST request to the score definition resource returns the following response:

```
creationResponse =  <Response [201]>

creation response content =
 {
    "creationTimeStamp": "2019-08-06T18:40:12.025Z",
    "modifiedTimeStamp": "2019-08-06T18:40:12.025Z",
    "createdBy": "user-ID",
    "modifiedBy": "user-ID",
    "id": "e12d0f5f-8a0a-4fa1-bd1a-811edf8390c7",
    "name": "test_1",
    "objectDescriptor": {
        "name": "Evaluate_Loans",
        "type": "decision",
        "uri": "/decisions/flows/e289b21b-4be1-4739-9313-639b9629cb42"
    },
    "inputData": {
        "type": "CASTable",
        "serverName": "cas-shared-default",
        "tableName": "HMEQ_TRAIN",
        "libraryName": "Public"
    },
    "mappings": [
        {
            "variableName": "value",
            "mappingType": "datasource",
            "mappingValue": "value"
        },
        {
            "variableName": "debtinc",
            "mappingType": "datasource",
            "mappingValue": "debtinc"
        },
        {
            "variableName": "derog",
            "mappingType": "datasource",
            "mappingValue": "derog"
        },
        {
```

```
            "variableName": "delinq",
            "mappingType": "datasource",
            "mappingValue": "delinq"
        }
    ],
    "properties": {
        "tableBaseName": "eval_loans_results",
        "outputServerName": "cas-shared-default",
        "outputLibraryName": "Public"
    },
    "links": [
             <<lines deleted>>
    ],
    "version": 1
}
```

The POST request to the score execution resource returns the following response:

```
execute response =  <Response [201]>

scoreExecutionResponse body =
 {
    "creationTimeStamp": "2019-08-06T18:40:16.578Z",
    "modifiedTimeStamp": "2019-08-06T18:40:16.578Z",
    "createdBy": "user-ID",
    "modifiedBy": "user-ID",
    "id": "f4ce301b-8338-4462-b56a-21c7920a17b7",
    "scoreExecutionRequest": {
        "type": "scoreDefinition",
        "name": "Evaluate_Loans_test",
        "hints": {
            "inputLibraryName": "Public",
            "inputTableName": "HMEQ_TRAIN",
            "objectURI": "/decisions/flows/e289b21b-4be1-4739-9313-639b9629cb42"
        },
        "scoreDefinitionId": "0f3843c3-1e84-4bc6-8e4e-551909a9bb84",
        "version": 1
    },
    "state": "running",
    "outputTable": {
        "tableName": "eval_loans_results_2019-08-06_18-40-14_output",
        "libraryName": "Public",
        "serverName": "cas-shared-default"
    },
    "codeFileUri": "/files/files/7aef9a58-eeab-4352-8ed2-22c7c669719e",
    "results": {
        "jobId": "e0fee109-6c3b-4b29-9b08-4acd01203c37"
    },
    "links": [
             <<lines deleted>>
    ],
    "version": 1
}
```

See Also

- Create a new score definition
- Creating a Score Definition
- Create a new score execution
- Creating a Score Execution

4

Lock Treatments

Introduction

This example retrieves the treatment groups that match the specified criteria and creates new, numbered and locked versions of the individual treatments in the treatment groups.

How To

1. Get an access token by submitting a GET request to the `/SASLogon/oauth/token` resource. The example described in Chapter 2, “Define Basic Methods and Get an Access Token” includes the code required to get an access token. Prepend that code to the code for this example.
2. Get the collection of treatment groups for which you want to lock the individual treatments.

   Modify the search criteria in the initial GET request to the `treatmentDefinitions/definitionGroups` resource so that the request returns the collection of treatment groups for which you want to lock the treatments.

> **TIP** Usage Notes for the Treatment Definitions API lists the members of the collection that you use to filter and sort the collection. For additional information, see SAS REST APIs: Filtering.

3. Use the `json.loads` method to convert the collection to a JSON object. This conversion enables you to access the key-value pairs in the JSON object by using the key name.
4. Define the content type for submitting GET requests to retrieve treatment group definitions.
5. For each treatment group in the collection that was returned in Step 2, complete these steps:
   a. Retrieve the treatment group ID from the collection that was returned in Step 2.
   b. Get the treatment group definition by submitting a GET request to `/treatmentDefinitions/definitionGroups/`*`group-ID`*. The response body is a byte object that contains the definition of the treatment group.
   c. Use the `json.loads()` method to convert the treatment group definition to a string.
   d. Set the Accept type for retrieving individual treatments.
   e. For each treatment in the treatment group, complete these steps:
      1. Retrieve the treatment definition ID from the `members` property of the treatment group definition that was returned in Step 5b.
      2. Get the treatment definition by submitting a GET request to `/treatmentDefinitions/definitions/`*`definition-ID`*.
      3. Create a new revision of the treatment definition by submitting a POST request to `/treatmentDefinitions/definitions/`*`definition-ID`*`/revisions`. Specify the response body that was returned in Step 5e.ii as the request body for this POST request.

         By default, this POST request creates a new minor version. To specify a new major version, include the `?revisionType=major` query parameter.

Example Code

```
<< Include the code in "Define Basic Methods and GEt an Access Token.
>>
<< That code is required to successfully execute this example.
>>

# Get the treatment groups based on filter criteria.
# Modify the filter criteria as needed for your application.
requestUrl= baseUrl1 + \
```

```
    "/treatmentDefinitions/definitionGroups?
filter=eq(name,'hmeq_Treat_Group')"
responseObj,responseHeaders =  get(requestUrl, accessToken1,
                               "application/vnd.sas.collection+json");
responseObj = json.loads(responseObj)

# Define the treatment defintions URL and the accept type for requests.
groupDefUrl = "/treatmentDefinitions/definitionGroups/"
contentType = "application/vnd.sas.treatment.definition+json"

# For each treatment group ID in the list...
for item in responseObj['items']:
    groupID = item['id']

    # Get the treatment group definition.
    acceptType = "application/vnd.sas.treatment.definition.group+json"
    requestUrl=baseUrl1 + groupDefUrl + groupID
    responseBody,responseHeaders = get(requestUrl, accessToken1,
acceptType)

    # Convert the response to string.
    responseBodyJson = json.loads(responseBody)

    # Print the response body in readable format.
    print ("treatment group = ", "\n", json.dumps(responseBodyJson,
           indent=4), end='\n\n')

    # Set the accept type for getting the individual
    # treatment definitions.
    acceptType = "application/json"

    # For each treatment in the treatment group...
    for member in responseBodyJson['members']:
        definitionId = member['definitionId']

        # Get the treatment definition.
        requestUrl = baseUrl1 + "/treatmentDefinitions/definitions/" +
\
                     definitionId
        requestBody, respHeaders = get(requestUrl, accessToken1,
acceptType)

        # Create a new version of the treatment definition.
        requestUrl = baseUrl1 + "/treatmentDefinitions/definitions/" \
           + definitionId + "/revisions?revisionType=major"
        revisionResponse = post(requestUrl, contentType, acceptType, \
            accessToken1, requestBody)

        print("definition name = ", member['definitionName'], end='\n')
        print("definition ID = ", definitionId, end='\n')
        print ("revisionResponse = ", revisionResponse, end='\n')
```

Responses

Retrieving the Treatment Group

The `members` property of the treatment group lists the name and ID of each treatment that is included in the group.


```
treatment group =
 {
    "creationTimeStamp": "2019-07-16T03:24:27.181Z",
    "modifiedTimeStamp": "2019-08-12T02:23:32.630Z",
    "createdBy": "edmdev",
    "modifiedBy": "edmdev",
    "id": "8daf2af7-ecde-4c83-ba0f-eab69bcf4860",
    "name": "hmeq_Treat_Group",
    "majorRevision": 1,
    "minorRevision": 16,
    "locked": false,
    "members": [
        {
            "definitionId": "4dddd0c1-4ee9-48ef-983f-e61a837d6961",
            "definitionName": "hmeq_treatment",
            "definitionRevisionId": "2e5d8bdd-46d6-42a2-9853-2cce9ae40052",
            "definitionRevisionName": "1.0"
        },
        {
            "definitionId": "ef5d6043-2212-4173-bb0f-949824386e12",
            "definitionName": "hmeq_treatment_2",
            "definitionRevisionId": "92d44ca7-c397-4783-8f30-0c3798ca09dd",
            "definitionRevisionName": "1.0"
        },
        {
            "definitionId": "3827bdd0-5120-4087-83f2-d54b715d08e0",
            "definitionName": "hmeq_treatment_3",
            "definitionRevisionId": "ef970024-84b3-4256-8fec-c57da83cf1d4",
            "definitionRevisionName": "1.0"
        },
        {
            "definitionId": "7dc18c98-1959-4132-9a52-4a4d920660fc",
            "definitionName": "hmeq_treatment_4",
            "definitionRevisionId": "4fa93e91-4b63-4b61-a29c-c2afe52f66ac",
            "definitionRevisionName": "1.0"
        }
    ],
    "links": [
           << lines deleted >>
            ],
    "version": 1,
```

```
    "status": "valid"
}
```

Locking and Creating Versions of Individual Treatments

The POST request that creates a new version of the individual treatments returns response code 201 for each request that is successful.

```
definition name =  hmeq_treatment
definition ID =  4dddd0c1-4ee9-48ef-983f-e61a837d6961
revisionResponse =  <Response [201]>

definition name =  hmeq_treatment_2
definition ID =  ef5d6043-2212-4173-bb0f-949824386e12
revisionResponse =  <Response [201]>

definition name =  hmeq_treatment_3
definition ID =  3827bdd0-5120-4087-83f2-d54b715d08e0
revisionResponse =  <Response [201]>

definition name =  hmeq_treatment_4
definition ID =  7dc18c98-1959-4132-9a52-4a4d920660fc
revisionResponse =  <Response [201]>
```

See Also

- Create a new revision of a treatment definition
- Lock the current revision and create a new revision of a Treatment Definition

5

Activate Treatment Groups

Introduction

This example activates the treatment groups that meet a specified search criteria.

How To

1 Get an access token by submitting a GET request to the `/SASLogon/oauth/token` resource. The example described in Chapter 2, "Define Basic Methods and Get an Access Token" includes the code required to get an access token. Prepend that code to the code for this example.

2 Get the collection of treatment group definitions that you want to activate.

Modify the search criteria in the initial GET request to the `/treatmentDefinitions/definitionGroups` resource so that it returns the collection of treatment groups that you want to activate.

> **TIP** The Usage Notes for the Treatment Definitions API list the members of the collection that you use to filter and sort the collection. For additional information, see SAS REST APIs: Filtering.

3 Use the `json.loads` method to convert the collection to a JSON object. This enables you to access the key-value pairs in the JSON by using the key name.

4 Define the URL for the treatment group definitions resource and the accept type for submitting GET requests to this resource.

5 For each treatment group in the collection that was returned in Step 2, complete these steps:

   a Retrieve the treatment group ID from the collection returned in Step 2 on page 25.

   b Get the treatment group definition by submitting a GET request to `/treatmentDefinitions/definitionGroups/group-ID`. The response body is a byte object that contains the definition of the treatment group.

   c Use the `decode()` method to convert the treatment group definition to a string. Specify this string as the requestBody parameter to the POST request in Step 5d.

   d Lock the current version of the treatment group, and create a new current version by submitting a POST request to `/treatmentDefinitions/definitionGroups/group-ID/revisions`.

   > **TIP** Creating a new version of the treatment group generates a new ETag value for the group.

   e Get the treatment group definition again, and extract the new ETag value for the group from the response header. For information about ETag values, see "Updating Objects with the put() Function" on page 5.

   f Get the list of revisions for the treatment group by submitting a GET request to `/treatmentDefinitions/definitionGroups/group-ID/revisions`. Specify the following parameters:

      `start=0&limit=100`
      requests the first 100 revisions. By default, the request returns only 10 revisions. If your treatment group has more than 10 revisions, specify a higher limit.

      `sortBy=sortBy=majorRevision:descending,minorRevision:descending`
      sorts the list of revisions in descending order by revision numbers. This sort order ensures that the query returns the most recently locked version as the second item in the list.

   g Use the `json.loads` method to convert the response body to a JSON object.

   h Retrieve the ID of the second revision in the list that was returned in Step 5f.

   i Activate the revision of the treatment group by submitting a PUT request to `/treatmentDefinitions/definitionGroups/group-ID/revisions/`

revision-ID/active. Specify If-Match as the conditional key and the group ETag value from Step 5e as the value of the key.

Example Code

```
<< Include the code in "Define Basic Methods and Get an Access Token".
>>
<< That code is required to successfully execute this example.
>>

# Get the treatment groups based on filter criteria.
# Modify the filter criteria as needed for your application.

requestUrl= baseUrl1 + \
    "/treatmentDefinitions/definitionGroups?
filter=eq(name,'Treat_Group_1')"
responseObj,responseHeaders =  get(requestUrl, accessToken1, \
                               "application/vnd.sas.collection+json");
responseObj = json.loads(responseObj)

# Print the response body in readable format.
print ("responseObj = ", "\n", json.dumps(responseObj,
       indent=4), end='\n\n')

# Define the treatment defintions URL and the accept type for requests.
acceptType = "application/vnd.sas.treatment.definition.group+json"
groupDefUrl = "/treatmentDefinitions/definitionGroups/"

# For each treatment group ID in the list...
for item in responseObj['items']:
    groupID = item['id']

    # Get the treatment group definition.
    requestUrl=baseUrl1 + groupDefUrl + groupID
    responseBody,responseHeaders = get(requestUrl, accessToken1,
acceptType)
    responseBodyGrpJson = json.loads(responseBody)

    print ("group responseBody = ", "\n",
json.dumps(responseBodyGrpJson,
           indent=4), end='\n\n')
    print ("group responseHeaders = ", "\n", responseHeaders, end='\n
\n')

    # Convert the treatment group definition to a string.
    # This string is passed as the request body when you create a
    # new revision of the treatment group.
    requestBody = responseBody.decode()

    # Lock the current version and create a new current version.
    requestUrl = baseUrl1 + groupDefUrl + groupID + "/revisions"
```

```
    responseObj = post(requestUrl,
                    "application/vnd.sas.treatment.definition.group
+json",
                    acceptType, accessToken1, requestBody)
    print ("new version response = ", responseObj)

    # Get the treatment group definition again and extract
    # the new ETag value for the group.
    requestUrl=baseUrl1 + groupDefUrl + groupID
    responseBody,responseHeaders = get(requestUrl, accessToken1,
acceptType)
    responseBodyGrpJson = json.loads(responseBody)
    groupEtag=responseHeaders['ETag']

    # Get the list of revisions for the treatment group with
ID=groupID,
    # sorted in descending order based on the major and minor revision
    # numbers. If your treatment group has more than 100 revisions,
    # modify the limit in the following URL.

    requestUrl=baseUrl1 + groupDefUrl + groupID + "/revisions" + \
     "?
start=0&limit=100&sortBy=major Revision:descending,minorRevision:descend
ing"
    responseBody,responseHeaders = get(requestUrl, accessToken1,
                                       "application/json")
    responseBodyJson = json.loads(responseBody)

    print("number of revisions = ", responseBodyJson['count'], end='\n
\n')
    printable = json.dumps(responseBodyJson, indent=4)
    print("revision list responseBody = ", '\n\n',
          printable, end='\n\n')

    # Get the revision ID.
    revisionID = responseBodyJson['items'][1]['id']

    print ("Activating treatment group ",
           responseBodyJson['items'][1]['name'], "\n",
           "Revision ", revisionID, end='\n\n')

    # Activate the locked version of the treatment group.

    requestUrl=baseUrl1 + groupDefUrl + groupID + "/active"
    response = put(requestUrl,"text/plain", acceptType,
                   accessToken1, "If-Match", groupEtag, revisionID)
    print ("Activation response = ", response)
```

Responses

Getting the Treatment Group Collection

The treatment group collection returned by the initial GET request contains, in this case, only one treatment definition group. The group name is “Treat_Group_1”, and the group ID is `8daf2af7-ecde-4c83-ba0f-eab69bcf4860`.

```
responseObj =
 {
    "links": [
                <<lines deleted>>
    ],
    "name": "treatmentDefinitionGroups",
    "accept": "application/vnd.sas.summary",
    "start": 0,
    "count": 1,
    "items": [
        {
            "creationTimeStamp": "2019-07-16T03:24:27.181Z",
            "modifiedTimeStamp": "2019-08-09T17:35:43.584Z",
            "createdBy": "user-ID",
            "modifiedBy": "user-ID",
            "id": "8daf2af7-ecde-4c83-ba0f-eab69bcf4860",
            "type": "treatmentGroup",
            "name": "Treat_Group_1",
            "links": [
                      <<lines deleted>>
            ],
            "version": 1
        }
    ],
    "limit": 10,
    "version": 2
}
```

Getting a Treatment Group

You can extract the IDs of the individual treatment groups from the collection and use the ID to retrieve the treatment group definition. The GET request for the treatment group with the ID `8daf2af7-ecde-4c83-ba0f-eab69bcf4860` returned the following response:

```
group responseBody =
 {
```

```
    "creationTimeStamp": "2019-07-16T03:24:27.181Z",
    "modifiedTimeStamp": "2019-08-09T17:35:43.584Z",
    "createdBy": "user-ID",
    "modifiedBy": "user-ID",
    "id": "8daf2af7-ecde-4c83-ba0f-eab69bcf4860",
    "name": "Treat_Group_1",
    "majorRevision": 1,
    "minorRevision": 15,
    "locked": false,
    "members": [
        {
            "definitionId": "4dddd0c1-4ee9-48ef-983f-e61a837d6961",
            "definitionName": "hmeq_treatment",
            "definitionRevisionId": "2e5d8bdd-46d6-42a2-9853-2cce9ae40052",
            "definitionRevisionName": "1.0"
        },
        {
            "definitionId": "ef5d6043-2212-4173-bb0f-949824386e12",
            "definitionName": "hmeq_treatment_2",
            "definitionRevisionId": "92d44ca7-c397-4783-8f30-0c3798ca09dd",
            "definitionRevisionName": "1.0"
        },
        {
            "definitionId": "3827bdd0-5120-4087-83f2-d54b715d08e0",
            "definitionName": "hmeq_treatment_3",
            "definitionRevisionId": "ef970024-84b3-4256-8fec-c57da83cf1d4",
            "definitionRevisionName": "1.0"
        },
        {
            "definitionId": "7dc18c98-1959-4132-9a52-4a4d920660fc",
            "definitionName": "hmeq_treatment_4",
            "definitionRevisionId": "4fa93e91-4b63-4b61-a29c-c2afe52f66ac",
            "definitionRevisionName": "1.0"
        }
    ],
    "links": [
            <<lines deleted>>
    ],
    "version": 1,
    "status": "valid"
}
```


The response header for the GET request of the treatment group contains the ETag for the group.

```
group responseHeaders =
Date: Fri, 09 Aug 2019 17:38:25 GMT
Server: Apache/2.4
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline'...
ETag: "jz4e5o80"
Last-Modified: Fri, 09 Aug 2019 17:35:43 GMT
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: SAMEORIGIN
Content-Type: application/vnd.sas.treatment.definition.group+json
X-Content-Type-Options: nosniff
```

```
X-XSS-Protection: 1; mode=block
Vary: User-Agent
Connection: close
Transfer-Encoding: chunked
```

Getting the Treatment Group Revision List

The GET request for the list of revisions for the treatment group returned a list of 17 revisions. Each revision has a unique ID. The response for the first two revisions is shown below.

```
number of revisions =  17

revision list responseBody =
 {
    "links": [
        <<lines deleted>>
    ],
    "name": "treatmentDefinitionGroupRevisions",
    "accept": "application/vnd.sas.treatment.definition.group",
    "start": 0,
    "count": 61,
    "items": [
        {
            "creationTimeStamp": "2019-08-09T17:32:36.952Z",
            "modifiedTimeStamp": "2019-08-09T17:32:36.952Z",
            "createdBy": "user-ID",
            "modifiedBy": "user-ID",
            "id": "a1c20d7f-4557-4b66-a69b-fc46857d876d",
            "type": "treatmentGroup",
            "name": "Treat_Group_1",
            "links": [
                lines deleted
            ],
            "version": 1
        },
        {
            "creationTimeStamp": "2019-08-09T17:31:29.883Z",
            "modifiedTimeStamp": "2019-08-09T17:31:29.883Z",
            "createdBy": "edmdev",
            "modifiedBy": "edmdev",
            "id": "b4c19e0d-3342-4994-80a1-72f6017c42e8",
            "type": "treatmentGroup",
            "name": "Treat_Group_1",
            "links": [
                <<lines deleted>>
            ],
            "version": 1
        }
    ],
    "limit": 100,
    "version": 2
 }
```

```
<<lines for additional revisions deleted>>
```

Activating the Treatment Group Revision

If the treatment group is activated, the request returns the HTTP response code 200.

```
Activating treatment group  Treat_Group_1
 Revision  b4c19e0d-3342-4994-80a1-72f6017c42e8

Activation response = <Response [200]>
```

See Also

- Get a collection of the treatment definition groups
- Get a treatmentDefinitionGroup
- Create a new revision of a treatmentDefinitionGroup
- Lock the current revision and create a new revision of a Treatment Definition Group
- Get a collection of the revisions of the specified treatmentDefinitionGroup
- Get a revision of the specified treatmentDefinitionGroup
- Activate a revision of a treatmentDefinitionGroup
- Activate a Treatment Definition Group revision
- SAS REST APIs: Sorting
- SAS REST APIs: Filtering
- Usage Notes for the Treatment Definitions API (see the topic on “Sorting and Filtering”)

6

Publish a Decision to the maslocal Destination

Introduction

This example locks the current version of a decision, creates a new unlocked version, and then publishes the locked version to the `maslocal` destination.

The `maslocal` destination is the default SAS Micro Analytic Service destination. However, your administrator might define additional SAS Micro Analytic Service destinations. For more information, see "Configuring Publishing Destinations" in *SAS Intelligent Decisioning: Administrator's Guide*.

How To

1 Get an access token by submitting a GET request to the `/SASLogon/oauth/token` resource. The example described in Chapter 2, "Define Basic Methods and Get an Access Token" includes the code required to get an access token. Prepend that code to the code for this example.

2 Specify the object ID of the decision that you want to publish. You can retrieve decision information by submitting GET requests to the `/flows` resource. You can use query parameters to narrow the results that are returned by the request. For more information, see Get a list of all decisions.

   This ID is also shown on the **Properties** tab when you open the decision in SAS Intelligent Decisioning.

3 Specify a name for the publishing request. This name is a descriptive name that is displayed when you view the log of API calls on your network.

4 Specify the name of the decision that you want to publish.

5 Get the contents of the decision by submitting a GET request to the `/decisions/flows/`*`object-ID`* resource.

6 Lock the current version of the decision (which becomes the new numbered version), and create a new, unlocked current version by submitting a POST request to the `/decisions/flows/`*`decision-ID`*`/revisions` resource. You can specify the `versionType` parameter to control whether a major version or a minor version is created.

   After this request executes, the object ID now points to the locked and numbered version of the decision.

7 Get the generated DS2 code for the locked version of the decision by submitting a GET request to the `/decisions/flows/`*`decision-ID`*`/code` resource. Because SAS Micro Analytic Service destinations do not generated rule-fired information or path-tracking information, include the following query parameters:

   ```
   ?lookupMode=inline&traversedPathFlag=false\
   &isGeneratingRuleFiredColumn=false&codeTarget=microAnalyticService
   ```

   The generated code is returned as a byte object.

8 Convert the byte object to a string, and escape newline characters and double quotation marks. Specify the resulting character string as the code property for the model in the publish request in Step 10.

9 Define the URL to the source code of the locked version of the decision.

10 Define the request body for the publish request. Specify the following parameters:

   type
   : specifies the destination type. For SAS Micro Analytic Service destinations, this value is `microAnalyticService`.

   destinationName
   : specifies the name of the destination. The name of the default SAS Micro Analytic Service destination is `maslocal`.

   name
   : specifies a name for the publishing request. This name is displayed when you view the log of API calls on your network.

   modelContents
   : provides information about the decisions that you are publishing.

     code
     : specifies the generated code for the decision.

codeType
specifies whether the source code is DS2 code (`ds2`) or DATA step code (`datastep`).

modelName
specifies the name of the decision that you are publishing.

> **TIP** The published name of the decision is defined in the generated code for the decision.

publishLevel
specifies the object type that you are publishing.

sourceUri
specifies the URI to the generated code for the decision.

11 Publish the decision by submitting a POST request to the `/modelPublish/models` resource.

Example Code

```
<< Include the code in "Define Basic Methods and Get an Access Token".
>>
<< That code is required to successfully execute this example.
>>

# Specify the ID of the decision that you want to publish, for example,
# "e289b21b-4be1-4739-9313-639b9629cb42".
objectID = "object-ID"

# Specify a name for the publishing request, for example,
# "publish_SID_decision".
# The request name is a descriptive name that is displayed
# when you view the log of API calls on your network.
requestName = "request-name"

# Specify the name of your decision, for example,
# "Evaluate_Loans".
modelName = "decision-name"

# Get the contents of the decision.
requestUrl= baseUrl1 + "/decisions/flows/" + objectID
decisionContent,rspheads = get(requestUrl, accessToken1, "application/
json")

contentStr = json.loads(decisionContent)
versionToLock=str(contentStr['majorRevision']) + "." + \
              str(contentStr['minorRevision'])
print("Locking version ", versionToLock, end='\n\n')
```

```
# Lock the current version of the decision and create a new,
# numbered, major version.
requestUrl= baseUrl1 + "/decisions/flows/" + objectID + \
            "/revisions?revisionType=major"
lockResponse= post (requestUrl, "application/vnd.sas.decision+json",
                        "application/vnd.sas.decision+json",
                        accessToken1, decisionContent)
print ("lock response = ", lockResponse, end='\n\n')
# The object ID now points to the locked version of the decision.

respContent = json.loads(lockResponse.content)
newVersion=str(respContent['majorRevision']) + "." + \
           str(respContent['minorRevision'])
print("Created new version", newVersion, "  ID =",
       respContent['id'], end='\n\n')

# Get the generated DS2 code for the locked version of the decision.
requestUrl= baseUrl1 + "/decisions/flows/" + objectID + \
"/code?lookupMode=inline&traversedPathFlag=false\
&isGeneratingRuleFiredColumn=false&codeTarget=microAnalyticService"

decisionCodeContent,rspheads = get(requestUrl, accessToken1,
    "text/vnd.sas.source.ds2")

# Convert the byte object to a string.
decisionCodeString= decisionCodeContent.decode()

# Escape control characters and double quotation marks.
decisionCodeSource = decisionCodeString.replace("\\","\\\
\").replace("\n", "\\n")\
.replace('"', '\\"')

# Define the URI to the source code for the decision.
sourceUri = "/decisions/flows/" + objectID + "/code"

# Create the request body.
publishRequestBody = '''
{
   "type": "microAnalyticService",
   "destinationName": "maslocal",
   "name": "%s",
   "modelContents": [
      {
      "code": "%s",
      "codeType": "ds2",
      "modelName": "%s",
      "publishLevel": "decision",
      "analyticStoreUri": [],
      "analyticStores": [],
      "sourceUri": "%s"
      }
    ]
}
''' % (requestName, decisionCodeSource, modelName, sourceUri)

# Publish the decision.
```

```
print("Publishing verison", versionToLock, end='\n\n')
requestUrl= baseUrl1 + "/modelPublish/models"
publishResponse= post(requestUrl,
                      "application/vnd.sas.models.publishing.request
+json",
                      "application/vnd.sas.models.publishing.publish
+json",
                      accessToken1, publishRequestBody);
print ("publish response = ", publishResponse)
```

Responses

This example produces the following output:

```
Locking version 39.0

lock response =  <Response[201]>

Created new version 40.0   ID = 62b91159-bd73-4984-ad55-89bd323d0f92

Publishing version 39.0

publish response =  <Response[201]>
```

See Also

- Get decision content
- Get a decision revision
- Get decision code
- Decisions API
- Publish models
- Publish a Model to a Destination

7

Execute a Published Decision

Introduction

This example executes a decision that has been published to the maslocal destination.

How To

1 Get an access token by submitting a GET request to the `/SASLogon/oauth/token` resource. The example described in Chapter 2, "Define Basic Methods and Get an Access Token" includes the code required to get an access token. Prepend that code to the code for this example.

2 Create the input body for the request. The input body defines the input data that you want to use to execute the decision. Specify the name and value for each input-only and input-output variable.

> **IMPORTANT** If the SAS Micro Analytic Service configuration property `service.removetrailingunderscoresfrominputs` and the SAS Intelligent Decisioning property `sas.decisions.masnode.removeTrailingUnderscoresFromInput` are

not defined or are set to **False**, add an underscore to the name of each input variable. If these options are set to **True**, do not add underscores. Your administrator can add this property to the `supplementalProperties` section in the `sas.microanalyticservice.system` configuration definition in SAS Environment Manager. The default value for these options is **False**. For additional information, see "sas.microanalyticservice.system: supplementalProperties" in *SAS Micro Analytic Service: Programming and Administration Guide* and "sas.decisions.masnode.removeTrailingUnderscoresFromInput" in *SAS Intelligent Decisioning: Administrator's Guide*.

TIP To specify a missing value for a variable, specify the variable name without specifying the value. For example: `{"name": "debtinc_"}`

3 Define the content type and the accept type for the publishing request.

4 Specify the module ID of the published decision. You specify this ID as the model name when you publish the decision by using the API, or in the **Published Name** column in the Publish Decisions window in SAS Intelligent Decisioning. This name is also displayed on the **History** tab for a decision in SAS Intelligent Decisioning.

5 Execute the decision by submitting a POST request to `/microanalyticScore/modules/`*`module_ID`*`/steps/execute`.

Each module that is published to SAS Micro Analytic Service contains a step named `execute`. When you post the request, this `execute` step is executed.

Example Code

```
<< Include the code in "Define Basic Methods and Get an Access Token".
>>
<< That code is required to successfully execute this example.
>>

# Create the request body. The request body specifies the
# input values required by the decision.
# Modify these key-value pairs for your decision.
# Include underscores if needed.

requestBody = '''
{
"inputs" : [
  {"name": "debtinc_", "value" : 37.1136},
  {"name": "delinq_", "value" : 0},
  {"name": "derog_", "value" : 4},
  {"name": "value_", "value" : 60850}
 ]
}
```

```
'''

# Define the content and accept types for the request header.
contentType = "application/json"
acceptType = "application/json"

# Specify the module ID of the published decision, for example,
# "evaluate_loans24_0".
moduleID = "module-ID"

# Define the request URL.
masModuleUrl = "/microanalyticScore/modules/" + moduleID
requestUrl = baseUrl1 + masModuleUrl + "/steps/execute"

# Execute the decision.
masExecutionResponse = post(requestUrl, contentType,
                        acceptType, accessToken1, requestBody)

# Display the response.
print ("response=", masExecutionResponse, end='\n\n')
print ("response content=", "\n",
       json.dumps(json.loads(masExecutionResponse.content),
       indent=4), end='\n\n')
```

Responses

The response content includes the output values from the decision. In this example, all of the variables in the decision are both input and output variables except for the `review` variable. The `review` variable is an output-only variable.

```
response = <Response [201]>

response content=
 {
    "links": [],
    "version": 2,
    "moduleId": "evaluate_loans24_0",
    "stepId": "execute",
    "executionState": "completed",
    "outputs": [
        {
            "name": "debtinc",
            "value": 37.1136
        },
        {
            "name": "delinq",
            "value": 0.0
        },
        {
            "name": "derog",
            "value": 4.0
        },
```

```
            {
                "name": "review"
            },
            {
                "name": "value",
                "value": 60850.0
            }
        ]
    }
```

See Also

Execute a step

8

Update Subject Contact History

Introduction

When you use a decision to determine the treatments for which a subject is eligible, the decision returns the list of treatments in the `treatmentsForConsideration` property. If the decision includes a record contacts node, this node creates a subject contact history record that includes a response tracking code. The subject contact history record contains the information that you want to track, which typically includes the treatments for which the subject is eligible. You can use the tracking code to update the subject contact history record with information about the subject's response to the treatments.

For more information about treatments and record contacts nodes, see the following topics:

- "About Treatments and Decisions" in *SAS Intelligent Decisioning: User's Guide*
- "Example: A Decision That Includes a Treatment Group" in *SAS Intelligent Decisioning: User's Guide*
- "Add a Record Contacts Node" in *SAS Intelligent Decisioning: User's Guide*

This example updates a subject contact history record. It updates both record-level and treatment-level properties. This example assumes that a customer service representative or a calling application provides the tracking code of the subject contact record, the treatment ID of the treatment that needs to be updated, and the list of updates to make to the treatment.

How To

1. Get an access token by submitting a GET request to the `/SASLogon/oauth/token` resource. The example described in Chapter 2, "Define Basic Methods and Get an Access Token" includes the code required to get an access token. Prepend that code to the code for this example.
2. Specify the tracking code of the subject contact record and treatment ID of the treatment that you want to update. These values are returned by the decision that determines the treatments for which the subject is eligible.
3. Define the record-level updates that you want to make to the subject contact record. You can update the following properties of the record:

   conclusionResponseValue
   : the subject's response to the results of the decision. You can use this property to record a response when a decision does not include treatments. When the decision includes treatments, this value might represent an overall response to all of the treatments in the subject contact record.

   conclusionResponseType
   : a summary value, category name, or other description for the response that is specified in the conclusionResponseValue property. For example, you could assign the type of **`Evaluating`**, **`No response`**, or **`Closed-Lost`** to the response. This value can be used as a target value in modeling.

4. Define the treatment-level updates that you want to make to the individual treatment. You can update the following properties of individual treatments:

   presented
   : specifies whether the treatment has been presented to the subject. Specify **`True`** or **`False`**.

   presentedTimeStamp
   : timestamp that indicates when the treatment was presented to the subject. Enter the timestamp in the following format:

     `yyyy-mm-ddThh:mm:ss.sssZ`

     Include the Coordinated Universal Time designator `Z` (for example, **`2019-11-14T18:47:40.719Z`**).

     For more information about timestamps, see Date and Time Formats.

   responseValue
   : the subject's response to the treatment. For example, this value could be a value or code such as **`Purchased`**, **`Rejected`**, **`Requested quote`**, **`Email opened`**, **`rt1`**, or **`em`**.

   responseChannel
   : the channel through which the user responded to the treatment. The response channel might be different from the contact channel. The possible values for the channel are defined in the Treatment Channels lookup table. For more information, see "Predefined Lookup Tables" in *SAS Intelligent Decisioning: User's Guide*.

responseType
: a summary value, category name, or other description for the response that is specified in the value of the responseValue property. For example, you could assign the category of **`Positive`**, **`Negative`**, or **`Neutral`** to the subject's response. This value can be used as a target value in modeling.

respondedTimeStamp
: a timestamp that indicates when the subject responded to the treatment. Enter the timestamp in the following format:

  `yyyy-mm-ddThh:mm:ss.sssZ`

  Include the Coordinated Universal Time designator `Z` (for example, **`2019-11-14T18:47:40.719Z`**).

  For more information about timestamps, see Date and Time Formats.

5 Use the tracking code to define the URL for retrieving the subject contact record, and define the accept type for the request.

6 Get the subject contact record by submitting a GET request to `/subjectContacts/contacts?filter=eq(responseTrackingCode,'trackingCode')`.

7 Convert the response body to a JSON object, and extract the object ID of the subject contact record.

8 Use the object ID to define the URL for retrieving the subject contact record, and define the accept type for the request.

9 Get the subject contact record by submitting a GET request to `/subjectContacts/contacts/object-ID`.

10 Retrieve the ETag value from the response header. For information about ETag values, see "Updating Objects with the put() Function" on page 5.

11 Convert the response body to a JSON object, and use the respBodyObjJSON object's update method to add the record-level updates.

12 Create an updated treatment list. For each treatment listed in the subject contact record that was returned in Step 9, complete these steps:

   a Test whether the treatment ID matches the ID of the treatment to update, and if so, append the treatment-level updates to the treatment.

   b Append the treatment to the updated treatment list.

13 Create a new Python dictionary with the key `treatmentsForConsideration` and the value of the updated treatments list.

14 Use the respBodyObjJSON object's update method to replace the `treatmentsForConsideration` property in the subject contact record with the updated treatments list. The respBodyObjJSON object now contains a complete, updated subject contact history record.

15 Convert the respBodyObjJSON object to a string so that you can specify it as the request body for the PUT request in Step 17.

16 Define the content type for submitting PUT requests to update subject contact records.

17 Update the subject contact record by submitting a PUT request to `/subjectContacts/contacts/`*`object-id`*. Specify `If-Match` as the conditional key and the ETag value from Step 10 as the value of the key.

Example Code

```
<< Include the code in "Define Basic Methods and Get an Access Token".
>>
<< That code is required to successfully execute this example.
>>

# Specify the tracking code for the subject contact record,
# and the object ID of the treatment that you want to update.
# Both values are GUIDs such as
# "b48ac290-f1a5-7343-8d85-e6f9fc85ff23".
trackingCode = "tracking-code"
treatmentToUpdate = "treatment_ID"

# Specify record-level updates. Modify these values
# values for your application.
recordUpdates = {"conclusionResponseValue" : "accepted",
                 "conclusionResponseType" : "crt_x"}

# Specify the updates that you want to make to the
# the treatment in the subject contact record.
# Modify these values for your application.

treatmentUpdates = {"presented" : "True",
                    "presentedTimeStamp" : "2019-11-14T18:47:40.719Z",
                    "responseValue" : "accepted", "responseChannel" :
"Web",
                    "respondedTimeStamp" : "2019-11-15T21:09:32.059Z"}

# Get the subject contact record for the tracking code.
requestUrl = baseUrl1 + \
    "/subjectContacts/contacts?filter=eq(responseTrackingCode,'" + \
    trackingCode + "')"
acceptType = "application/vnd.sas.collection+json"
respBodyTC,respHeaders = get(requestUrl, accessToken1, acceptType)
print ("response body = ", "\n", json.dumps(json.loads(respBodyTC),
       indent=4), end='\n\n')

# Convert the response to a JSON object, and
# extract the object ID.
respStrTC = json.loads(respBodyTC)
objID = respStrTC['items'][0]['id']

# Get the subject contact record using the object ID.
# The header returned by this request includes the
# ETag value that you need to update the record.
acceptType = "application/vnd.sas.decision.subject.contact+json"
```

```
requestUrl = baseUrl1 + "/subjectContacts/contacts/" + objID
respBodyOBJ,respHeaders = get(requestUrl, accessToken1, acceptType)

# Convert the response to a JSON object, and
# extract the ETag value.
respBodyObjJSON = json.loads(respBodyOBJ)
ETag = respHeaders['ETag']

# The variable respBodyObjJSON will contain all of the updates
# that need to be added to the subject contact record.
# Add the conclusionRepsonseValue and conclusionResponseType
# to the updates.
respBodyObjJSON.update(recordUpdates)

# For each treatment in the record, add the treatment data to
# the object respBodyObjJSON. For the treatment that is being
# updated, append the updates.
treatmentList=[]
for item in respBodyObjJSON['treatmentsForConsideration']:
    if item['id'] == treatmentToUpdate:
        item.update(treatmentUpdates)
    treatmentList.append(item)
contactUpdate = {"treatmentsForConsideration" : treatmentList}
respBodyObjJSON.update(contactUpdate)

# Convert the updated object to a string.
requestBody = json.dumps(respBodyObjJSON)

#Update the subject contact history record.
contentType = "application/vnd.sas.decision.subject.contact+json"
putResponse = put(requestUrl, contentType, acceptType, \
                  accessToken1, "If-Match", ETag, requestBody)
print(putResponse, end='\n\n')
print ("putResponse = ","\n",
json.dumps(json.loads(putResponse.content),
       indent=4), end='\n\n')
```

Responses

The initial response is the original, unmodified content of the subject contact history record.

```
response body =
 {
    "links": [
        <<lines deleted>>
        ],
    "name": "contacts",
    "accept": "application/vnd.sas.decision.subject.contact",
    "start": 0,
    "count": 1,
    "items": [
```

```
        {
            "creationTimeStamp": "2019-11-07T22:22:49.331Z",
            "modifiedTimeStamp": "2019-11-08T04:52:19.538Z",
            "createdBy": "anonymousUser",
            "modifiedBy": "userID",
            "id": "8177a38d-e834-4f6e-809b-3f59a9067409",
            "subjectId": "65420",
            "subjectLevel": "Customer",
            "objectUri": "/decisions/flows/2dd9fb9e-3f28-49ad-924e-5a340153a2d2",
            "objectRevisionId": "6b050ae1-52eb-49f7-a305-bc4df813d74d",
            "objectType": "decision",
            "treatmentsForConsideration": [
                {
                    "id": "1439c499-8cb3-453d-8795-9515cbbeda92",
                    "treatmentId": "41f29518-8709-4c42-bc5e-3f33dd48d297",
                    "treatmentRevisionId": "1af8e168-0c0e-44a1-8588-a1f3c75edcef",
                    "treatmentGroupId": "8c7dbfb5-2fc6-4820-95ea-a1cbc5850900",
                    "treatmentGroupRevisionId": "4842e53e-3295-445e-8b76-7b6140e6aff0",
                    "objectNodeId": "2d3e623f-02d4-4af7-a2e0-6d22a6dfb409",
                    "presented": false,
                    "subjectContactId": "8177a38d-e834-4f6e-809b-3f59a9067409"
                },
                {
                    "id": "f35a3742-15d6-4c04-b380-04d42024520b",
                    "treatmentId": "e07fc719-42be-41ae-a001-e24552010b51",
                    "treatmentRevisionId": "ec01fcc8-3624-496e-8064-bb61deff11f5",
                    "treatmentGroupId": "8c7dbfb5-2fc6-4820-95ea-a1cbc5850900",
                    "treatmentGroupRevisionId": "4842e53e-3295-445e-8b76-7b6140e6aff0",
                    "objectNodeId": "2d3e623f-02d4-4af7-a2e0-6d22a6dfb409",
                    "presented": false,
                    "subjectContactId": "8177a38d-e834-4f6e-809b-3f59a9067409"
                },
                {
                    "id": "ef9361f7-02c1-4d89-a3d0-045a1e6fbb59",
                    "treatmentId": "99b0749a-902c-4bcd-9685-f2bac6f71985",
                    "treatmentRevisionId": "7a57dae1-8784-4717-94f7-e965b7a75adc",
                    "treatmentGroupId": "8c7dbfb5-2fc6-4820-95ea-a1cbc5850900",
                    "treatmentGroupRevisionId": "4842e53e-3295-445e-8b76-7b6140e6aff0",
                    "objectNodeId": "2d3e623f-02d4-4af7-a2e0-6d22a6dfb409",
                    "presented": false,
                    "subjectContactId": "8177a38d-e834-4f6e-809b-3f59a9067409"
                }
            ],
            "ruleFired": "",
            "pathTraversed": "/b7e7bf1c-b363-468b-a849-1beb0f2a406c/e35e9783-01d2-...",
            "responseTrackingCode": "b48ac290-f1a5-7343-8d85-e6f9fc85ff23",
            "channel": "NONE",
            "excludeFromContactRule": false,
            "version": 1,
            "links": [
                <<lines deleted>>
                 ]
        }
    ],
    "limit": 10,
    "version": 2
```

```
}
```

If the PUT request is successful, the request updates the record and returns response code 200.

```
<Response [200]>

putResponse =
 {
    "creationTimeStamp": "2019-11-07T22:22:49.331Z",
    "modifiedTimeStamp": "2019-11-08T04:57:42.735Z",
    "createdBy": "anonymousUser",
    "modifiedBy": "userID",
    "id": "8177a38d-e834-4f6e-809b-3f59a9067409",
    "subjectId": "65420",
    "subjectLevel": "Customer",
    "objectUri": "/decisions/flows/2dd9fb9e-3f28-49ad-924e-5a340153a2d2",
    "objectRevisionId": "6b050ae1-52eb-49f7-a305-bc4df813d74d",
    "objectType": "decision",
    "treatmentsForConsideration": [
        {
            "id": "1439c499-8cb3-453d-8795-9515cbbeda92",
            "treatmentId": "41f29518-8709-4c42-bc5e-3f33dd48d297",
            "treatmentRevisionId": "1af8e168-0c0e-44a1-8588-a1f3c75edcef",
            "treatmentGroupId": "8c7dbfb5-2fc6-4820-95ea-a1cbc5850900",
            "treatmentGroupRevisionId": "4842e53e-3295-445e-8b76-7b6140e6aff0",
            "objectNodeId": "2d3e623f-02d4-4af7-a2e0-6d22a6dfb409",
            "presented": true,
            "presentedTimeStamp": "2019-11-14T18:47:40.719Z",
            "respondedTimeStamp": "2019-11-15T21:09:32.059Z",
            "responseValue": "accepted",
            "responseChannel": "Web",
            "subjectContactId": "8177a38d-e834-4f6e-809b-3f59a9067409"
        },
        {
            "id": "f35a3742-15d6-4c04-b380-04d42024520b",
            "treatmentId": "e07fc719-42be-41ae-a001-e24552010b51",
            "treatmentRevisionId": "ec01fcc8-3624-496e-8064-bb61deff11f5",
            "treatmentGroupId": "8c7dbfb5-2fc6-4820-95ea-a1cbc5850900",
            "treatmentGroupRevisionId": "4842e53e-3295-445e-8b76-7b6140e6aff0",
            "objectNodeId": "2d3e623f-02d4-4af7-a2e0-6d22a6dfb409",
            "presented": false,
            "subjectContactId": "8177a38d-e834-4f6e-809b-3f59a9067409"
        },
        {
            "id": "ef9361f7-02c1-4d89-a3d0-045a1e6fbb59",
            "treatmentId": "99b0749a-902c-4bcd-9685-f2bac6f71985",
            "treatmentRevisionId": "7a57dae1-8784-4717-94f7-e965b7a75adc",
            "treatmentGroupId": "8c7dbfb5-2fc6-4820-95ea-a1cbc5850900",
            "treatmentGroupRevisionId": "4842e53e-3295-445e-8b76-7b6140e6aff0",
            "objectNodeId": "2d3e623f-02d4-4af7-a2e0-6d22a6dfb409",
            "presented": false,
            "subjectContactId": "8177a38d-e834-4f6e-809b-3f59a9067409"
        }
    ],
    "ruleFired": "",
    "pathTraversed": "/b7e7bf1c-b363-468b-a849-1beb0f2a406c/e35e9783-01d2-...",
```

```
    "responseTrackingCode": "b48ac290-f1a5-7343-8d85-e6f9fc85ff23",
    "channel": "NONE",
    "conclusionResponseValue": "accepted",
    "conclusionResponseType": "crt_x",
    "excludeFromContactRule": false,
    "version": 1,
    "links": [
        <<lines deleted>>
    ]
}
```


See Also

Update a contact record

Update a Contact Record to Record Presentation