Paper SAS2145-2018

Integrating SAS® Analytics into Your Web Page

James Kochuba and David Hare, SAS Institute Inc.

ABSTRACT

SAS® Viya® adds enhancements to the SAS® Platform that include the ability to access SAS® services from other applications. Whether your application is in Python, Java, Lua, or R, you can now access SAS analytics and integrate them directly in your application. You can even use REST APIs. In this session, we look at using the REST APIs in SAS Viya to execute SAS® Cloud Analytic Services (CAS) actions and embed them in an application, which in this case is a web page. Examples include uploading a table, performing a SAS analytic procedure and displaying the output, and publishing a report. This method provides you with much greater flexibility and customization for building dashboards and reporting sites. Through this session, you will gain an understanding of how you can embed analytics from SAS Viya into your very own application.

INTRODUCTION

Companies have many pre-existing, web present forms that need enhancements like adding real-time or interactive charts and analytics for adaptive response. The web present form can be a simple web page or custom built web tool or even a chatbot. Adding these enhancements to the web present form can be extremely costly, in technical expertise and time, or potentially not possible based on the technology used to create them. Integrating SAS capabilities, like analytics, scoring, or charting, into the web present form would address the enhancements commonly needed and wanted in the market. The integration between the various technologies is done using APIs. Below are various examples of integration scenarios where SAS APIs easily added the enhancements to the web present form.

WEB APPLICATION SHARING REPORTS

In SAS® Viya®, there is a single interface for exploring and visualizing data with SAS® Visual Analytics, SAS® Visual Statistics, and SAS® Visual Data Mining and Machine Learning. This interface allows the user to work with a data set, and perform visualizations ranging from forecasting, clustering, and regressions to the most advanced machine learning algorithms such as neural networks and gradient boosting. The results of the user's work, including all visualizations, can be saved in a report, which can be shared with other users and visualized in the SAS® Report Viewer. This is a common practice used by many SAS Viya customers today.

However, each individual report object (decision tree, geospatial plot, correlation matrix, and so on) can also be shared to access other applications besides the SAS Report Viewer. There are four main options for sharing report objects:

- Email a link to the report to a coworker or colleague.
- Share a direct HTTP link to the object. This link can be added to another web page and then redirect users to the SAS Report Viewer to view the object.
- Share a static image of the object. This is not an interactive view, however, so an example of this use case might be for printed materials.
- Share the object as embeddable HTML code, allowing users to interact with that object wherever that code is integrated.

The last option (embeddable interactive) is the most interesting, and it is what we will focus on in this paper. It is important to note, that in all of these scenarios, the users with whom the report is being shared must have Read permissions to access the report, and Read permissions on the data the report is based on. You can also enable access for anyone to view the report objects. We'll cover this in more detail in the **Guest Access** section.

PROCESS TO SHARE A REPORT

The steps to share a report with other users are well documented and can be found in the Resources section. At a high level, in the SAS Report Viewer with a report open, hold your pointer over an object in the report and click the "overflow" icon, which some call the snowman icon. ⋮

From the **Share Object** menu, select the **Embeddable link** check box, and then customize using the available link options.

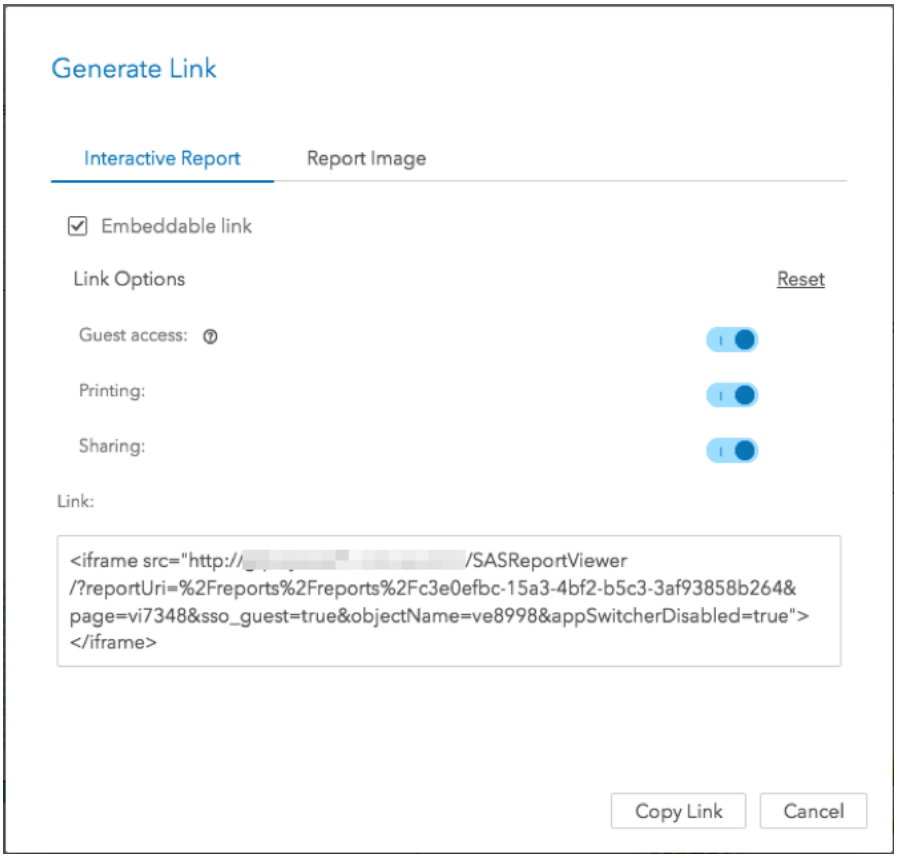


Image 1. Generate a Link to an Interactive Report

Note, the Guest access link option is displayed only if Guest Access has been enabled in your environment. For more information, see the **Guest Access** section.

The generated link is produced as HTML iframe code and can be embedded into a web page. This web page can be hosted on the SAS Viya server as well, with no other steps needed to integrate the report into the web page. If the web page is hosted on a different server, SAS Viya requires some extra configuration steps. These steps are covered in the **Embedding Externally** section.

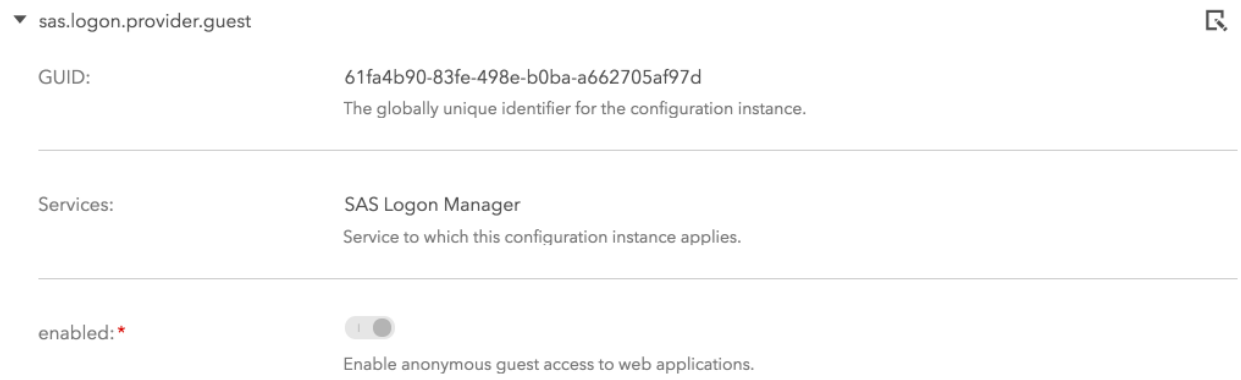
GUEST ACCESS

Guest Access is a new feature in SAS Visual Analytics 8.2 on SAS Viya that allows users to access content without having to authenticate. This is commonly used for dashboards and reports that are shared with executive teams or for embedding report content into other applications. By default, Guest Access is disabled. There are three main steps to enabling Guest Access:

1. Enable Guest Access on the environment. This essentially means turning the Guest Access feature on, but not enabling it across the environment.
2. Enable Guest Access on a specific report.
3. Enable Guest Access on the data in the report.

The process for completing these steps is fully documented in the Resources at the end of this paper, but we'll summarize the steps below.

1. In SAS® Environment Manager, create a new configuration instance for **sas.logon.provider.guest** and enable the Guest Access option.

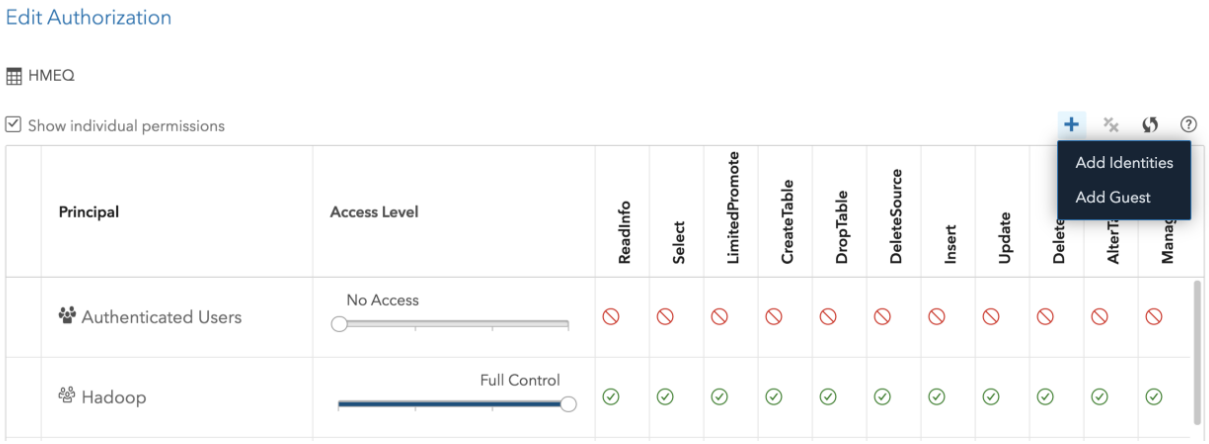


2. Modify the Authorization rules to allow Guest Access. This can be done via the command line using the sas-admin scripting utility. If you have not used this tool before, you must first authenticate to create a token:

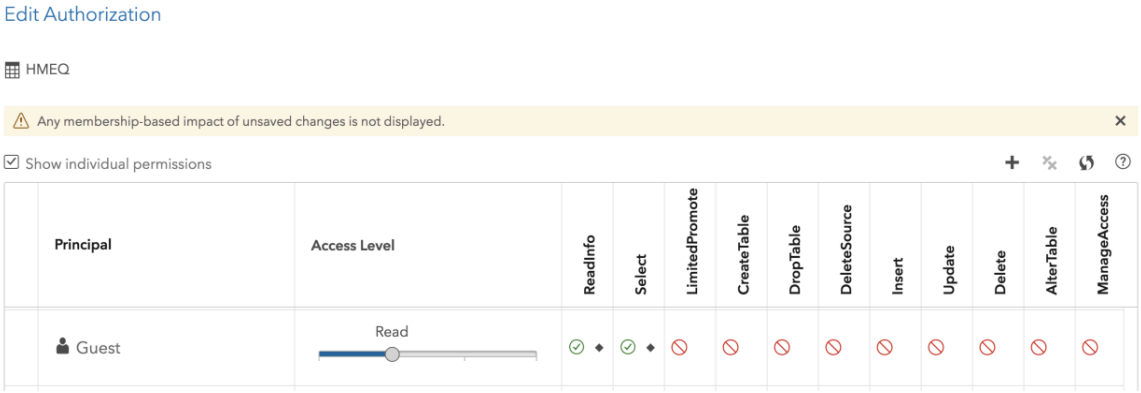
```
[root@viya123 ~]# sas-admin auth login
Enter credentials for http://viya123.unx.sas.com:

Userid> joeuser

Password>
Login succeeded. Token saved.

[root@viya123 ~]# sas-admin authorization facilitate-guest
Guest access is enabled.
```

3. At this point, configuring Guest Access on the data backing the report can be performed via the same sas-admin command line tool, or in SAS Environment Manager.

4. In SAS Environment Manager, simply find the table in question, and under the Edit Authorization options, select Add Guest.



That will add Guest with no permissions. You then need to slide the access level to **Read** so that guest users can read the data from that table.



Now you can enable Guest Access on the report that you want to share. Again, we can do this in SAS Environment Manager, following the steps in the **Process to Share a Report** section. Be sure to enable the Guest Access slider under Link Options:

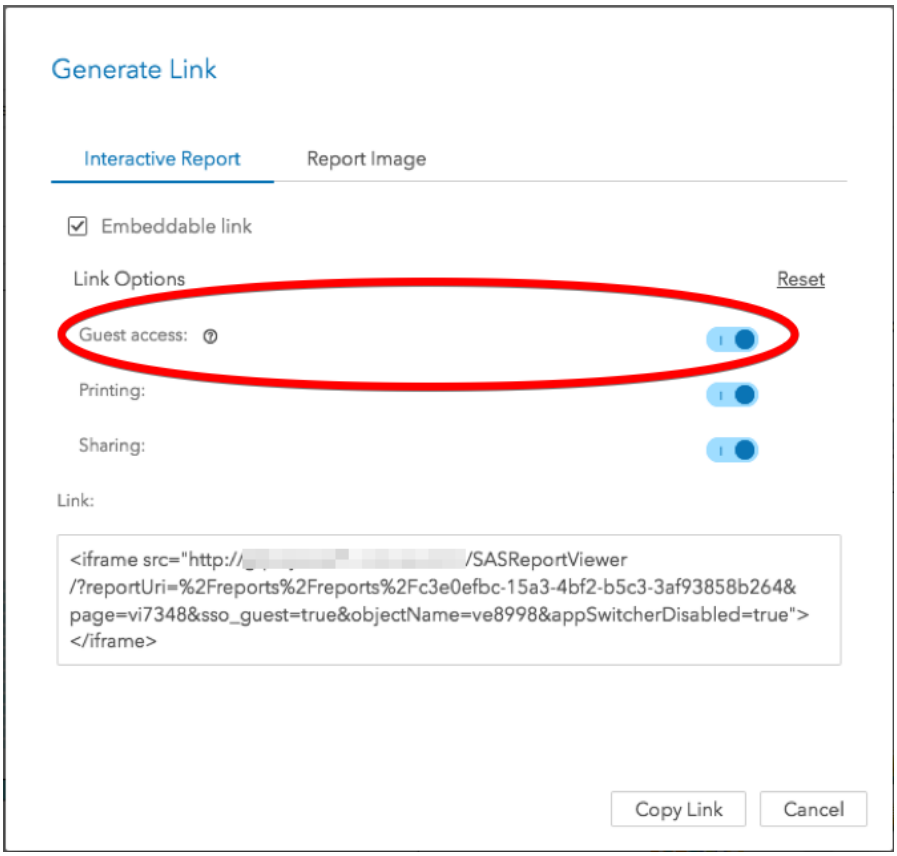


Image 2. Enabling Guest Access on a Report link

At this point, an anonymous user should be able to access the report in question directly, without being prompted to authenticate.

Note, if the report is using any views with maps, the data on the maps will not be displayed. This is because the maps use an additional data source, as evidenced by the Maps server log:

```
2017-12-05 10:03:42.768  WARN 19233 --- [o-auto-1-exec-3]
com.sas.maps.MapsService                 : guest(f1569adf)
[b727d8a84002797e] Error running query for map regions -  - Cannot
find the requested data source.
```

To resolve this problem, Guest Access must be granted on the AppData CAS Library. Currently, this cannot be configured in SAS Environment Manager. However, we can use the CAS Server Monitor tool to edit the authorization for the AppData CAS Library.

Under Global Caslibs, find the AppData library, click the overflow icon, and select Edit Access Controls.

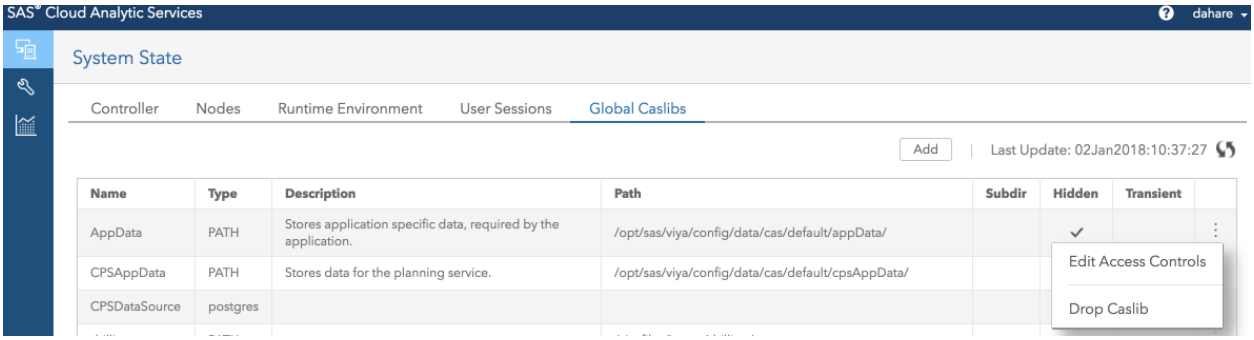


Image 3. Edit Access Controls for the AppData library

Then add two new rows for Guest to have Read Info, and select **Activity**.

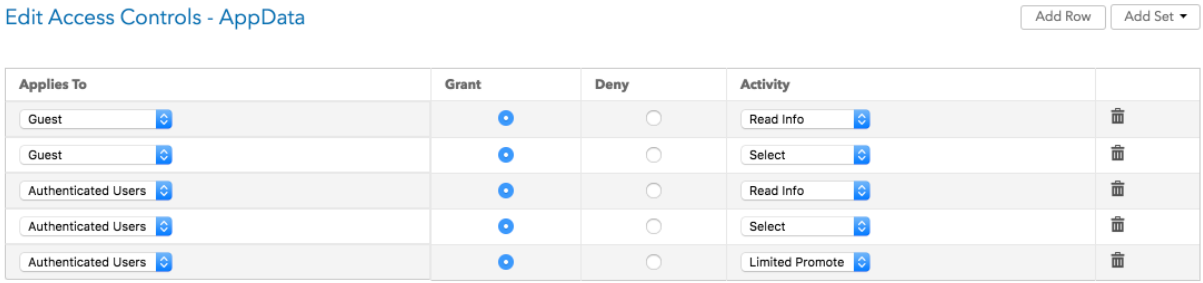


Image 4. Enable Guest Access on the AppData library

After completion, any reports with map views should be rendered successfully to guest users.

EMBEDDING EXTERNALLY

The information above will help you embed reports into web pages, as long as the web page is hosted on the same machine (same host name) as the SAS Report Viewer is running. If you try to embed the report into a web page running on an HTTP server on a different host, the attempt will fail, and nothing will be displayed. When you go to a debugger like Firebug, or in **Chrome -> Developer Tools -> Console,** the following error will be displayed:

Refused to display 'http://viya123.unx.sas.com/SASReportViewer/?reportUri=%2Freports%2Freports%2Fc3e0efbc-15a3-4bf2-b5c3-3af93858b264&page=vi7348&objectName=ve9828&appSwitcherDisabled=true' in a frame because it set 'X-Frame-Options' to 'sameorigin'.

This is because SAS has prevented embedding SAS Viya content into other sources for security protection. However, this can be controlled if you know and trust the host in which the content will be embedded.

In SAS Environment Manager, under **Configuration**, select the **Definitions** menu, and select **sas.commons.web.security**. If a definition hasn't already been created, you must create a new one, and configure it for the SAS Report Viewer only. Find and disable the **x-frame-options-enabled** setting:

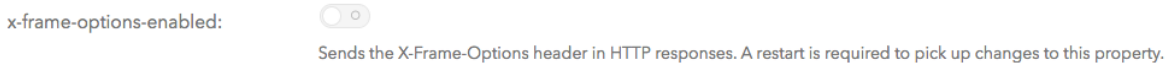


Image 5. Disabling the x-frame-options-enabled setting

This will allow any SAS Report Viewer content to be embedded into any host.
If you want to restrict the host list from any host to specific hosts, leave the x-frame-options-enabled setting on, but specify a host list that is allowed to access the content. Here is an example:

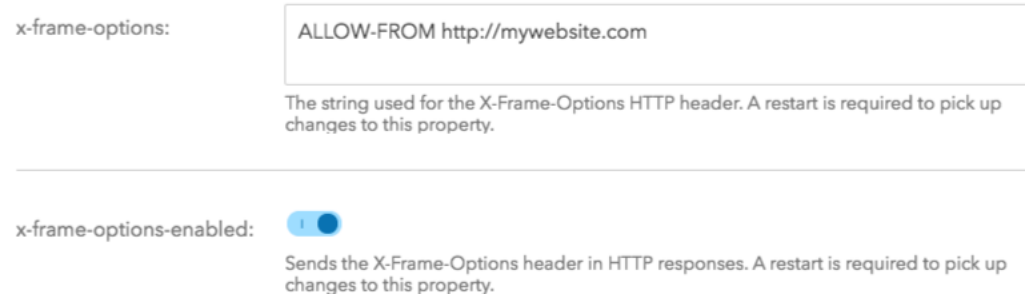


Image 6. Allow embedding from a specific host

In this scenario, SAS Report Viewer content could be embedded only to HTML running on the host that is specified. However, not all browsers (Chrome, for example) support the ALLOW-FROM x-frame-option.

Whichever configuration you choose, you must restart the SAS Report Viewer service for the change to take effect.

```
[root@viya123 ~]# service sas-viya-sasreportviewer-default restart
sas-viya-sasreportviewer-default is stopped
sas-viya-sasreportviewer-default is running
```

After you complete these steps, report content will be loaded in an external HTML page without prompting for any login credentials.

WEB APPLICATION SCORING

In addition to looking at report information, you might want your application to make a call to score new data against a model that you've built in SAS Viya. This can be easily accomplished with the features in SAS® Model Studio, a tool that is included with SAS Visual Data Mining and Machine Learning, SAS® Visual Forecasting, or SAS® Visual Text Analytics.

In SAS Model Studio, after building your pipeline and determining your model champion in the Pipeline Comparison view, simply select the overflow icon and you can download score code for that model.

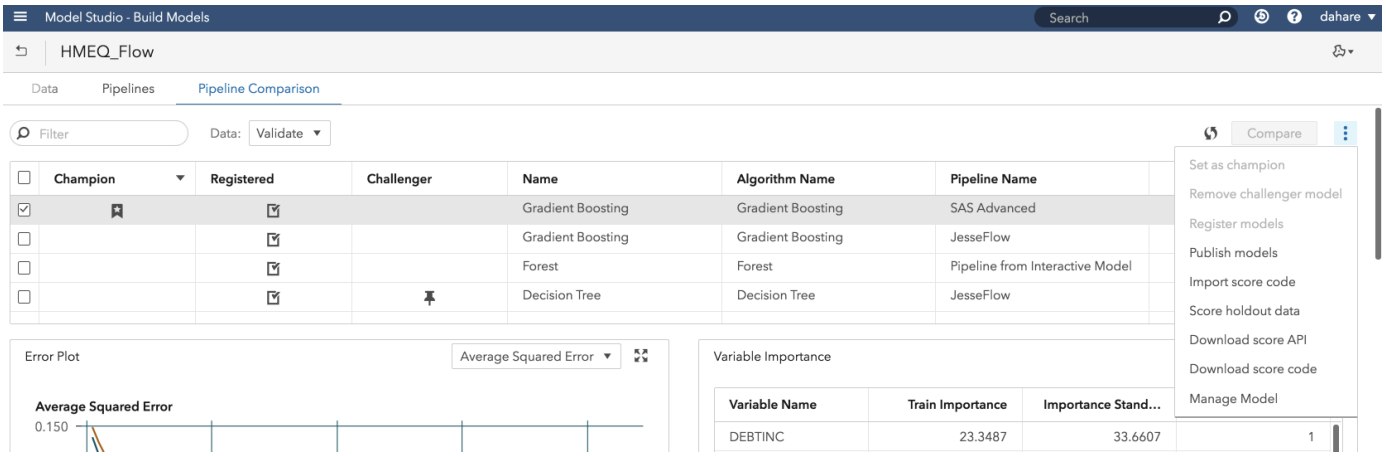


Image 7. Downloading score code for a model

In the Download score code option, we can choose between score code in the SAS language, Python, or REST.

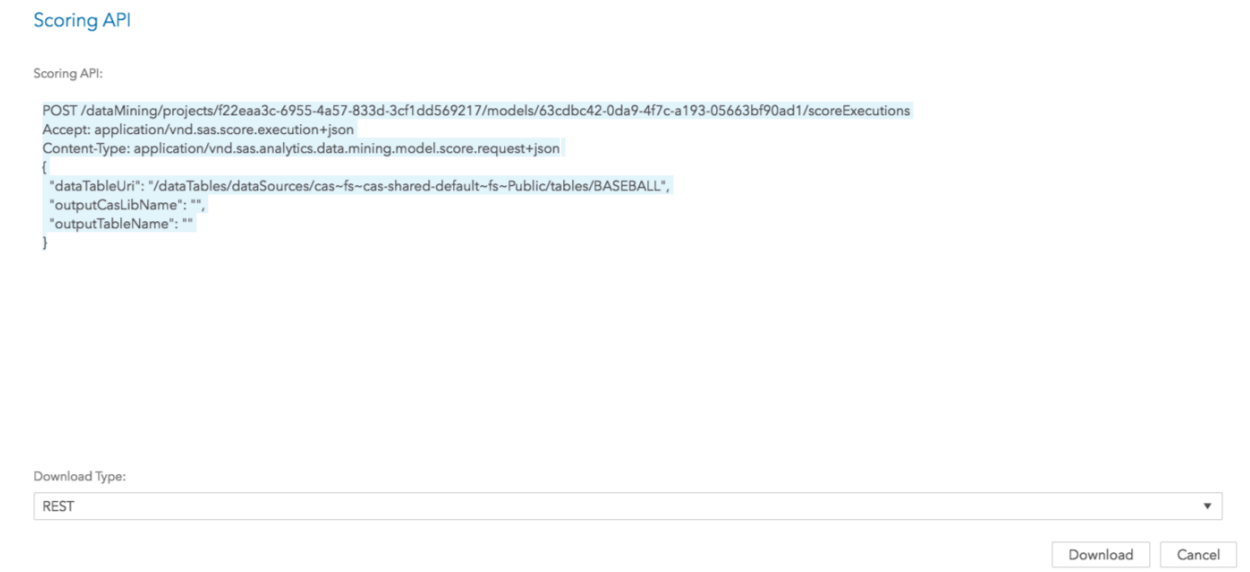


Image 8. REST Score code example

This score code can be integrated into your web-based application with the REST option. This will allow you to call the scoring action on your champion model with your new data.

WEB APPLICATION CHATBOT INTEGRATION

Chatbots are programs that conduct a conversation to simulate how a human would engage with someone. Chatbot programs are growing in usage, and that is producing many different options to implement in your company. You can build from scratch or choose a third-party tool to help generate your chatbot. A chatbot can be simple when it echoes the user input, but not very useful. Creating a chatbot to deal with specific common questions can require complex programming with business logic, but still be static and require rewrites to add more questions. Creating a chatbot that can adapt brings up words like artificial intelligence (AI) and sounds nearly impossible.

But, by integrating analytics into the chatbot, you can make a chatbot become more adaptive in addressing its AI capabilities. With the many forms of chatbot, adding analytics can be very restrictive (limited list of capabilities) or hard to implement. SAS can provide simple APIs to integrate into the various different types of chatbot implementations. The most common integration is using REST API calls to SAS Viya.

Here is an example when using Motion AI to create a chatbot that uses the power of SAS Viya. The bot's editor page is where you can add various modules to design your chatbot. We added a generic “Node.js” module, allowing us to customize the integration in the chatbot flow. Here is a picture of a bot editor where we added an Existing Sentiment module to check the user’s experience during the conversation:

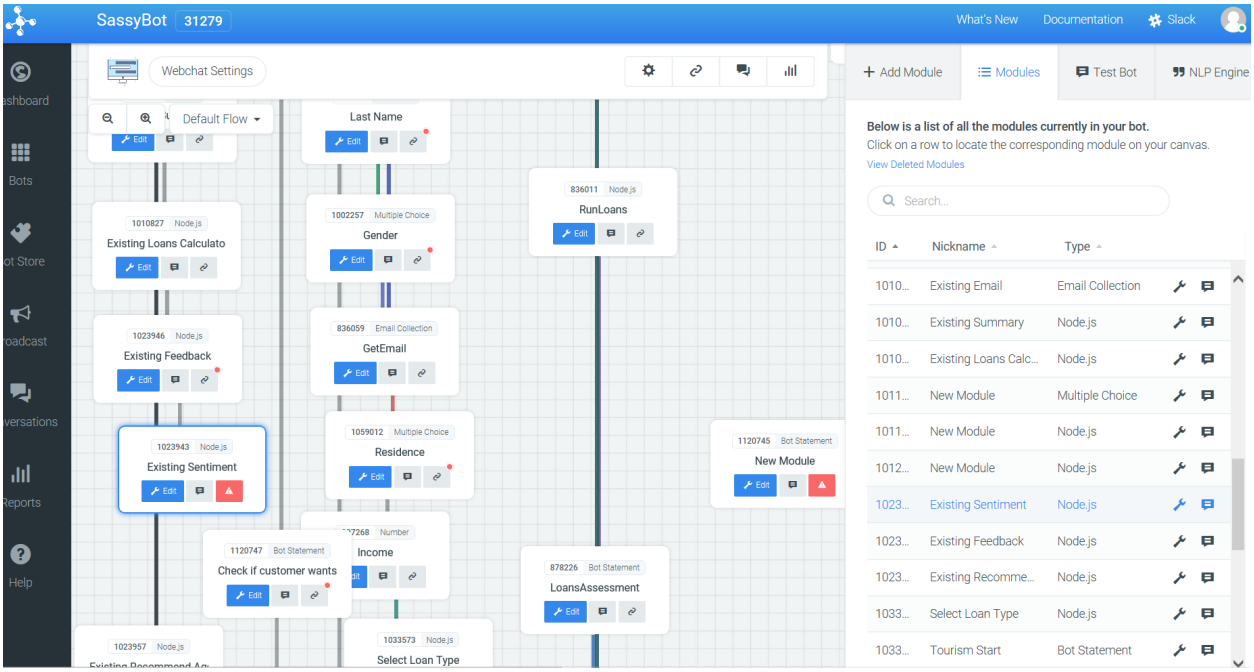


Image 9. Configuring a chatbot example

In the Node.js module, you make a REST call to SAS Viya and review the response to determine the chatbot action.

A tip with a chatbot: Since most times chatbots deal with anonymous users, we did not want to expose SAS Viya directly to anonymous users. Therefore, we created a Python REST interface for Node.js to call, thus allowing for a generic login. Then a Python server would authenticate to SAS Viya. This would keep the chatbot code simpler (see below) and allow for us to add better input validation and avoid potential attacks.

Example Motion AI Node.js Module Code

```
server2 = 'http://viyapython.sas.com/run?model=SentimentCheck'
    request.post(server2, function (error, response, body) {
  var body = JSON.parse(body)
  response = body.sentiment
  if (response == 'Positive') {
    var sentence = 'Thank you, I am flattered by your positive response! '
    var nextmod = 1120747
}
```

```
if (response == 'Neutral') {
    var sentence = 'Thank you for your time and feedback! '
    var nextmod = 1120747
}

if (response == 'Negative') {
    var sentence = 'I am really sorry! I will try to do better next time!
'
    var nextmod = 1023957
}
```

CONCLUSION

In this paper, we have shown how SAS capabilities such as viewing reports and scoring new data can be integrated into other applications. These applications can range from web pages to chatbots, and many in between, thanks to the SAS Viya ability to allow third-party languages such as Python and Java to access the SAS® analytics engine. The various ways this can be achieved are countless, but we hope this paper has shown a few options to consider when planning your application's integration with SAS analytics.

RESOURCES

SAS VISUAL ANALYTICS, SAS VISUAL STATISTICS, SAS VISUAL DATA MINING AND MACHINE LEARNING – CUMULATIVE FUNCTIONALITY

http://go.documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.3&docsetId=viyaov&docsetTarget=n00000sasviya000architecture.htm&locale=en#n15iagnf1rd897n1id2f3oq997hj

SHARING REPORTS WITH OTHER USERS

http://go.documentation.sas.com/?docsetId=vareports&docsetTarget=n0objrswfw1dcmn1c210zb1diksc.htm&docsetVersion=8.2&locale=en

SHARING REPORTS OR OBJECTS

http://go.documentation.sas.com/?docsetId=vavwr&docsetTarget=p05l1xrrfv3s4en18yy6j4ajx6ut.htm&docsetVersion=8.2&locale=en

USING URL PARAMETERS TO VIEW A REPORT

http://go.documentation.sas.com/?docsetId=vavwr&docsetTarget=p0l4zt68r3id4wn1fk3y3kconfg4.htm&docsetVersion=8.2&locale=en

ENABLING GUEST ACCESS

http://go.documentation.sas.com/?docsetId=calauthmdl&docsetTarget=n067qoyrgu1yohn19nq4ehy8o0b3.htm&docsetVersion=3.3&locale=en

SAS ADMIN SCRIPTING UTILITY

http://go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.3&docsetId=calcli&docsetTarget=n01xwtcatlinzrn1gztsglukb34a.htm&locale=en

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

James Kochuba
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
James.Kochuba@sas.com
http://www.sas.com

David Hare
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
David.Hare@sas.com
http://www.sas.com