SAS® Job Execution Web Application 2.2: User’s Guide

What’s New

SAS Job Execution Web Application includes the following enhancements:

- Separate tabs are created when you perform different functions (editing, viewing, displaying forms, or executing jobs), which enables you to work on multiple jobs concurrently.
- HTML input forms, prompts, and source code can all be stored together within a single job definition. When you copy or move a job, its user interface moves with it.
- You can select **Show log** in the Job Submit Options dialog box to open a new tab that contains the log when the job has completed. You can also select a job on the Jobs page and click 📋 to display the SAS log for the job execution.
- Jobs can be scheduled within the application.
- The following samples have been added:
  - Dynamic Prompts Using CARS Data illustrates how to create SAS Viya dynamic and cascading prompts.
  - Return GCHART Image Output uses the GCHART procedure to create a bar chart and returns the image data to the client.
  - Return SGPLOT Image Output uses the SGPLOT procedure to create a bar chart and returns the image data to the client.
- You can use the SAS prompting interface to create prompts using XML. Prompts have been added to most of the samples. You can use _ACTION=prompts to display prompts for a job.

You can also access jobs that were created in SAS Studio and execute them in the SAS Job Execution Web Application.
Overview

A SAS Viya job consists of a program and its definition, which includes information such as the job name, author, and creation date and time. You can use jobs for web reporting, performing analytics, building web applications, and delivering content to clients. One such client is the SAS Job Execution Web Application. This client works with jobs that contain SAS code. These jobs can access any SAS data source or external file and create new tables, files, or other data targets that are supported by SAS.

By default, jobs that are submitted through the SAS Job Execution Web Application begin with the %JESBEGIN macro and end with the %JESEND macro. These macros produce HTML output by default.

An input form can be created from HTML or the SAS prompting interface to provide a user interface to the job. The job definitions and user interface definitions are assigned unique identifiers, stored in the SAS Infrastructure Data Server, and executed in real time by client applications.

The SAS Job Execution Web Application is a web-based client used to create, manage, and execute jobs. This application, written in Java, provides access to data in combination with a powerful array of analysis and presentation procedures running on a server. No SAS software is required on the client machine.

To access and analyze data, a web user typically completes an input form displayed by the SAS Job Execution Web Application. When the user selects the option to submit the information, data specified in the form is passed to a waiting SAS session as global macro variables. The SAS program runs and the results are returned to the web browser.

You do not need Java or script programming experience to use the SAS Job Execution Web Application. You can create the web user interface and retrieve SAS data for display on the web using only HTML or the SAS prompting interface and SAS code.

Use the SAS Job Execution Web Application if you

- want to analyze and display information dynamically on the web and let your web users immediately retrieve the information that they need.
- have SAS programming experience but little or no web programming experience. You can create the web user interface and retrieve the SAS data for display on the web.
- want to create applications that provide web output without investing a lot of programming time.
- want to create applications that run on a variety of web browsers.

The SAS Job Execution Web Application has several types of users, as follows:

- End users enter information in an input form, select a link, or view an inline image that is displayed in a web browser.
- Web page authors create the input forms or pages that collect and submit input to the SAS job. These individuals could be SAS application developers.
- SAS job program component developers create the SAS programs that receive information entered in the input form.

The SAS Job Execution Web Application executes a stored job when it receives a request with a parameter indicating the location of the job. The request is initiated by accessing a URL with the following general format:

http://host:port/SASJobExecution/?_program=/SomeFolder/Hello World
In this example, the Hello World job stored in the `/SomeFolder` location is executed.

Various SAS Viya services are used to retrieve the job, submit it for processing, and display the job output. The following figure shows the components that are used to run a job:

**Figure 1  Components of Running a Job**

The execution flow consists of the following steps:

1. A request is submitted to the SAS Job Execution Web Application, usually initiated by a web browser client.

2. If the job is referenced using a folder path, then the job location is obtained from the Folders service. This step is skipped if the job location was specified in the `_JOB` parameter.

3. The job definition is retrieved from the Job Definitions service by using the job location obtained in step 2.

4. Input parameters, execution parameters, and the job definition are submitted to the Job Execution service.

5. The request is passed to the Compute service, which uses the Launcher service to start a compute server.

6. The SAS program runs in the compute server and creates output in the SAS Infrastructure Data Server using the Files service. The files are associated with the job execution object to facilitate retrieval.

   Note: The compute server can access the SAS Cloud Analytic Services (CAS) server though the CAS procedure or through the CAS LIBNAME engine. For more information, see *SAS Cloud Analytic Services: CASL Reference* and *SAS Cloud Analytic Services: User’s Guide*, respectively.

7. The Job Execution service monitors the job execution and alerts the SAS Job Execution Web Application when the job has completed.
The SAS Job Execution Web Application instructs the web browser to use the Files service to retrieve the desired output files linked to the job execution object. If the _SAVEFOLDER option is used, then the output is saved to the specified folder.

The job execution object has an expiration time, which is set when the job has finished executing. When it expires, the job execution object and all files associated with it are deleted.

Working with the SAS Job Execution Web Application

Accessing the Application

Access the SAS Job Execution Web Application using this URL:

http://host:port/SASJobExecution

If you do not specify a value for port, the default value of 80 is used.

Accessing Content

Overview

The SAS Content page enables you to locate, manage, and execute job-related files. If the output of a job is saved to SAS content folders, it can also be displayed from this page. The SAS Content page appears by default when you enter the SAS Job Execution Web Application, but if you are on another page in the web application, you can click to access it.

The left pane of the SAS Content displays folders and their contents, and the right pane is used as a presentation and working area for the various job functions. Separate tabs are created in the right pane when you perform different functions (editing, viewing, displaying forms, or executing jobs). Tabs are reused based on function, member name, and type (source code, HTML form, or prompts). A new tab is created every time you edit any folder member, regardless of whether you already have an editing tab open.

The full version of the SAS Content page is available to users who have authorization to develop jobs. A more limited SAS Content page is available to users who are allowed only to execute existing jobs and view output. The display and capabilities are automatically adjusted according to the authorization of each user. For example, HTML input forms are displayed only for developers. By default, all users have authorization to develop jobs. See “Changing Access to Application Functions” on page 24 for more information about how to assign users access to developer or user views.

The following actions are common to both views when you select a job:

- Click to retrieve the output from the most recent execution of the job, if it is still available.
- Click to submit a job for execution.
- Click to open the following dialog box with execution options:
You can add parameters using the format `name1=value1&name2=value2`.

The **Show log** option adds the `_debug=log` parameter setting to the request. You can use this option to open a new tab that contains the log when the job has completed (if New window is not selected). The **Show log** option does not work if `_debug=trace` is active.

The **New window** option displays the job output (and input form, if one exists) in a new browser tab or window, depending on how your browser is configured. Otherwise, the job output is displayed in a new tab within the SAS Job Execution Web Application. These options remain in effect for job submissions during the current browser session.

The `_FOLDER URL` parameter can be used to alter the folder list for the SAS Content page. Specifying `_folder=/SomeFolder` opens the display at the indicated folder path. Here is an example:

http://host:port/SASJobExecution?_folder=/SomeFolder

See "Accessing Application Pages through a URL" on page 9 for more information about accessing pages through a URL.

**Content for Developers**

Users who have authorization to develop jobs can access the full version of the SAS Content page using the `/developer` path.
Developers can add, delete, and rename folders and jobs. You cannot restore jobs that have been deleted. The following actions are available only for developers when a job is selected:

Note: Depending on the width of the left pane, you might need to click to access some of these actions.

- Click to add or edit the source code, HTML input form, or prompts for a job.
- Click to view the source code, HTML input form, or prompts for a job.
- Click to display the input form or the prompts for a job.
- Click to make a copy of the job definition. In the Save As dialog box, specify the folder location and file name for the new job definition.
- Click to move the job definition to another folder. In the Choose a Location dialog box, specify the folder location for the job definition.
- Click to access properties and parameters for the job. Developers can edit or add parameters to jobs. See “Passing User Input to a Job Using a Job Definition Parameter” on page 32 for more information about setting parameters.

See the Creating a Simple Job that Uses DATA Step Code and Passing User Input to Your Job Using an HTML Input Form sections in “Development Concepts” on page 27 for more information about creating and editing files.

Content for Users

Users who are allowed only to execute existing jobs and view output can access a more limited SAS Content page using the /user path.
Users can select a job and click to view the properties and parameters for a job.

Managing Jobs

Click to access the Jobs page, where you can display and maintain jobs that have run previously and have not yet expired. Each row of the table lists a previously submitted job, with information about the location of the job, its status, run time, who submitted the job and when, and when the job expires.

The following actions are available when a job is selected:

- Click to specify the date and time at which to delete a job.
- Click to display the job output in a new browser tab or window.
- Click to display the SAS log for the job execution.
- Click to save all output from the job. In the Save As dialog box, specify the folder location and file name for the output. If multiple output files were created, then you are prompted for a location and file name for each file.
- Click to open the Schedule Job dialog box on the Scheduling page. You can rename the scheduled job, edit the description, and add a new trigger to schedule how often and when to run the job again. See "Scheduling a Job" on page 16 for more information about using the _ACTION parameter to schedule a job.

You can specify the _PROGRAM input parameter to display results for a single job. For example, the following URL displays results only for the Hello World job:
http://host:port/SASJobExecution/jobs?_program=/SomeFolder/Hello_World

The value of the _PROGRAM parameter is case-sensitive and must exactly match the location and name of the job.

Scheduling a Job

Click to access the Scheduling page, where you can display jobs that have been prepared for scheduling or are already scheduled. Each row of the table lists a previously submitted job, whether it has been scheduled to run again, and who prepared the job for scheduling and when.

The following actions are available when a job is selected:

- Click to open the Schedule Job dialog box. You can rename the scheduled job and edit the description, which is useful if you want to schedule different variations of the job. You can add a new trigger to schedule how often and when to run the job. See “Jobs: How To” in SAS Viya Administration: Jobs for more information about scheduling jobs and creating a time trigger.

- Click to submit a job for execution. Click to access the Jobs page, where you can view the job output.

- Click to view the properties, arguments (which include parameters), and scheduling information for a job. You can edit the job's arguments or source code, which affects only this instance of the scheduled job, not the original job definition.

Note: Arguments that are used by the Job Execution service are case sensitive in this context. However, when using these arguments as job definition parameters, they are case insensitive. Some arguments (such as _EXPIRATION) must be set in the original job definition rather than in the instance of the scheduled job.

Installing the Samples

A set of sample jobs is available for installation. These samples illustrate various SAS program coding techniques and how jobs are executed with the SAS Job Execution Web Application.

Click to access the Samples administration page, where you can view a table of available samples. This is a list of items available for installation, not those that have already been installed.
To install samples and use them, copy the sample job definitions supplied by SAS to a folder. Select the desired items and then click **Copy To**. In the folder selector dialog box, select a destination folder and then click **OK** to copy the items. If an item already exists in that location, an option to skip, replace the item, or quit copying is presented.

Copying sample job definitions to a folder is typically a one-time operation performed as a post-installation step.

Accessing Application Pages through a URL

The page that appears when you access the application is determined by the authorization that is enabled. The default page lists job files and provides access to editing and administration tools. You can use SAS Environment Manager to create authorization rules that determine which users have access to different pages. See “Changing Access to Application Functions” on page 24 for more information.

Several pages are available with the SAS Job Execution Web Application. The different pages are displayed by adding paths to the base SAS Job Execution Web Application URL:

<table>
<thead>
<tr>
<th>Path</th>
<th>Page Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/apiMeta</td>
<td>Returns the application version and build information.</td>
</tr>
<tr>
<td>/developer</td>
<td>Displays all files and administration tools. Depending on a user’s authorization, this might be the default SAS Content page, which is used to create and maintain job-related files.</td>
</tr>
<tr>
<td>/env</td>
<td>Displays system environment information such as properties and parameters (restricted to SAS Administrators).</td>
</tr>
<tr>
<td>/jobs</td>
<td>Enables you to administer previous job runs.</td>
</tr>
<tr>
<td>/logout</td>
<td>Logs out of the application.</td>
</tr>
<tr>
<td>/samples</td>
<td>Enables you to administer sample jobs.</td>
</tr>
<tr>
<td>/scheduling</td>
<td>Enables you to schedule jobs.</td>
</tr>
<tr>
<td>Path</td>
<td>Page Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>/theme</td>
<td>Returns the CSS for the current theme.</td>
</tr>
<tr>
<td>/user</td>
<td>Displays and executes job files. Depending on a user's authorization, this might be the default SAS Content page.</td>
</tr>
</tbody>
</table>

For example, use the following URL to display and execute job files:

http://host:port/SASJobExecution/user

 Executing Jobs

Specifying a Job to Run

Jobs are typically referenced by a multi-level path. Jobs can be created in any folder, and they are referenced using the _PROGRAM parameter:

_program=/SomeFolder/jobname

When the SAS Job Execution Web Application receives this parameter, the job location is obtained from the program path and the job definition is retrieved.

Alternatively, an exact location can be entered using the _JOB parameter. The Job Definitions URI, which contains the unique identifier for the job, is specified in the following form:

_job=/jobDefinitions/definitions/3b9f3a5e-deb1-4873-a90a-be6280e35deb

Using Job Input Parameters

Job input parameters pass data to your SAS program as a list of name/value pairs. The name/value pairs can be specified in the URL, in fields in an HTML input form, in prompt fields, in job definition parameters, or in the SAS Job Execution Web Application configuration pane. SAS global macro variables, which are created from these name/value pairs, are available for use in your program.

Note: When prompts are displayed, they are populated with any parameter values that are specified in the URL or job definition. These values take precedence over parameter values that are set in prompt XML.

For example, when a job is executed using the following URL:

http://host:port/SASJobExecution/form/?_program=/Folder1/myJob&p=123

The following macro variables are created:
Macro Variable Name | Macro Variable Value
---------------------|---------------------
_program             | /Folder1/myJob      
p                     | 123                 

To avoid security vulnerabilities, the compute server processes incoming variable values to handle "special" characters. Unsafe characters that are masked for input parameters are the ampersand (&), apostrophe ('), percent sign (%), quotation marks ("), and semicolon (;). See "Input Parameters" in SAS Viya Administration: Programming Run-Time Servers for more information about special characters that are masked. See SAS Functions and CALL Routines: Reference for more information about using the COMPSRV_OVAL and COMPSRV_UNQUOTE2 functions to unmask some or all of the unsafe characters in an input parameter.

Prompting for Input Parameters with Job Forms

You can create HTML input forms or prompt forms that are stored separately from the job definition. Storing the form within the job definition makes it simpler to keep track of the form and to move or copy job definitions and forms. However, creating forms that are stored separately enables you to share forms between multiple job definitions. To create a separate form, navigate to the folder where you want to store the form and click from the File type drop-down list.

If you do this, you can either use the _FORM job definition parameter to specify the location of the form, or the form and job definition can be in the same location and have the same name. The _FORM parameter can be specified in the URL as an input parameter or preset as a job definition parameter using the Properties function of the SAS Content page.

Note: You must have access to developer functionality to modify properties for a job.

Here is an example of specifying the HTML input form using an input parameter:

http://host:port/SASJobExecution/form/?_program=/Folder1/myJob&_form=/Folder2/myJobForm

The HTML input form named myJobForm stored in /Folder2 is displayed. Note that the HTML input form does not need to be in the same folder as the job (/Folder1) nor does it need to have the same name as the job (myJob).

If the _FORM parameter is not specified, then an HTML input form with the same name and location as the corresponding job is displayed, if it exists:

http://host:port/SASJobExecution/form/?_program=/Folder1/myJob

The HTML input form named myJob stored in /Folder1 is displayed. The sample jobs provided by SAS take advantage of this feature.

Alternatively, you can use the _ACTION input parameter in the URL instead of the /form path. The previous examples become:

http://host:port/SASJobExecution/?_action=form&_program=/Folder1/myJob&_form=/Folder2/myJobForm

and

http://host:port/SASJobExecution/?_action=form&_program=/Folder1/myJob
To display a prompt form with the same name and location as the corresponding job, you can use the _ACTION=prompts input parameter in the URL, or the /prompts path as follows:

http://host:port/SASJobExecution/prompts/?_program=/Folder1/myJob

**Note:** If the job definition contains the _ACTION=form parameter, then using the /prompts path turns the parameter into _ACTION=form,prompts, which defaults to displaying the HTML input form.

Use the Properties function of the SAS Content page to specify _FORM as a preset job parameter. This technique is useful when you have multiple jobs that need to use the same HTML input form.

**Note:** You must have access to developer functionality to modify properties for a job.

Suppose that you have two jobs that use the HTML input form name Shared Input Form stored in the /Folder2 folder. Specify /Folder2/Shared Input Form for the _FORM parameter of each job. The input form is then displayed when both jobs are run using /form or _action=form, without the need for specifying _FORM in the URL:

http://host:port/SASJobExecution/form/?_program=/Folder1/myJob1

and

http://host:port/SASJobExecution/?_action=form&_program=/Folder1/myJob2

See “Passing User Input to a Job Using a Job Form” on page 32 for an example of how to use an HTML input form to pass input to a job.

---

**Specifying Output Files**

When a job is run by the compute server, files stored in the SAS Infrastructure Data Server can be returned to the web browser for display. After the job has completed, the SAS Job Execution Web Application directs the web browser to retrieve and display the desired files.

A file matching the pattern _webout.* is returned by default. If more than one file matches this pattern, then they are all returned in alphabetical order with the case and file extension considered in the ordering. The first file is displayed in an IFRAME in the HTML page, and links are provided to display the remaining files. Click a link to display the content of the file in the IFRAME. If the IFRAME is unable to render the content (for example, an RTF file or a SAS data set), then the file is downloaded.

Use the _RESULTFILE parameter to indicate which files should be returned. The parameter accepts a comma–separated list and supports the "*" (multi-character) and "?" (single-character) wildcards. The following table provides some examples:

<table>
<thead>
<tr>
<th>_RESULTFILE Value</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>All files</td>
</tr>
<tr>
<td>*.csv</td>
<td>Only CSV files</td>
</tr>
<tr>
<td><em>.html,</em>.htm</td>
<td>HTML and HTM files, in that order</td>
</tr>
<tr>
<td><em>.htm</em>,<em>.pdf,</em>.csv</td>
<td>HTM, HTML, CSV, and PDF files, in that order</td>
</tr>
<tr>
<td>_RESULTFILE Value</td>
<td>Returns</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>state??..csv</td>
<td>CSV files beginning with state and ending with any two characters (for example, state01.csv)</td>
</tr>
<tr>
<td>retail.htm,retail.pdf,class.htm,class.pdf</td>
<td>Only the files that are listed in the order in which they are listed</td>
</tr>
</tbody>
</table>

Use \_action=json if you have an application that requires a list in JSON format of output files that are created by the job. Here is an example:

http://host:port/SASJobExecution/?_program=/Folder1/myJob\&_action=json

This returns the following JSON instead of the _webout.htm file:

```
|
|    |
|    |
|    |
|    |
```

Use the URI in the HREF property to retrieve the _webout.htm file.

The following example uses the _RESULTFILE and _OMITTEXTLOG parameters to return all files that are created as a result of running the job, including the SAS log:

http://host:port/SASJobExecution/?_program=/Folder1/myJob\&_action=json\&_resultfile=*&\&_omittextlog=false

The following JSON is returned:

```
|
|    |
|    |
|    |
|    |
```

Using Output Job Parameters

A status message can be displayed to the user at the end of the job execution. If the SAS macro variable _STATUS_MESSAGE contains a value, then the value is displayed in a dialog box. See
Modifying the Job Execution

Job parameters can be used in the URL to change how jobs are executed. Default values for many of these job parameters can be set as configuration properties. See “Setting Configuration Properties” on page 19 for more information about setting configuration properties. Job parameters specified in the URL become macro variables, and they are available for use within your SAS program.

The _ACTION job parameter performs many different functions, as described in the following sections. See “Prompting for Input Parameters with Job Forms” on page 11 for more information about how to use it to display HTML input forms and prompts. The nobanner value is reserved for future use.

Specifying SAS System Options

You can use special input parameters to set SAS system options for a job. These options are specified upon SAS invocation. The general format is as follows:

%opt-unique-name option-specification

Always begin with %opt- (case insensitive) followed by a name not used by another input parameter. These parameters can be specified in an HTML input form, as job definition parameters, or in the query string. No SAS macro variables are created for this type of input parameter.

The following table shows sample values for setting a single option using an HTML input form or job definition parameter:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>%opt-myopt</td>
<td>linesize=max</td>
</tr>
</tbody>
</table>

The following table shows sample values for setting more than one option using an HTML input form or job definition parameter:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>%opt-myopt1</td>
<td>linesize=max</td>
</tr>
<tr>
<td>%opt-myopt2</td>
<td>nobyline</td>
</tr>
</tbody>
</table>

When specifying the parameter as part of the query string, be sure to URL encode the percent sign ("%") in %OPT- and also the equal sign ("=") if it is present in your option value. For example, use this URL parameter to specify 200 for the LINESIZE option:

http://host:port/SASJobExecution/?_program=/Folder1/myJob&%25opt-myopt=linesize%3d200
Job Execution Time-out

By default, the SAS Job Execution Web Application waits 300 seconds for a job to complete. You can change the time-out value for a specific request by specifying the _TIMEOUT job parameter in the URL. For example, to specify a value of 480 seconds, use the following value:

_timeout=480

The default time-out value for all jobs can be set using the EXECUTETIMEOUT configuration property. See “Setting Configuration Properties” on page 19 for more information about setting configuration properties.

If you execute a job that is expected to take more than 300 seconds to complete and you want the browser to wait, you can use _action=wait, which has no default time-out. Also, jobs that run using the _action=background parameter do not have a time-out value.

If you execute a job that requires more than 300 seconds to complete, then you might not get any output or you might receive the following error message:

Proxy Error
The proxy server received an invalid response from an upstream server.
The proxy server could not handle the request POST/SASJobExecution/.
Reason: Error reading from remote server.

To prevent this error, perform the following steps to increase the time-out value for the Apache httpd server:

1. Modify the petrichor.conf file, which is located in /etc/httpd/conf.d/.
2. Change the time-out value. The default value is 300 seconds and is a system-wide value that affects all services.
3. Stop and restart the Apache httpd server using the following command:

   sudo service httpd restart

Job Output Expiration

The output files created by a job are available for 30 minutes by default. Use the _EXPIRATION job parameter to specify a different duration. The value is formatted according to the W3C XML duration data type. For example, the following value specifies that the job expires within 60 minutes after execution has completed:

_expiration=PT60M

Specifying zero (0) indicates no expiration, and the output files are not deleted. The default value can be set using the EXPIRATION configuration property.

Background Processing

Use the _ACTION job parameter to perform several operations. Specify the following value to execute a job in the background without waiting for completion:

_action=background

The following message is displayed after the job is submitted:

   Job /Folder/job-name submitted for background processing.
You can use the Jobs page or the SAS Content page to view the results once the job has finished running.

For all jobs that are submitted, the job ID is returned in the HTTP header X-SAS-JOBEXEC-ID. For jobs that run in the background, the job ID can be used as follows to poll the jobExecution service to identify the state of the job:

/jobExecution/jobs/<job-ID>/state

When the job has finished running, you can use the job ID as follows to view the results:

/SASJobExecution/?_jobexec=/jobExecution/jobs/<job-ID>

Jobs run in the background expire after 24 hours, unless a different duration is specified using the _EXPIRATION parameter. Use the Jobs page to set the expiration after a job has executed.

Perform the following steps to use the BACKGROUND configuration property to specify a custom HTML page to display instead of the default message:

1. Create an HTML file with your customized message and then save it to a folder. See “Passing User Input to a Job Using a Job Form” on page 32 for more information about creating an HTML file.

2. Follow the instructions in “Setting Configuration Properties” on page 19 to create the BACKGROUND property with the full path and name of the HTML file. For example, specify /Folder/My Custom Background Message as the value.

Wait Screen

You can use the _ACTION job parameter to display a wait screen with informational text while the job is executing:

_action=wait

The default value for the text is Please wait. Use the WAITTEXT configuration property to specify the text of the wait message.

Retrieving Previous Results

Output from a previous job execution can be displayed if it has not yet expired:

_action=lastjob

This avoids the time needed to execute the job in the compute server and can be used to provide semi-static reports. This feature can improve performance and reduce server load if multiple users request a job that produces exactly the same results.

If there is no previous output, then you receive an error message indicating that previous job output does not exist.

The _ACTION job parameter also accepts a comma-separated list of values. Add execute to force the job to execute if the previous job output does not exist:

_action=lastjob,execute

Scheduling a Job

When you execute a job, you can use the _ACTION parameter as follows to save a copy of the job definition and run–time parameters (arguments) in the Scheduling table in SAS Environment Manager:
You can use SAS Environment Manager to complete the steps needed to schedule the job. See “Jobs: How To” in SAS Viya Administration: Jobs for more information about scheduling jobs in SAS Environment Manager.

See “Scheduling a Job” on page 8 for more information about manually performing this action within SAS Job Execution Web Application.

Saving Job Output

Job output files can be saved for later viewing. Use the _SAVEFOLDER parameter as follows to specify the folder in which to save the output:

```_savefolder=/Folder1/Folder2```

By default, the file names that are specified in the NAME option of the Files service access method FILENAME statement are used. Existing files are replaced. See the sections about assigning a FILEREFS in “Development Concepts” on page 27 for more information about how to use the FILENAME statement.

The _SAVEFILE parameter specifies the file to save:

```_savefile=_webout.html```

Only the _webout.html file, specified in the NAME option of the Files service access method FILENAME statement, is saved. Omit this parameter to save all job output files or if the job creates only one file. Note that by default, only files named with the pattern _webout.* are returned. Use the _RESULTFILE parameter to specify return files with names that do not match this pattern.

Specify the parameter multiple times to save only some of the output files created by a job. Only the _webout.html and _webout.pdf files are saved from a job that creates multiple files:

```_savefile=_webout.html &_savefile=_webout.pdf```

and

```_resultfile=* &_savefile=MyOutput.html &_savefile=MyOutput.pdf```

The _SAVEFILE parameter can be used to save a file with a new name:

```_savefile=original-name,new-name```

The _webout.html file is saved as MyOutput.html:

```_savefile=_webout.html,MyOutput.html```

See “Managing Jobs” on page 7 for more information about manually saving job output.

After the job output is saved, it is available in SAS Drive. From there, it can be downloaded, displayed, and accessed by other applications.

Debugging

Debugging information can be returned to assist in program development and execution. Like the _ACTION job parameter, _DEBUG accepts a single value or a comma-separated list of values:

```_debug=fields,log,time,trace```

The following values are available:
<table>
<thead>
<tr>
<th>Value</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields</td>
<td>List of all input parameters. Those listed in the Arguments section become SAS global macro variables.</td>
</tr>
<tr>
<td>log</td>
<td>SAS job log.</td>
</tr>
<tr>
<td>time</td>
<td>Total time in seconds to execute the job; incremental and total time are reported when used in conjunction with trace.</td>
</tr>
<tr>
<td>trace</td>
<td>Information for each step of the job.</td>
</tr>
</tbody>
</table>

Use `trace` to display detailed information about the job execution, including preset and input parameters, the job definition and SAS code, and results created by the job.

By default, the `_DEBUG` parameter can be used by any authenticated user. To restrict its usage on a group or user basis, change the authorization rules for the `/SASJobExecution/debug/` key using SAS Environment Manager. See “Security for SAS Viya Jobs” on page 24 for more information.

Use the `DEBUGDISALLOW` configuration property to disallow the use of specific debug values for all users.

You can use SAS Environment Manager to view log messages generated by the SAS Job Execution Web Application. Click 📦 to display the Logs Filter and Messages panels. Select `jobexecapp` in the Logs Filter and then click Apply to display only the pertinent messages.

Perform the following steps to change the amount of information that is generated in the log:

1. Click 📦 in SAS Environment Manager, and then select Definitions from the View drop-down list.
2. Click logging.level to display the properties that are used to configure logging levels.
3. Specify `com.sas.jobexec` in the Filter field at the top of the right panel.
4. If no properties are found, then you must add one. Click New Configuration and then supply the required value in the New logging.level Configuration dialog box. Otherwise, select the desired value from the level drop-down list. For example, specify the following values to set the level to report debug messages:
   - For Services, specify SAS Job Execution.
   - For level, specify DEBUG.
   - For name, specify `com.sas.jobexec`. 
Administrative Tasks

Administration of the SAS Job Execution Web Application consists of setting configuration properties and preset job parameters, enabling logon options, and installing sample programs. An account belonging to the SASAdministrators group is required to perform these actions.

Setting Configuration Properties

Configuration properties control the default behavior of the SAS Job Execution Web Application. Use SAS Environment Manager to view and modify these parameters.

Start SAS Environment Manager and then select "All services" from the drop-down menu in the left pane and then select "SAS Job Execution".

The configuration properties are displayed in the right pane and can be changed by selecting the "Add property" option in the upper right corner of the sas.jobexecapp section. To add a new property, scroll to the bottom of the list, click "Add property", and then specify the property name and value in the Add Property dialog box.

The following properties can be changed or added:

<table>
<thead>
<tr>
<th>Services:</th>
<th>SAS Job Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>level:</td>
<td>DEBUG</td>
</tr>
<tr>
<td>name:</td>
<td>com.sas.jobexec</td>
</tr>
</tbody>
</table>

Save | Cancel
### Table 2  Configuration Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>actiondisallow</td>
<td>Comma-separated list of _ACTION values to disallow</td>
<td></td>
</tr>
<tr>
<td>background</td>
<td>SAS folder path and name of HTML file to display for background processing</td>
<td></td>
</tr>
<tr>
<td>contextname</td>
<td>SAS Job Execution compute context</td>
<td>Compute service context name</td>
</tr>
<tr>
<td>createapslist</td>
<td>false</td>
<td>Specifies whether to return the _APSLIST macro variable when a job is executed</td>
</tr>
<tr>
<td>debugdisallow</td>
<td>Comma-separated list of debug values to disallow</td>
<td></td>
</tr>
<tr>
<td>executetimeout</td>
<td>300</td>
<td>Job run time-out in seconds</td>
</tr>
<tr>
<td>expiration</td>
<td>PT30M</td>
<td>Job expiration duration (30 minutes) in the format defined in the W3C XML duration data type</td>
</tr>
<tr>
<td>indextitle</td>
<td>SAS Job Execution</td>
<td>Banner title for the application</td>
</tr>
<tr>
<td>maxfilecount</td>
<td>5</td>
<td>File upload maximum file count</td>
</tr>
<tr>
<td>maxfilesize</td>
<td>100000000</td>
<td>File upload maximum file size in bytes</td>
</tr>
<tr>
<td>waittext</td>
<td>Please wait</td>
<td>Message to display on job wait display</td>
</tr>
<tr>
<td>welcome</td>
<td>SAS folder path of HTML file to display if no parameters are specified</td>
<td></td>
</tr>
</tbody>
</table>

### Setting Preset Parameters

Like configuration properties, preset parameters are defined using SAS Environment Manager. The parameter name/value pairs are set when a job executes and SAS global macro variables are created from the name/value pairs. The preset parameter definitions are in the same pane as the configuration properties. Follow the instructions in “Setting Configuration Properties” on page 19 to access this pane, and then scroll to the Preset Parameters section.

These parameters can be set to fixed strings or values that are substituted when the job is run. The properties that are available for substitution are listed in the Request Properties, System.
The following table lists some common request properties:

<table>
<thead>
<tr>
<th>Recommended SAS Variable Name</th>
<th>Request Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_AUTHTYP</td>
<td>jobexec.auth.type</td>
<td>Name of the authentication scheme that is used (for example, BASIC, SSL, or blank if no protection).</td>
</tr>
<tr>
<td></td>
<td>jobexec.character.encoding</td>
<td>Name of the character encoding that is used in the body of the request.</td>
</tr>
<tr>
<td></td>
<td>jobexec.content.length</td>
<td>Length (in bytes) of the request body, which is made available by the data source. If the length is not known, the value is –1.</td>
</tr>
<tr>
<td></td>
<td>jobexec.content.type</td>
<td>MIME type of the body of the request. If the type is not known, the value is blank.</td>
</tr>
<tr>
<td></td>
<td>jobexec.context.path</td>
<td>Portion of the request URL that indicates the context of the request.</td>
</tr>
<tr>
<td></td>
<td>jobexec.cookies</td>
<td>Cookie strings that the client sent with this request.</td>
</tr>
<tr>
<td></td>
<td>jobexec.header</td>
<td>All HTTP request headers.</td>
</tr>
<tr>
<td></td>
<td>jobexec.header.name</td>
<td>A particular HTTP request header line as it was received, where name is the header name.</td>
</tr>
<tr>
<td></td>
<td>jobexec.header.user-agent</td>
<td>A particular HTTP request header line as it was received, where user-agent is the name of the user agent.</td>
</tr>
<tr>
<td>_HTUA</td>
<td>jobexec.jsessionid</td>
<td>Web application session ID.</td>
</tr>
<tr>
<td>_USERLOCALE</td>
<td>jobexec.locale</td>
<td>Preferred locale in which the client accepts content, based on the Accept-Language header.</td>
</tr>
<tr>
<td></td>
<td>jobexec.method</td>
<td>HTTP method used when this request was made (for example, GET, POST, or PUT).</td>
</tr>
<tr>
<td></td>
<td>jobexec.path</td>
<td>URL pathname.</td>
</tr>
<tr>
<td>Recommended SAS Variable Name</td>
<td>Request Property Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>jobexec.protocol</td>
<td>Name and version of the protocol that the request uses in the form protocol/majorVersion.minorVersion (for example, HTTP/1.1).</td>
</tr>
<tr>
<td></td>
<td>jobexec.query.string</td>
<td>Query string that is contained in the request URL after the path.</td>
</tr>
<tr>
<td>_RMTADDR</td>
<td>jobexec.remote.addr</td>
<td>Internet Protocol (IP) address of the client that sent the request.</td>
</tr>
<tr>
<td>_RMTHOST</td>
<td>jobexec.remote.host</td>
<td>Fully qualified name of the client that sent the request or the IP address of the client if the name cannot be determined.</td>
</tr>
<tr>
<td></td>
<td>jobexec.remote.user</td>
<td>Login ID of the user that made this request if the user has been authenticated. If the user has not been authenticated, the value is blank.</td>
</tr>
<tr>
<td></td>
<td>jobexec.request.inputencoding</td>
<td>Request input encoding. The default encoding is UTF-8.</td>
</tr>
<tr>
<td></td>
<td>jobexec.scheme</td>
<td>Name of the scheme that was used to make this request (for example, HTTP, HTTPS, or FTP).</td>
</tr>
<tr>
<td></td>
<td>jobexec.secure</td>
<td>A value of true or false, indicating whether this request was made using a secure channel, such as HTTPS.</td>
</tr>
<tr>
<td>_URL</td>
<td>jobexec.uri</td>
<td>Part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.</td>
</tr>
<tr>
<td>_VERSION</td>
<td>jobexec.version</td>
<td>SAS Job Execution Web Application version.</td>
</tr>
<tr>
<td>_XFORWARD</td>
<td>jobexec.header.x-forwarded-host</td>
<td>Request HTTP header value for x-forwarded-host.</td>
</tr>
<tr>
<td>_CLIENTNAME</td>
<td>SASJobExecution</td>
<td>Name of the client.</td>
</tr>
</tbody>
</table>
Use a dollar sign character ($) followed by the property name to perform run-time substitution. Any unresolved values result in the corresponding parameter being set to a zero-length string. The following table provides some examples:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_SOME_TEXT</td>
<td>This is some text</td>
<td>Hardcoded text</td>
</tr>
<tr>
<td>_JAVA_VERSION</td>
<td>$java.version</td>
<td>Version of Java on the Java Application Server</td>
</tr>
<tr>
<td>_SCHEME</td>
<td>$jobexec.scheme</td>
<td>Name of the scheme for the request</td>
</tr>
<tr>
<td>_HTTP_HOST</td>
<td>$jobexec.header.host</td>
<td>Host and port number for the request</td>
</tr>
</tbody>
</table>

The last example illustrates how to assign the value of an HTTP request header to a macro variable using the general syntax $jobexec.header.name. Be sure to specify header when referencing an HTTP header.

It is good practice to use an underscore character (_) at the beginning of system macro variable names. Refrain from creating macro variables in your SAS program that begin with an underscore character to avoid overwriting system macro variable values.

### Configuring the Compute Server

By default, each client user must have a user account on the SAS compute server machine to run a request. However, you can use service accounts with the compute server, which enables the administrator to configure a compute server context to be associated with a service account. Then all compute server sessions for that context will run as the specified service account (a shared account). This removes the need for end-users to have operating system accounts on the hosts where the compute server is running. These alternate compute contexts can be used for specific jobs by specifying the context name in the _CONTEXTNAME parameter for the job.

After you configure the compute server to run under a shared account, you can configure the server to be reusable. This enables the server to be reused by a later session instead of terminating when a session ends, which in turn helps performance because you do not have to wait for a new server to start.

See "Server Contexts: How To" in SAS Viya Administration: Server Contexts for more information about configuring the compute server.
Security for SAS Viya Jobs

Setting Authorization for a Folder

By default, the authorization for a top-level folder allows the owner full access to the folder. For more information about viewing and editing authorization for folders and folder objects, see SAS Viya Administration: Content Management.

Changing Access to Application Functions

Access to the SAS Content page for developers and other functions is determined by authorization rules applied to object URIs. Use SAS Environment Manager to locate and then edit or add rules that control which users and groups can access the object URIs.

Using an administrative account, click in the left navigation bar to access the SAS Environment Manager Rules page. If you want to restrict who can create job definitions and job forms, change the rule on /SASJobExecution/developer/** to specify the specific users or groups that should have developer capabilities. Specify /SASJobExecution/developer/** in the Object URI field of the Rules Filter, and then click Apply to filter the rules of this URI:
For more information about general authorization and working with authorization rules or about permissions that you can grant or prohibit, see SAS Viya Administration: General Authorization. For most functions, the SAS Job Execution Web Application checks permissions based on the HTTP request method used to access the URI.

Use the Read permission to control access to the _DEBUG job parameter. For all other URIs, use the Read permission to control access to the URI using the HTTP GET method and CREATE for the POST method.

The following table lists the URIs and default principals for all of the application functions:

**Table 5  Application Function URIs and Principals**

<table>
<thead>
<tr>
<th>Function</th>
<th>Object URI</th>
<th>Default Principal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application root</td>
<td>/SASJobExecution</td>
<td>Authenticated Users</td>
</tr>
<tr>
<td>Administration interface</td>
<td>/SASJobExecution/developer/**</td>
<td>Authenticated Users</td>
</tr>
<tr>
<td>_DEBUG job parameter</td>
<td>/SASJobExecution/debug/**</td>
<td>Authenticated Users</td>
</tr>
<tr>
<td>Display system environment</td>
<td>/SASJobExecution/env/**</td>
<td>SASAdministrators</td>
</tr>
<tr>
<td>Display an HTML input form before the job executes</td>
<td>/SASJobExecution/form/**</td>
<td>Authenticated Users</td>
</tr>
<tr>
<td>Function</td>
<td>Object URI</td>
<td>Default Principal</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Display prompts before the job executes</td>
<td>/SASJobExecution/prompts/**</td>
<td>Authenticated Users</td>
</tr>
<tr>
<td>Standard interface to display and execute jobs</td>
<td>/SASJobExecution/user/**</td>
<td>Authenticated Users</td>
</tr>
<tr>
<td>Administer previous job runs</td>
<td>/SASJobExecution/jobs/**</td>
<td>Authenticated Users</td>
</tr>
<tr>
<td>Schedule jobs</td>
<td>/SASJobExecution/scheduling/**</td>
<td>Authenticated Users</td>
</tr>
<tr>
<td>Administer sample jobs</td>
<td>/SASJobExecution/samples/**</td>
<td>Authenticated Users</td>
</tr>
<tr>
<td>Return the CSS for the current theme</td>
<td>/SASJobExecution/theme</td>
<td>Authenticated Users</td>
</tr>
</tbody>
</table>

### Sharing Jobs and Files with Group Members

You can use SAS Environment Manager to add the following new rules if you want to enable members of a particular group to view jobs and files (such as output and logs) on the Jobs page.

Table 6  Rules and Values for Sharing Jobs and Files

<table>
<thead>
<tr>
<th>Rule Field</th>
<th>Value for Enabling Members to View Files</th>
<th>Value for Enabling Members to View Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object URI</td>
<td>/files/files/**</td>
<td>/jobExecution/jobs/**</td>
</tr>
<tr>
<td>Principal type</td>
<td>Group</td>
<td>Group</td>
</tr>
<tr>
<td>Principal</td>
<td>group-name</td>
<td>group-name</td>
</tr>
<tr>
<td>Rule type</td>
<td>Grant</td>
<td>Grant</td>
</tr>
<tr>
<td>Condition</td>
<td>groupsForUser(#file?.createdBy).contains(&quot;group-name&quot;)</td>
<td>groupsForUser(#job?.createdBy).contains(&quot;group-name&quot;)</td>
</tr>
<tr>
<td>Permissions</td>
<td>Read</td>
<td>Read</td>
</tr>
</tbody>
</table>

For more information about how to add rules, see “Rules Page” in SAS Viya Administration: General Authorization.
Testing the Installation

Executing the Ping Program

A special internal program is available to determine whether the system is functioning correctly:

http://host:port/SASJobExecution/?_program=ping

An HTML page is returned when this job runs to completion with the message:

Job completed successfully

Running a Sample Job

You can further test the installation by running one or more sample jobs, if they are installed on your system. See "Samples" on page 51 for a list of samples and their expected output.

Development Concepts

Creating Jobs Using the %JESBEGIN and %JESEND Macros

Overview of %JESBEGIN and %JESEND Macros

The %JESBEGIN and %JESEND utility macros are automatically added to your code stream. There is no need to include them in your program code.

The %JESBEGIN utility macro sets up the job execution environment and executes before your SAS code. You use job input parameter values to control the macro.

For example, specify _OUTPUT_TYPE=ods_html5 if your job uses ODS to create HTML output. The macro assigns a FILEREF named _WEBOUT to return output to the web browser or client application and issues an ODS HTML5 statement.

Note: The %JESBEGIN macro creates HTML5 output by default.

The %JESEND autocall macro cleans up the job execution environment and executes after the SAS code.
If you specify _OUTPUT_TYPE=none, the %JESBEGIN macro displays a list of global macro variables in the SAS log. Specify _ADDJESBEGINENDMACROS=false to prevent the macros from being added to your code stream.

See "Samples" on page 51 and "%JESBEGIN and %JESEND AutoCall Macros" on page 111 for more information about using these macros.

Creating a Simple Job That Uses DATA Step Code

The following example creates a simple job that uses DATA step code to return HTML to the client.

You can use the SAS Job Execution Web Application to create jobs by using the SAS Content page:

http://host:port/SASJobExecution

Note: You must have access to developer functionality to create jobs.

Perform the following steps to create the sample job:

1. Navigate to a folder location where you want to store the job, and then click.

2. Specify the name for the file (this example uses Simple HTML) and accept the default value for the File type field. Click OK.

3. Click the Simple HTML job, and then click Properties to display the Properties dialog box.
4 Click **Parameters**, and then click **Add a new parameter**. Specify the following properties for the parameter:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>_OUTPUT_TYPE</td>
</tr>
<tr>
<td>Default Value</td>
<td>html</td>
</tr>
<tr>
<td>Field Type</td>
<td>Character</td>
</tr>
</tbody>
</table>

Click **Save**.

This job creates HTML output but not HTML output generated by ODS. See “Creating Simple HTML Output Using ODS” on page 34 for more information about creating ODS output from a job.

5 Click 🖼 and select **Source code**.

6 Enter the following code into the editor window:
* Write the custom HTML to _webout;

```sas
data _null_;
file _webout;
put '<html>'; put '<head><title>Hello World!</title></head>'; put '<body>'; put '<h1>Hello World!</h1>'; put '</body>'; put '</html>'; run;
```

7 Click and then click Close to save the job and close the editor window.

This code writes HTML to the _WEBOUT FILEREF that is assigned by the %JESBEGIN macro. This FILEREF is assigned based on the value specified for the _OUTPUT_TYPE job parameter. The SAS Job Execution Web Application displays the HTML written to this FILEREF. See the sections about assigning a FILEREF in “Development Concepts” on page 27 for more information about this type of FILENAME statement.

**Executing a Job Using Direct URL Access**

You can execute a job by entering a URL into the address bar of your web browser. The URL for the example in “Creating a Simple Job That Uses DATA Step Code” on page 28 is as follows:

http://host:port/SASJobExecution/?_program=/SomeFolder/Simple HTML

Specify the complete path and name of the job in the _PROGRAM URL parameter. Perform the following steps to get a copy of the URL without input parameters:

1 Click a job to select it, and then click and select Properties to display the Properties dialog box.

2 Click Details to see the location of the job definition details and the URL for submitting the job. For the Simple HTML example, you might see values such as the following:

   Job details /jobDefinitions/definitions/493ba851-303b-4fb1-ac12-22199e085320
   Job submit http://host:port/SASJobExecution/?_program=/SomeFolder/Simple HTML

3 Copy the value for Job submit.

SAS global macro variables are created from all query string parameters to the right of the question mark (?) in the URL. These macro variables are available for use in your SAS program. In this case, a macro variable named _PROGRAM is created with a value of /SomeFolder/Simple HTML.

The program creates the following output:

**Hello World!**

You can also execute a job by accessing a link in a web page. Specify the previous URL in the HREF attribute of an anchor tag:

```html
<a href="http://host:port/SASJobExecution/?_program=/SomeFolder/Simple HTML">Click here to execute job</a>
```
Executing a Job Using the SAS Job Execution Web Application

The SAS Content page of the SAS Job Execution Web Application provides a basic user interface to list and execute jobs.

Note: Users who do not have access to developer functionality see fewer icons and files. Developers see the job forms as well as the job definitions. Other users see only the job definitions.

In the left pane, navigate to the job that you want to execute, select it, and then click ▶ to execute the job.

The job output is displayed in the right pane.

Passing User Input to a Job Using the Query String

Most jobs require information from the client to perform their intended function. This information can be in the form of presentation options for a report, selection criteria for data to be analyzed, names of data tables to be used or created, or an unlimited number of other possibilities.

Input parameters are the most common way to deliver information from a client to a job. They are defined as name/value pairs and appear in your SAS program as global macro variables. The _PROGRAM parameter used in “Executing a Job Using Direct URL Access” on page 30 is an example of a URL parameter.

The simplest way to pass user input to a job is by specifying name/value pairs in the SAS Job Execution Web Application query string. Consider this modification to the web address:

http://host:port/SASJobExecution/?_program=/SomeFolder/Simple HTML&mynname=John

A global macro variable named MYNAME with a value of John is created before the SAS code is executed, and it is available for use in your SAS program. Make the following change to the code from "Creating a Simple Job That Uses DATA Step Code" on page 28:

* Declare input parameter;

%global MYNAME;

* Write the custom HTML to _webout;

data _null_;  
file _webout;  
put '<html>';  
put '<head><title>Hello World! </title></head>' ;  
put '<body>';  
put '<h1>Hello %sysfunc(htmlencode(&MYNAME))!</h1>';  
put '</body>';  
put '</html>';  
run;

It is good practice to declare input parameters at the beginning of the program so that macro variables resolve. Also, use double quotation marks in the PUT statement to resolve the macro variable.

Executing the job using the previous web address creates the following output:
Hello John!

Passing User Input to a Job Using a Job Definition Parameter

Run the Simple HTML job that you created previously without specifying a value for the MYNAME input parameter in the URL:

http://host:port/SASJobExecution/?_program=/SomeFolder/Simple HTML

The following output is displayed:

Hello!

Perform the following steps to specify a default value for the MYNAME input parameter by adding a job definition parameter:

1. Use the SAS Content page to navigate to the Simple HTML job, select it, and then click and select Properties.

   Note: You must have access to developer functionality to modify properties for a job.

2. Click Parameters and then click Add a new parameter. Specify the following properties for the parameter:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>myname</td>
</tr>
<tr>
<td>Default Value</td>
<td>John</td>
</tr>
<tr>
<td>Field Type</td>
<td>Character</td>
</tr>
</tbody>
</table>

3. Select Required only if you want to ensure that a non-blank value be specified for the parameter at run time.

4. Click Save to save the parameter.

5. Run the job again using the previous URL. The following output is now displayed:

   Hello John!

Passing User Input to a Job Using a Job Form

You can use the SAS Content page to create HTML input forms or prompts that accept input from a user and then pass that input to the SAS code. This example uses the Simple HTML job that was created in “Creating a Simple Job That Uses DATA Step Code” on page 28. The example creates an HTML input form to replace the technique of passing input to the SAS program using the query string.
Note: You must have access to developer functionality to create job forms.

When you create a job form with the same name as the job that is located in the same folder as the job, it can be automatically displayed when the job executes. Create a job parameter named _ACTION with the following properties to take advantage of this behavior for an HTML input form:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>_ACTION</td>
</tr>
<tr>
<td>Default Value</td>
<td>form,execute</td>
</tr>
<tr>
<td>Field Type</td>
<td>Character</td>
</tr>
</tbody>
</table>

If an HTML input form exists, then it is displayed. Otherwise, the job executes.

Perform the following steps to create an HTML input form:

1. Click the Simple HTML job definition that you just created, and then click and select **New HTML form**.

2. Enter the following HTML code into the editor window:

```html
<!DOCTYPE html>
<head>
<title>Simple HTML Example</title>
</head>
<body>

<h1>Simple HTML Example</h1>

<form action="/SASJobExecution/" target="_tab">
  <input type="hidden" name="_program" value="$PROGRAM$">
  <input type="hidden" name="_action" value="execute">
  <label>Specify a name for the greeting: </label>
  <input type="text" name="myname" value="World" required>
  
  <input type="submit" value="Run code">
</form>

</body>
</html>
```

Alternatively, you can use a text editor or web development tool to create the HTML, and then copy and paste it into the editor window.

Always specify /SASJobExecution/ in the ACTION attribute of the FORM tag to indicate that the form data is submitted to the SAS Job Execution Web Application for processing. You can specify _tab in the TARGET attribute to force the output to always appear in a new tab within the application. Omit this attribute if you do not want this behavior.
The first input tag specifies that a non-visual object named _PROGRAM has a value of $\text{PROGRAM}$. When the HTML input form is displayed, the path and name of the program to execute are substituted for $\text{PROGRAM}$.

The second input tag specifies a value of execute for the _ACTION parameter, which overrides the default value specified in the job parameter and executes the job.

The third input tag prompts the user for the name to use in the greeting. This object is named MYNAME and its default value is World.

The last input tag displays a button with the label Run code.

Click [Run code], and then click Close to save the HTML input form and close the editor window.

Click the Simple HTML job to select it, and then click [Run code]. The following HTML input form appears in the right pane:

```
Simple HTML Example

Specify a name for the greeting: World

Run code
```

Specify Jane as the name for the greeting, and then click Run code.

The web browser uses data from all form elements except the submit button to automatically construct a URL similar to the URL in "Passing User Input to a Job Using the Query String" on page 31. The form data is submitted when the button is clicked, resulting in the following URL:

http://host:port/SASJobExecution/?_program=/SomeFolder/Simple HTML&myname=Jane

The results are displayed in a new browser window or tab, using the updated value of the MYNAME macro variable:

```
Hello Jane!
```

Creating Simple HTML Output Using ODS

The Output Delivery System (ODS) enables you to generate different types of output from your procedure code. An ODS destination controls the type of output that is generated (HTML, RTF, PDF, and so on). An ODS style controls the appearance of the output.

Many jobs create ODS HTML as their primary type of output. The %JESBEGIN macro can issue an ODS statement in addition to the _WEBOUT FILEREF. Add the following parameters to the job to use the ODS HTML5 destination and the HTMLBlue style:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>_ODSTYLE</td>
</tr>
<tr>
<td>Default Value</td>
<td>HTMLBlue</td>
</tr>
<tr>
<td>Field Type</td>
<td>Character</td>
</tr>
</tbody>
</table>
This job uses PROC PRINT to display all of the data in the SASHELP.CLASS table:

```sas
* Display the SASHELP.CLASS table;

title 'Student Data';

proc print data=sashelp.class noobs;
  var name sex age height weight;
run; quit;
```

Here is the output:

<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>59.0</td>
<td>112.5</td>
</tr>
<tr>
<td>Alice</td>
<td>F</td>
<td>13</td>
<td>60.5</td>
<td>84.0</td>
</tr>
<tr>
<td>Barbara</td>
<td>F</td>
<td>13</td>
<td>55.3</td>
<td>80.0</td>
</tr>
<tr>
<td>Carol</td>
<td>F</td>
<td>14</td>
<td>82.0</td>
<td>102.5</td>
</tr>
<tr>
<td>Henry</td>
<td>M</td>
<td>14</td>
<td>83.5</td>
<td>122.5</td>
</tr>
<tr>
<td>James</td>
<td>M</td>
<td>12</td>
<td>87.3</td>
<td>83.0</td>
</tr>
<tr>
<td>Jane</td>
<td>F</td>
<td>12</td>
<td>59.8</td>
<td>84.5</td>
</tr>
<tr>
<td>Janet</td>
<td>F</td>
<td>15</td>
<td>82.5</td>
<td>112.5</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>M</td>
<td>13</td>
<td>82.5</td>
<td>84.0</td>
</tr>
<tr>
<td>John</td>
<td>M</td>
<td>12</td>
<td>59.0</td>
<td>82.0</td>
</tr>
<tr>
<td>Joyce</td>
<td>F</td>
<td>11</td>
<td>51.3</td>
<td>50.5</td>
</tr>
<tr>
<td>Judy</td>
<td>F</td>
<td>14</td>
<td>94.3</td>
<td>90.0</td>
</tr>
<tr>
<td>Louise</td>
<td>F</td>
<td>12</td>
<td>59.5</td>
<td>77.0</td>
</tr>
<tr>
<td>Mary</td>
<td>F</td>
<td>16</td>
<td>88.5</td>
<td>112.0</td>
</tr>
<tr>
<td>Philip</td>
<td>M</td>
<td>18</td>
<td>72.0</td>
<td>150.0</td>
</tr>
<tr>
<td>Robert</td>
<td>M</td>
<td>12</td>
<td>84.8</td>
<td>128.0</td>
</tr>
<tr>
<td>Ronald</td>
<td>M</td>
<td>15</td>
<td>87.0</td>
<td>133.0</td>
</tr>
<tr>
<td>Thomas</td>
<td>M</td>
<td>11</td>
<td>57.5</td>
<td>85.0</td>
</tr>
<tr>
<td>William</td>
<td>M</td>
<td>15</td>
<td>55.5</td>
<td>112.0</td>
</tr>
</tbody>
</table>

Creating Simple PDF or RTF Output Using ODS

You can make small changes to the sample in "Creating Simple HTML Output Using ODS" on page 34 to create PDF or RTF output. Specify `ods_pdf` for the `_OUTPUT_TYPE` input parameter to create PDF output and `ods_rtf` to create RTF output. The PDF output is rendered by the web browser, and the RTF output is downloaded as a file named SASResults.rtf that can be opened using an application such as Microsoft Word.

Using Input Parameters with Multiple Values

Parameters with multiple values (or, alternatively, multiple input parameters with the same name) can be useful in some jobs. For example, an HTML input form might contain a multiple selection list box named COLS that allows the user to choose which columns of a table to display. This example shows a parameter with multiple values.

Create a job definition named Multiple Input Values with the following parameters:
Add the following code:

```html
<param_list macro definition here>

* Convert the selections to a space-separated list;

`%param_list(mvar=cols, outvar=column_list)`

title 'Student Data';

proc print data=sashelp.class;
  var &COLUMN_LIST;
run; quit;
```

This example uses a multiple selection list box to choose the columns to display from the SASHELP.CLASS table. Click the Multiple Input Values job definition that you just created, and then click ✔ and select **New HTML form** to create an HTML input form with the following HTML:

```html
<!DOCTYPE html>
<head>
title>Multiple Input Values Example</title>
</head>
<body>
<h1>Multiple Input Values Example</h1>
<form action="/SASJobExecution/" target="_tab">
  <input type="hidden" name="_program" value="$PROGRAM$">
  <input type="hidden" name="_action" value="execute">

  Use Ctrl+Click to choose columns to display: 
  <select name="cols" multiple required size="5">
    <option value='name'>First Name</option>
    <option value='sex'>Gender</option>
    <option value='age'>Age (y)</option>
    <option value='height'>Height (in)</option>
    <option value='weight'>Weight (lb)</option>
  </select>

  <input type="submit" value="Run code">
</form>
```
The MULTIPLE attribute of the SELECT tag indicates that multiple selections are allowed, and the optional SIZE attribute specifies the number of rows to display when the HTML page is rendered. The name of the SAS macro variable COLS is specified in the NAME attribute.

The OPTION tags specify the values sent to the SAS program as well as the values displayed when the HTML page is rendered. This example uses display values that differ from the macro variable values:

<table>
<thead>
<tr>
<th>Display Value</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>name</td>
</tr>
<tr>
<td>Gender</td>
<td>sex</td>
</tr>
<tr>
<td>Age (y)</td>
<td>age</td>
</tr>
<tr>
<td>Height (in)</td>
<td>height</td>
</tr>
<tr>
<td>Weight (lb)</td>
<td>weight</td>
</tr>
</tbody>
</table>

Here is a partial view of the rendered HTML file:

If you select only **First Name** and then submit the form, a global macro variable named COLS is created with a value of *name*. If you select **First Name**, **Age (y)**, and **Height (in)** and then submit the form, the following macro variables are created:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Macro Variable Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLS</td>
<td>name</td>
<td>Specifies the first value</td>
</tr>
<tr>
<td>COLS0</td>
<td>3</td>
<td>Specifies the number of values</td>
</tr>
<tr>
<td>COLS1</td>
<td>name</td>
<td>Specifies the first value</td>
</tr>
<tr>
<td>COLS2</td>
<td>age</td>
<td>Specifies the second value</td>
</tr>
<tr>
<td>COLS3</td>
<td>weight</td>
<td>Specifies the third value</td>
</tr>
<tr>
<td>COLS_COUNT</td>
<td>3</td>
<td>Specifies the number of values</td>
</tr>
</tbody>
</table>
Because macro variables cannot hold more than one value, a numeric suffix is added to the parameter name to distinguish between values. The number of values is set in the `param-name0` and `param-name_COUNT` variables. The first value is set in the `param-name1` variable, and so on, as shown in the previous table. Note that the original parameter macro variable is always set to the first parameter value.

This format is seldom useful in SAS code. For example, the pseudo-array of user selections must be transformed before they can be used in a VAR or SELECT statement. You can use the PARAM_LIST macro available in “PARAM_LIST Macro” on page 117 to convert the user selections into a usable format.

The COLS selection list in the HTML input form is used to choose one or more columns in the SASHELP.CLASS table. The PARAM_LIST macro takes the individual selections and converts them to a single list that is used in the VAR statement.

For example, if the user selects First Name, Age (y), and Height (in), then the macro variable COLUMN_LIST, which was created by the PARAM_LIST macro, resolves to the following:

```
name age height
```

See “PARAM_LIST Macro” on page 117 for more information and examples.

### Linking One Job to Another (Drill Down)

You might want to display summarized information with the option to click a link to display more detailed data. This is an example of performing drill down. Developing this type of application usually involves at least two different jobs: one job to create the summarized information with the links, and a second job that displays the detailed data related to that link.

This example summarizes sales data in the SASHELP.SHOES table by sales region, and then uses PROC PRINT to display it with links to the detailed data:

<table>
<thead>
<tr>
<th>Region</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asia</td>
<td>$2,242,693</td>
</tr>
<tr>
<td>Asia</td>
<td>$480,231</td>
</tr>
<tr>
<td>Canada</td>
<td>$2,265,712</td>
</tr>
<tr>
<td>Central America/Caribbean</td>
<td>$3,857,753</td>
</tr>
<tr>
<td>Eastern Europe</td>
<td>$2,394,940</td>
</tr>
<tr>
<td>Middle East</td>
<td>$5,031,779</td>
</tr>
<tr>
<td>Pacific</td>
<td>$2,299,794</td>
</tr>
<tr>
<td>South America</td>
<td>$2,434,703</td>
</tr>
<tr>
<td>United States</td>
<td>$5,403,028</td>
</tr>
<tr>
<td>Western Europe</td>
<td>$4,873,000</td>
</tr>
</tbody>
</table>

Detailed sales information is displayed when you click a link. Here is the detailed data for Asia:

```
Name: John Smith
Age: 35
Height: 70 in
Sales: $12,000
```
Create a job definition named Drilldown with the following parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Field Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>_OUTPUT_TYPE</td>
<td>ods_html5</td>
<td>Character</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
<td>Character</td>
</tr>
</tbody>
</table>

Add the following code to produce the summarized report with the drill-down links:

* Summarize the data;

```sas
proc means data=sashelp.shoes sum noprint;
  var sales;
  class region;
  output out=work.shoes_summary(where=(_type_ eq 1)) sum=sales;
run; quit;
```

* Define the base URL for the drill-down link;

```sas
%let BASE_URL=&_URL.?_program=  
/SomeFolder/Drilldown2&_action=wait&nbsp;&lt;br
```

* Set the ODS escape character;

```sas
ods escapechar='^';
```

* Add the drill-down links to the summarized data;

```sas
data work.shoes_summary;
set work.shoes_summary;
```
length region_link varchar(1024);

region_link = "{style [url='&BASE_URL= ||
urlencode(strip(region)) ||
"{}" ||
strip(region) ||
'}";
run;

* Display the summarized data with drill-down links;

title 'Sales Totals by Region';

proc print data=work.shoes_summary noobs label;
var region_link sales;
sum sales;
label region_link = 'Region'
sales = 'Sales';
format sales dollar11.;
run; quit;

The BASE_URL macro variable is used to create a portion of the drill-down link. The _URL reserved macro variable ensures that the URL is valid. The job that displays the detail data, Drilldown2, is referenced here. This job is created in the next step, and you must specify the full path to that job.

The complete drill-down link text is stored in the REGION_LINK variable. ODS inline formatting is used to create the link using the URL style attribute. This technique creates links for HTML as well as other ODS output formats. The general syntax of the inline style is as follows:

^{style [url='URL-of-second-job']link-text}

Here is an example of the value of the REGION_LINK variable created in the code:

^{style [url='/SASJobExecution/?_program=/SomeFolder/Drilldown2&region=Asia']Asia}

The value of the sales region of interest is passed to the Drilldown2 job using the REGION input parameter.

The STRIP function removes any trailing blanks in the data value, and the URLENCODE function handles parameter values that need to be encoded:

^{style [url='/SASJobExecution/?_program=/SomeFolder/Drilldown2&region=Central%20America%20Caribbean']Central America/Caribbean}

The PRINT procedure displays the summarized data with links to execution of the detail data program.

Next, create a job definition named Drilldown2 that displays the detail data using the following parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Field Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>_OUTPUT_TYPE</td>
<td>ods_html5</td>
<td>Character</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
<td>Character</td>
</tr>
</tbody>
</table>

Add the following code to produce the detailed report for a specified region:
* Declare input parameter;

%global REGION;

* Include the value of the REGION input parameter in the report title;

title "Detailed Sales Information for &REGION";

proc report data=sashelp.shoes nowd;
  where (region eq "&REGION");

  column region subsidiary product sales inventory returns;

  define region       / order noprint 'Region';
  define subsidiary   / order         'Subsidiary';
  define product      / order         'Shoe Style';
  define sales        / sum           'Sales';
  define inventory    / sum           'Inventory';
  define returns      / sum           'Returns';

  break after subsidiary / summarize suppress style=header;
  rbreak after           / summarize;
run; quit;

Though not required, it is good practice to declare all input parameters using a %GLOBAL statement. The value of the REGION input parameter, which is passed to the program as a URL parameter from the link in the Drilldown job, is used in the report title and also to subset the data so that only the detailed information is displayed for the specified region.

Run the Drilldown job, and then click a link for a sales region. The detailed data for the region appears.

Creating Jobs without Using the %JESBEGIN and %JESEND Macros

Overview of Jobs without the %JESBEGIN and %JESEND Macros

In some cases, you might need to use very specific FILENAME and ODS statements. In these cases, it is best to prevent the %JESBEGIN utility macro from generating these statements for you.

Specify the input parameter _OUTPUT_TYPE=none to suppress the generation of these statements. Alternatively, you can specify the _ADDJESBEGINNENDMACROS=false input parameter. An advantage of specifying _OUTPUT_TYPE=none is that the %JESBEGIN macro displays the macro variables created from input parameters.

This section provides a sample job that requires specific FILENAME and ODS statements. It also provides information to help you construct your own FILENAME statements. All of the techniques discussed in previous sections can be used when you specify your own FILENAME and ODS statements.

Sending ODS Output to an Email Recipient

The following example sends ODS output as the body of an email message.
Create a job definition named Email Report where the _OUTPUT_TYPE parameter has a value of none. Add the following code:

* Close all open destinations;

    ods _all_ close;

* ODS output is sent directly to the email recipient;

    filename mail email 'email-recipient@email-recipient-domain'
      subject='Your SAS Report' type='text/html';

* The HTML3 destination provides better rendering in some email clients;

    ods html3 file=mail style=HTMLBlue;

title 'Student Data';

proc print data=sashelp.class; run; quit;

    ods html3 close;

%let _STATUS_MESSAGE=Email sent.;

Specify appropriate values for email-recipient and email-recipient-domain. Use the HTML3 destination because some email clients do not support the HTML created by the HTML4 and HTML5 destinations.

The job does not create visual output displayed by the web browser client. You can use _STATUS_MESSAGE to display a message in the web browser.

You might have to specify one or more system options to successfully send email. See SAS System Options: Reference for more information about email communications system options.

Assigning a FILEREF for HTML Output

A FILENAME statement is required to define the location of your output. You can choose any valid name for the FILEREF, but the device type (FILESRVC) and the PARENTURI option should be specified exactly as follows:

    filename _webout filesrvc parenturi="&SYS_JES_JOB_URI"
      name='_webout.htm';

Unlike the DISK device type, the FILESRVC device writes files to the SAS Infrastructure Data Server using the Files service, not the external file system. The FILESRVC access method creates global macro variables of the form _FILESRVC_fileref, where FILEREF is the fileref used in the FILENAME statement. This macro variable provides a relative URL that can be used to reference and retrieve the file using the Files service (for example, /files/files/74d8179e-e922-4b58-a8fe-0863b2aa3bfc).

See “Report with Download Links” on page 82 for an example that uses this macro variable.

The SYS_JES_JOB_URI macro variable provides a reference to the job execution object. When used with the PARENTURI option, this ensures that the file is associated with the job execution object. All files associated with a job execution object can be displayed by the SAS Job Execution Web Application until the job expires. By default, job output is deleted 30 minutes after it is created. See “Job Output Expiration” on page 15 for more information about how to change the expiration time.

See “Saving Job Output” on page 17 for more information about how to save a permanent copy of your job output files.
In most cases, you should use the name _webout.htm because the SAS Job Execution Web Application searches the job results object for an entry named _webout.* and displays the first result that it finds. This behavior can be altered using the _RESULTFILE parameter.

Assigning a FILEREF for Other Types of Output

The FILENAME statement that supports other types of ODS output is similar to the format used for HTML. The following examples show some common ODS output formats. See SAS Global Statements: Reference for more information about the FILENAME statement.

Use the following format if your web browser supports rendering PDF files:

```
filename _webout filesrvc parenturi="&SYS_JES_JOB_URI"
  name='._webout.pdf';
```

Use the following format if your web browser does not support rendering PDF files or if you want the content to be downloaded as a file. Specify the desired file name in the FILENAME attribute of the CONTENTDISP option:

```
filename _webout filesrvc  parenturi="&SYS_JES_JOB_URI"
  name='._webout.pdf'
  contentdisp='attachment; filename="MyFile.pdf"';
```

See “Report with Download Links” on page 82 for an example that uses this format.

Use the following format to download RTF content:

```
filename _webout filesrvc  parenturi="&SYS_JES_JOB_URI"
  name='._webout.rtf'
  contentdisp='attachment; filename="MyFile.rtf"';
```

Most web browsers support rendering XML content:

```
filename _webout filesrvc  parenturi="&SYS_JES_JOB_URI"
  name='._webout.xml';
```

Use the following format to download XML content:

```
filename _webout filesrvc  parenturi="&SYS_JES_JOB_URI"
  name='._webout.xml'
  contentdisp='attachment; filename="MyFile.xml"';
```

Use one of the following formats to handle JSON content:

```
filename _webout filesrvc  parenturi="&SYS_JES_JOB_URI"
  name='._webout.json';
```

```
filename _webout filesrvc  parenturi="&SYS_JES_JOB_URI"
  name='._webout.json'
  contentdisp='attachment; filename="MyFile.json"';
```

Use the following format if your program uses the tagsets.ExcelXP ODS destination:

```
filename _webout filesrvc  parenturi="&SYS_JES_JOB_URI"
  name='._webout.xml'
  contenttype='application/vnd.ms-excel'
  contentdisp='attachment; filename="MyFile.xml"';
```

See “Report with Download Links” on page 82 for an example that uses this format.
Assigning a FILEREF to Store Source Code in the Files Service

If you have a large program or want to share code with other users, you can store source code in the Files service. An additional benefit is that if you use code from the Files service in a scheduled job, the source code can be changed without rescheduling the job.

The following example shows how to format the source code if you want to access the file using a URI:

```sas
filename mycode filesrvc '/files/files/URI';
%include mycode;
```

The following format shows how to format the source code if you want to access the file by name and folder:

```sas
filename mycode filesrvc folderpath='/MyFolder/programs' name='MyProgram.sas';
%include mycode;
```

In both of these examples, mycode is the fileref that you assign. See “FILESRVVC Options” in SAS Global Statements: Reference for more information about how to use the Files service.

Advanced Programming

Sending JSON Data to a Job Using an Input Parameter

Data in JSON format can be sent to a job as an input parameter if the data contains fewer than 32,767 bytes. Data that is sent in this way can be accessed as a macro variable in the SAS job. In the following example, the MYJSON parameter contains JSON data that was sent to the JSON1 job. Note that %7B and %7D represent the URL-encoded values for the left and right brace characters, { and }, respectively:

```
http://server:port/SASJobExecution/?_program=/Test/json1&myjson=%7B"aaa":"AAA",
"bbb":222, "ccc":false%7D
```

The SAS program for the JSON1 job uses the JSON LIBNAME engine to read the JSON data from the input parameter and then to convert it to SAS tables:

* Declare input parameter;

```sas
%global MYJSON;
```

* Copy the JSON data from input parameter to a file;

```sas
filename indata temp;

data _null_; file indata;
  length str $32767;
  str = resolve(symget('myjson'));
  put str;
run;
```
* Use the JSON engine to provide read-only sequential access to JSON data;

```
libname indata json;
```

title 'ALLDATA Table from JSON Input';
proc print data=indata.alldata; run; quit;

title 'ROOT Table from JSON Input';
proc print data=indata.root; run; quit;

The PROC PRINT output is shown in the following figure:

![ALLDATA Table from JSON Input](image1)

![ROOT Table from JSON Input](image2)

---

**Sending JSON Data to a Job by Uploading a File**

JSON data that exceeds 32,767 bytes can be sent to the job as a file. See the sample “Upload a File” on page 106 for information about how to upload a file. Use the following SAS program to access the uploaded data:

* Reference the uploaded JSON data;

```
filename indata filesrvc "&_WEBIN_FILEURI";
```

* Use the JSON engine to provide read-only sequential access to JSON data;

```
libname indata json;
```

title 'ALLDATA Table from JSON Input';
proc print data=indata.alldata; run; quit;

title 'ROOT Table from JSON Input';
proc print data=indata.root; run; quit;

The output is shown in “Sending JSON Data to a Job Using an Input Parameter” on page 44.
Executing a Job Using JavaScript - Sending Small Data to the Job

You can use this technique to control how job output is handled. The following example uses the FormData JavaScript object and strings to send input parameters to the previous JSON1 job. This is another way to send JSON data containing less than 32,767 bytes to a job.

The JavaScript code in the following HTML input form dynamically creates a form with parameters, submits the form using the POST method, and then displays the output in a DIV element:

```html
<!DOCTYPE html>
<html>
<head>
<title>JavaScript Job Execution</title>
<script>
function submitForm() {
  var formData = new FormData();
  // Your small JSON object here
  var json = {aaa:"AAA", bbb:222, ccc:false};
  // Create the input parameter for the JSON data
  formData.append("myjson", JSON.stringify(json));
  // Create other input parameters
  formData.append("_program", "/Folder/json1");
  formData.append("_action", "execute");
  formData.append("_csrf", "$CSRF$" );
  // Create the request object
  var request = new XMLHttpRequest();
  request.addEventListener("error", function(event) {
    alert("Something went wrong.");
  });
  request.onreadystatechange = function() {
    if (this.readyState == 4) {
      if (this.status == 200) {
        // Display the results in the DIV
        document.getElementById("JobResults").innerHTML = this.responseText;
      } else {
        document.getElementById("JobResults").innerHTML = "Status: " + this.status;
      }
    }
  }
  request.open("post", "/SASJobExecution/" );
  // Submit the form
  request.send(formData);
  // Display a temporary message in the DIV
  document.getElementById("JobResults").innerHTML = "Please wait ..." ;
}
</script>
</head>
<body>
</body>
```
Executing a Job Using JavaScript - Sending Large Data to the Job

The following example is like the previous one, except that more than 32,767 bytes of data can be sent to the job. This is accomplished using the JavaScript Blob object in the HTML input form:

```html
<!DOCTYPE html>
<html>
<head>
<title>JavaScript Job Execution</title>
</head>
<script>
function submitForm() {
  var formData = new FormData();
  // Your large JSON object here
  var json = {aaa:"AAA", bbb:222, ccc:false};
  var blob = new Blob([JSON.stringify(json)], {type : 'application/json'});
  // Create the input parameter for the JSON data
  formData.append("myjsonfile", blob);
  // Create other input parameters
  formData.append("_program", "/Folder/json2");
  formData.append("_action", "execute");
  formData.append("_csrf", "$CSRF$" abdominal building energy calculation calibration, river cold
  // Create the request object
  var request = new XMLHttpRequest();
  request.onreadystatechange = function(event) {
    if (this.readyState == 4) {
      if (this.status == 200) {
        // Display the results in the DIV
        document.getElementById("JobResults").innerHTML = this.responseText;
      } else { // Something went wrong.
    
    request.onreadychange = function() {
      if (this.readyState == 4) {
        if (this.status == 200) {
          // Display the results in the DIV
          document.getElementById("JobResults").innerHTML = this.responseText;
        } else {
          alert("Something went wrong.");
        }
      }
    }
  }
}
</script>
</body>
</html>
```

When you make a POST request, you must specify the _CSRF input parameter exactly as shown. This tag ensures that the request is considered non-malicious by sending a Cross-Site Request Forgery token to the server.

See “Sending JSON Data to a Job Using an Input Parameter” on page 44 for the results and the SAS code used to process the data.
The JSON data is uploaded as a file using the technique in “Sending JSON Data to a Job by Uploading a File” on page 45. See that section for the results and the SAS code used to process the data.

Returning JSON Data from a Job

The Simple JSON sample (see “Simple JSON” on page 96) displays the SASHELP.CLASS table in JSON format. You can use JavaScript to execute the job and then post-process the JSON data instead of displaying it. For example, you might want to use the returned JSON data with a JavaScript object, such as a grid or a chart object.

The following HTML input form executes the Simple JSON sample and then stores the returned JSON in the JSONString variable:

```html
document.getElementById("JobResults").innerHTML = "Status: " + this.status;
}
};
request.open("post", "/SASJobExecution/*");
// Submit the form
request.send(formData);
// Display a temporary message in the DIV
document.getElementById("JobResults").innerHTML = "Please wait ...";
}
</script>
</head>
<body>
<!-- Other content of your web application here -->
<div id="JobResults"></div>
<script>submitForm();</script>
<!-- Other content of your web application here -->
</body>
</html>
```

The JSON data is uploaded as a file using the technique in “Sending JSON Data to a Job by Uploading a File” on page 45. See that section for the results and the SAS code used to process the data.
request.addEventListener("error", function(event) {
  alert("Something went wrong.");
});
request.onreadystatechange = function() {
  if (this.readyState == 4) {
    if (this.status == 200) {
      // Store the returned JSON data in a variable for later use
      var JSONString = this.responseText;
      // Your code to post process the JSON data here
    }
    else {
      alert("Status: " + this.status);
    }
  }
};
request.open("post", "/SASJobExecution/");
// Submit the form
request.send(formData);
}
submitForm();
</script>
</head>
<body>
<!--[Content of your web application here -->

</body>
</html>

Returning a List of Output Files in JSON Format

The Report with Download Links sample (see “Report with Download Links” on page 82) creates XML for use with Microsoft Excel, PDF, and HTML output files. Your application might retrieve this output and provide special handling of the files. You can use the _ACTION and _RESULTFILE input parameters, which is discussed in “Specifying Output Files” on page 12, to return a list of output files in JSON format:

```
[
  { "name": "Class.pdf",
    "href": "/files/files/3db9d66-106b-4403-aea9-65a6ea3cb514/content"
  },
  { "name": "Class.xml",
    "href": "/files/files/dcef367b-4c37-48c8-b53d-29e6189c2dc2/content"
  },
  { "name": ".webout.htm",
    "href": "/files/files/3ee1ceef-8ff3-46ae-8193-6e408df0a4f/content"
  }
]
```

Your application can use the URIs in the HREF keys to retrieve the file content and handle it appropriately.
The following HTML input form executes the Report with Download Links sample and then stores the returned list of files in the JSONString variable:

```html
<!DOCTYPE html>
<html>
<head>
<title>JavaScript Job Execution</title>
<script>
function submitForm() {
  var formData = new FormData();
  // Create input parameters
  formData.append("_program", "*/Folder/Report with Download Links");
  formData.append("_action", "json,execute");
  formData.append("_resultfile", "*");
  formData.append("_csrf", "$CSRF$");
  // Create the request object
  var request = new XMLHttpRequest();
  request.addEventListener("error", function(event) {
    alert("Something went wrong.");
  });
  request.onreadystatechange = function() {
    if (this.readyState == 4) {
      if (this.status == 200) {
        // Store the returned JSON data in a variable for later use
        var JSONString = this.responseText;
        // Your code to post process the JSON data here
      } else {
        alert("Status: " + this.status);
      }
    } else {
    }
  request.onreadystatechange = function() {
    if (this.readyState == 4) {
      if (this.status == 200) {
        // Store the returned JSON data in a variable for later use
        var JSONString = this.responseText;
        // Your code to post process the JSON data here
      } else {
        alert("Status: " + this.status);
      }
    }
  request.open("post", "/SASJobExecution/");
  // Submit the form
  request.send(formData);
}
submitForm();
</script>
</head>
<body>
<!-- Content of your web application here -->
</body>
</html>
```

**Working with SAS Viya Services**

You can use the HTTP procedure to access any resource that supports HTTP requests. This includes external web services and SAS Viya services. See *Base SAS Procedures Guide* for more information about PROC HTTP. See [https://developer.sas.com/apis/rest/](https://developer.sas.com/apis/rest/) for more information about SAS Viya services.
You can use the following code to call the Job Definitions service to retrieve the first 50 job definitions in JSON format stored on your system and then display some of the fields:

* Base URI for the service call;

```sas
%let BASE_URI=%sysfunc(getoption(servicesbaseurl));
```

* FILEREFs for the response and the response headers;

```sas
filename resp     temp;
filename resp_hdr temp;
```

```sas
proc http url="&BASE_URI/jobDefinitions/definitions/?limit=50"
  method='get'
  oauth_bearer=sas_services
  out=resp
  headerout=resp_hdr
  headerout_overwrite;
run; quit;
```

* Use the JSON engine to provide read-only sequential access to JSON data;

```sas
libname resp json;
```

```sas
title 'Job Definitions';
```

```sas
proc print data=resp.items;
  var name creationTimeStamp createdBy modifiedTimeStamp modifiedBy description;
run; quit;
```

The SAS_SERVICES keyword specified in the OAUTH_BEARER option ensures that an access token is obtained using the identity of the user executing the job.

---

**Samples**

**Accessing the Samples**

Information about the sample jobs that are supplied by SAS is provided in this section. Use the Standard user interface to navigate to the location of the sample jobs, click a job to select it, and then click ▶ to execute it. The HTML input form or prompts are displayed in the right pane, and the output is displayed in a new tab within the application.

**Items Common to Most Samples**

Many samples use the same or similar job definition parameters and HTML markup in their input forms. These common items are explained in this section instead of repeating the explanation for each sample.
Job Definition Parameters

All sample jobs include the _ACTION input parameter to display their respective HTML input forms or prompts. Additional parameters are specified to ensure that the jobs run successfully if no parameters are specified in the URL, as discussed in “Executing a Job Using Direct URL Access” on page 30.

The following table contains commonly used parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,prompts,execute</td>
<td>Displays the HTML input form or prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>varies</td>
<td>Specifies the type of output created by the job, which can be none</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
<td>Specifies the name of the ODS style if the job creates ODS output</td>
</tr>
</tbody>
</table>

%JESBEGIN and %JESEND Macros

The %JESBEGIN macro is automatically executed before the first line of the program code. This macro uses the values of the _OUTPUT_TYPE and _ODS* parameters to configure job output. In most cases, a FILENAME statement is issued to return output to the web browser. An ODS statement is issued in addition to the FILENAME statement for samples that use ODS to create output. If the _OUTPUT_TYPE and _ODS* parameters are not defined, the %JESBEGIN macro configures the job to create HTML output using ODS.

The %JESEND macro executes after the last line of code and closes all open ODS destinations.

Note: The %JESBEGIN and %JESEND macros are included by default and do not usually need to be added. If they do need to be manually added, then _ADDJESBEGINENDMACROS must be specified with a value of false.

Cascading Style Sheet Code in HTML Input Forms

The following code that appears at the beginning of the HTML input form retrieves the Cascading Style Sheet (CSS) for the current theme and controls the appearance of the input form:

```html
<link rel="stylesheet" href="/SASJobExecution/theme">
```

You can omit this LINK element if you do not want to use the theme in your HTML input form.

HTML Attributes to Support Accessibility

The HTML in the job forms that are included with the samples includes the following attributes, which support US government Section 508 accessibility standards:

```html
<html lang="en">
<body role="main">
```
Prompts
Most of the sample jobs include prompts. You can use _ACTION=prompts to display the prompts for a sample. The “Display Macro Variables” and “Dynamic Prompts Using CARS Data” sample sections contain the XML code for the prompts. See SAS Studio Developer’s Guide: Working with Jobs for more information about the SAS prompting interface and how to create prompts using XML.

Note: There are no prompts for the Upload a CSV File or Upload a File samples.

Display Macro Variables
This example illustrates how fields from an HTML input form are converted to global macro variables. The global and system macro variables are displayed in HTML format.

Output
The following sample output contains selected values, assuming that default selections were made in the HTML input form or prompts.
Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,prompts,execute</td>
<td>Displays the HTML input form or prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>ods_html5</td>
<td>Specifies that ODS HTML5 output is created by the job</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
<td>Specifies the name of the ODS style</td>
</tr>
</tbody>
</table>
HTML Input Form

**SAS® Job Execution**

**Display Macro Variables**

The program invoked by this HTML page is used to display the macro variables created for a job. The SAS Job Execution Web Application creates multiple macro variables when the same name is used for multiple fields in the HTML form.

The following three text fields are all named TEXT.

<table>
<thead>
<tr>
<th>1</th>
<th>First</th>
<th>Second</th>
<th>Third</th>
</tr>
</thead>
</table>

The next three checkboxes are all named CBOX. The value shown is the value specified in the INPUT tag.

- First
- Second
- Third

Now we have a selection box, LBOX, that allows multiple selections.

<table>
<thead>
<tr>
<th>3</th>
<th>First</th>
<th>Second</th>
<th>Third</th>
<th>Fourth</th>
</tr>
</thead>
</table>

ODS style: HTMLBlue

- Exclude blank macro variables from the report.

Run code  Show SAS Log

<!DOCTYPE html>
<html lang="en">
<head>
<title>Display Macro Variables</title>
<link rel="stylesheet" href="/SASJobExecution/theme">
</head>
<body role="main" class="jobexec_body">
<form class="jobexec_form" action="/SASJobExecution/" target="_tab">
<input type="hidden" name="_program" value="$PROGRAM$" />
<input type="hidden" name="_action" value="execute" />
<input type="hidden" name="_output_type" value="ods_html5" />
<div class="jobexec_sample_header">SAS® Job Execution</div>
<h1 class="jobexec_sample_name">Display Macro Variables</h1>
<p>The program invoked by this HTML page is used to display the macro variables created for a job.
The SAS Job Execution Web Application creates multiple macro variables when the same name is used for multiple fields in the HTML input form.

The following three text fields are all named TEXT.

```
1. <input type="text" name="text" value="First" class="jobexec_sample_input_text" aria-label="First text field"/>
2. <input type="text" name="text" value="Second" class="jobexec_sample_input_text" aria-label="Second text field"/>
3. <input type="text" name="text" value="Third" class="jobexec_sample_input_text" aria-label="Third text field"/>
```

The next three checkboxes are all named CBOX. The value shown is the value specified in the INPUT tag.

```
1. <input type="checkbox" name="cbox" id="cbox1" value="First" checked class="jobexec_sample_input_checkbox" aria-labelledby="checkboxfields cbox1"/>
2. <label for="cbox1">First</label>
3. <input type="checkbox" name="cbox" id="cbox2" value="Second" class="jobexec_sample_input_checkbox" aria-labelledby="checkboxfields cbox2"/>
4. <label for="cbox2">Second</label>
5. <input type="checkbox" name="cbox" id="cbox3" value="Third" checked class="jobexec_sample_input_checkbox" aria-labelledby="checkboxfields cbox3"/>
6. <label for="cbox3">Third</label>
```

Now we have a selection box, LBOX, that allows multiple selections.

```
1. <select name="lbox" multiple class="jobexec_sample_multiselect" aria-label="Selection box named LBOX">
2.   <option value="First">First</option>
3.   <option value="Second" selected>Second</option>
4.   <option value="Third">Third</option>
5.   <option value="Fourth" selected>Fourth</option>
6. </select>
```

```
1. <label for="_odsstyle">ODS style:</label>
2. <select name="_odsstyle" id="_odsstyle" class="jobexec_sample_select">
3.   [More values here]
4.   <option value="HTMLBlue" selected>HTMLBlue</option>
5.   [More values here]
6. </select>
```

```
This element, along with elements 2 and 3, enables you to specify multiple values. The creation of multiple value global macro variables and their usage is discussed in "Using Input Parameters with Multiple Values" on page 35. The output shows the values for the macro variables created from these elements when the default values are selected and the form is submitted.

This element enables you to specify multiple values.

This element enables you to specify multiple values.

The final SELECT tag creates a drop-down list that enables you to select the ODS style to apply to your output. The HTMLBlue style is selected by default. The selected value is stored in the _ODSTYLE global macro variable.

All macro variables are displayed in the output by default. Select the BLANKS check box to exclude macro variables with blank values.

Specify $PROGRAM$ as the value for the _PROGRAM input tag. When the HTML input form is displayed, the path and name of the program to execute are substituted for $PROGRAM$.

View the output to see the macro variables that are created when the form is submitted.

The HTML in this form has some of the same elements that are used in "ODS Output with Embedded Graphics" on page 77. See that section for more information.
Prompts

SAS® Job Execution

Display Macro Variables

The program invoked by this prompt page is used to display the macro variables created for a job. The SAS Job Execution Web Application creates multiple macro variables when the same name is used for multiple fields in the prompt page.

The following three text fields are all named TEXT:
First  Second  Third

This is dual selector named DUAL.
Select Edit to select values
First
Third

Now we have a selection box, LBOX, that allows multiple selections.

First
Second
Third
Fourth

QoS style: HTMLBlue

Exclude blank macro variables from the report:
No

Show SAS Log:
No

<?xml version="1.0" encoding="utf-8"?>
<Task schemaVersion="7.2">
  <Registration>
    <Name>Display Macro Variables Prompts</Name>
    <Description></Description>
    <Version>5.2</Version>
  </Registration>
  <Metadata>
    <DataSources/>
    <Options>
      <Option name="_program" inputType="inputtext" active="true" defaultValue=""/>
      <Option name="_action" inputType="inputtext" active="true" defaultValue="execute"/>
    </Options>
  </Metadata>
</Task>
The program invoked by this prompt page is used to display the macro variables created for a job. The SAS Job Execution Web Application creates multiple macro variables when the same name is used for multiple fields in the prompt page.

The following three text fields are all named TEXT.

- **First**
- **Second**
- **Third**

This is dual selector named DUAL. Select Edit to select values.

Now we have a selection box, LBOX, that allows multiple selections.

ODS style:
- **HTMLBlue**
- [More values here]
- [More values here]

Exclude blank macro variables from the report:
<Option inputType="string" name="blanks_2" returnValue="ExcludeBlanks">Yes</Option>

<Option name="_debug_label" inputType="string" Show SAS Log:</Option>

6<Option name="_debug" inputType="combobox" defaultValue="_debug_1"></Option>

<Option inputType="string" name="_debug_1" returnValue="">No</Option>

<Option inputType="string" name="_debug_2" returnValue="log">Yes</Option>

</Options>

</Metadata>

<UI>

<OptionItem option="header"/>

<OptionItem option="name"/>

<OptionItem option="desc1"/>

<OptionItem option="hr"/>

<OptionItem option="desc2"/>

<HorizontalLayout>

<OptionItem option="text_1"/>

<OptionItem option="text_2"/>

<OptionItem option="text_3"/>

</HorizontalLayout>

<OptionItem option="blankline"/>

<OptionChoice option="dual">

<OptionItem option="dual_1"/>

<OptionItem option="dual_2"/>

<OptionItem option="dual_3"/>

</OptionChoice>

<OptionItem option="blankline"/>

<OptionItem option="desc4"/>

<OptionChoice option="lbox">

<OptionItem option="lbox_1"/>

<OptionItem option="lbox_2"/>

<OptionItem option="lbox_3"/>

<OptionItem option="lbox_4"/>

</OptionChoice>

<OptionItem option="blankline"/>

<HorizontalLayout>

<OptionItem option="_odsstyle_label"/>

<OptionChoice option="_odsstyle">

[More values here]

</OptionChoice>

</HorizontalLayout>

<OptionItem option="blankline"/>

<HorizontalLayout>

<OptionItem option="blanks_label"/>

<OptionChoice option="blanks">

<OptionItem option="blanks_1"/>

<OptionItem option="blanks_2"/>

</OptionChoice>

</HorizontalLayout>

</OptionChoice>
The path and name of the program to execute are substituted at run time by the SAS Job Execution Web Application. The value for _PROGRAM is not needed in the XML because the value for the _PROGRAM URL parameter is used.

The blankline and hr objects are used to add vertical white space and a horizontal line to the prompt form, respectively.

These elements enable you to specify multiple values.

This element creates a drop-down list that enables you to select the ODS style to apply to your output. The HTMLBlue style is selected by default. The selected value is stored in the _ODSTYLE global macro variable.

All macro variables are displayed in the output by default. This element enables you to specify whether to exclude macro variables with blank values.

The _debug object creates a drop-down list with two choices. Selecting Yes returns a value of log in the _DEBUG macro variable.

The UI section displays the objects that were previously defined.

See SAS Studio Developer's Guide: Working with Jobs for more information about the SAS prompting interface and how to create prompts using XML.

Program

* Declare input parameter;

%global BLANKS;

%macro blanks;
   $if %length(&BLANKS) ne 0 $then and compress(value) ne ' ';
%mend;

*;
* Get the macro variables in the current SAS session.
* The RESOLVE function insures that the unmasked macro
* variable values are obtained.
*;

proc sort data=sashelp.vmacro out=work.vmacro sortseq=ebcdic;
   by scope name offset;
run; quit;

data work.globalvars
   work.systemvars;
set work.vmacro;
where (name ne 'TCPLISTN' %BLANKS) and
   (name not like '_RR_') and
   (name not like 'SQL%') and
   (name not like 'SYS_SQL%');
by scope name;
length full_value $32767;
retain full_value;
keep name full_value;
if (first.name)
   then full_value=value;
else full_value=cats(full_value, value);
if last.name then do;
   full_value = htmlencode(resolve(full_value));
   if (scope eq 'GLOBAL') then output work.globalvars;
   else if (scope eq 'AUTOMATIC') then output work.systemvars;
end;
label name = 'Macro Variable Name'
   full_value = 'Macro Variable Value';
run;

* Specify PROTECTSPECIALCHARS=off to prevent "double encoding" the values;

proc print data=work.globalvars label noobs
   style(header)=[just = center]
   style(column)=[protectspecialchar=off]
run;
quit;

proc print data=work.systemvars label noobs
   style(header)=[just = center]
   style(column)=[protectspecialchar=off]
run; quit;

Program Description
The %JESBEGIN macro performs several tasks before executing the code. A FILENAME statement
is issued to return HTML output to the web browser because ods html5 is specified as the value for
the _OUTPUT_TYPE input parameter. An ODS statement for the HTML5 destination is also issued.
The HTML is written to the _WEBOUT FILEREF assigned by the %JESBEGIN macro, and the style
specified in the _ODSSTYLE input parameter controls the appearance of the output.
The BLANKS macro generates code to omit blank macro variables from the output, if this option was
selected in the input form.
Macro variable information spans multiple records in the SASHELP.VMACRO view when the value exceeds 200 characters. The DATA step code collects information from multiple records and then stores the first 32,767 characters of the value.

The HTMLENCODE function is used when resolving the full value to prevent execution of malicious code in the web browser.

The PRINT procedure displays the global and system macro variable names and values.

The %JESEND macro executes after the last line of code and closes all open ODS destinations.

---

**Dynamic Prompts Using CARS Data**

This sample illustrates how to create SAS Viya dynamic and cascading prompts. *Dynamic prompts* allow the lookup of possible prompt values from a data source such as a SAS data set. *Cascading prompts* populate prompt values based on selections in other prompts. In this sample, prompts are sourced dynamically from the SASHELP.CARS data set and provide a user interface to the program. After you select a value for the vehicle make, you can select a value for the vehicle type. The values that are available for vehicle type depend on the value that was selected for vehicle make. An HTML input form is not available for this sample.

### Output

<table>
<thead>
<tr>
<th>Prompt Selections</th>
<th>Model</th>
<th>Engine Size (L)</th>
<th>Horsepower</th>
<th>MSRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make</td>
<td>Type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saab</td>
<td>Sedan</td>
<td>9-3 Aero 4dr</td>
<td>2.0</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9-3 Aero convertible 2dr</td>
<td>2.0</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9-3 Arc Sport 4dr</td>
<td>2.0</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9-3 Arc convertible 2dr</td>
<td>2.0</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9-5 Aero 4dr</td>
<td>2.3</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9-5 Arc 4dr</td>
<td>2.3</td>
<td>220</td>
</tr>
</tbody>
</table>

### Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>prompts,execute</td>
<td>Displays the prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>ods_html5</td>
<td>Specifies that ODS HTML5 output is created by the job</td>
</tr>
</tbody>
</table>
### Prompts

**SAS® Job Execution**

**Dynamic Prompts Using CARS Data**

This sample uses data in the SASHELP.CARS data set to source dynamic prompts.

#### Data set:

SASHELP.CARS

Select a vehicle make: 

![Saab dropdown](Saab)

Select a vehicle type: 

![Sedan dropdown](Sedan)

Show SAS Log:  

No

```xml
<?xml version="1.0" encoding="utf-8"?>
<Task schemaVersion="7.2">
  <Registration>
    <Name>Dynamic Prompts Using CARS Data Prompts</Name>
    <Description></Description>
    <Version>5.2</Version>
  </Registration>
  <Metadata>
    <DataSources>
      <DataSource name="dataset" defaultValue="sashelp.cars" where="false" required="true" readOnly="true">
        <Filters>
          <Filter name="filterMake">
            <Column column="make" sortBy="label" sortDirection="ascending"/>
          </Filter>
          <Filter name="filterType">
            <Column column="type" sortBy="label" sortDirection="ascending"/>
          </Filter>
        </Expressions>
      </DataSource>
    </DataSources>
  </Metadata>
</Task>
```
<Expression operation="eq">
  <Column column="make" sortBy="label"/>
  <Value option="make"/>
</Expression>
</Expressions>
</Where>
</Filter>
</Filters>
</DataSource>
</DataSources>

<Options>
  <Option name="_program" inputType="inputtext" active="true" defaultValue="/">
  <Option name="_action" inputType="inputtext" active="true" defaultValue="wait"/>

  <Option name="blankline" inputType="string"/>

  <Option name="header" inputType="markdown">&lt;h1&gt;SAS® Job Execution&lt;/h1&gt;
  <Option name="name" inputType="markdown">&lt;b&gt;Dynamic Prompts Using CARS Data&lt;/b&gt;
  <Option name="desc" inputType="string" readOnly="true">This sample uses data in the SASHELP.CARS data set to source dynamic prompts.
  <Option name="hr" inputType="markdown">&lt;hr&gt;
  <Option name="dataset_label" inputType="string">Data set:</Option>

  <Option name="make" inputType="combobox" filter="filterMake" defaultValue="Saab" required="true">Select a vehicle make:</Option>
  <Option name="type" inputType="combobox" filter="filterType" defaultValue="Sedan" required="true">Select a vehicle type:</Option>

  <Option name="_debug_label" inputType="string">Show SAS Log:</Option>

  <Option name="_debug" inputType="combobox" defaultValue="_debug_1">No</Option>
  <Option inputType="string" name="_debug_1" returnValue=""/>Yes</Option>
</Options>
</Metadata>

<UI>
  <OptionItem option="header"/>
  <OptionItem option="name"/>
  <OptionItem option="desc"/>

  <OptionItem option="hr"/>

  <OptionItem option="dataset_label"/>
  <DataItem data="dataset"/>
  <OptionItem option="blankline"/>

  <OptionChoice option="make"/>
  <OptionItem option="blankline"/>

  <OptionChoice option="type"/>
The `DataSource` element specifies the SAS data set to use as a data source for the prompts. You cannot select a value at run time because the `readOnly` attribute is set to `true`.

The first `Filter` element specifies that the values of the `MAKE` variable in `SASHELP.CARS` are displayed.

The second `Filter` element specifies that the values of the `TYPE` variable in `SASHELP.CARS` are displayed. Only values matching the selected value of `MAKE` are displayed because a `WHERE` expression is specified.

The path and name of the program to execute are substituted at run time by the SAS Job Execution Web Application. The value for `_PROGRAM` is not needed in the XML because the value for the `_PROGRAM` URL parameter is used.

The `blankline` and `hr` objects are used to add vertical white space and a horizontal line to the prompt form, respectively.

The `make` object creates a drop-down list populated with the values of the `MAKE` variable. The data displayed is determined by the `filter` attribute.

The `type` object creates a drop-down list populated with the values of the `TYPE` variable, but only for observations that match the selected value of `MAKE`.

The `_debug` object creates a drop-down list with two choices. Selecting `Yes` returns a value of `log` in the `_DEBUG` macro variable.

The UI section displays the objects that were previously defined.

See *SAS Studio Developer’s Guide: Working with Jobs* for more information about the SAS prompting interface and how to create prompts using XML.

When the prompts are submitted, the following global macro variables are defined just before SAS code execution, but only the `MAKE` and `TYPE` variables are used by the program:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>_ACTION</code></td>
<td><code>wait</code></td>
</tr>
<tr>
<td><code>_OUTPUT_TYPE</code></td>
<td><code>ods_html5</code></td>
</tr>
<tr>
<td><code>_PROGRAM</code></td>
<td><code>//Folder/Dynamic Prompts Using CARS Data</code></td>
</tr>
<tr>
<td><code>MAKE</code></td>
<td><code>Saab</code></td>
</tr>
<tr>
<td><code>TYPE</code></td>
<td><code>Sedan</code></td>
</tr>
</tbody>
</table>
Macro Variable Name | Macro Variable Value
--- | ---
_DEBUG | log (if Yes is selected in the drop-down list)

Program

* Declare input parameters;

```plaintext
%global MAKE
    TYPE;
```

* Subset the data based on prompt selections;

```plaintext
proc sql;
    create table work.cars as
        select make, type, model, enginesize, horsepower, msrp
        from sashelp.cars
        where (make eq "&MAKE") and
            (type eq "&TYPE")
        order by enginesize, model, msrp;
quit;
```

```plaintext
title 'Data Based on Prompt Selections';
```

```
proc report data=work.cars;
    column ('Prompt Selections' make type) model enginesize horsepower msrp;
    define make--type / order;
    define enginesize / format=4.1;
run; quit;
```

Program Description

The `%JESBEGIN` macro performs several tasks before executing the code.

A FILENAME statement is issued to return HTML output to the web browser because `ods_html5` was specified in the `_OUTPUT_TYPE` input parameter. An ODS statement for the HTML5 destination is also issued. If a blank value is specified for the `_OUTPUT_TYPE` input parameter, then a FILENAME statement for HTML output is issued, and the ODS HTML5 destination is used.

ODS writes the output to the _WEBOUT FILEREF assigned by the `%JESBEGIN` macro, and uses the default style for the HTML5 destination because the `_ODSSTYLE` input parameter was not specified.

The SQL procedure selects data from the SASHELP.CARS data set based on the user selections in the prompt form, and the REPORT procedure displays this data.

The `%JESEND` macro executes after the last line of code and closes all open ODS destinations.

Note: The `%JESBEGIN` and `%JESEND` macros are included by default and do not usually need to be added.
Hello World

This example creates HTML output using DATA step code and returns it to the web browser. An HTML input form, which accepts a name for the greeting, provides a basic user interface to the program.

Output

Hello World!

Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,prompts,execute</td>
<td>Displays the HTML input form or prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>html</td>
<td>Specifies that non-ODS HTML output is created by the job</td>
</tr>
<tr>
<td>myname</td>
<td>World</td>
<td>Specifies the default value used in the greeting</td>
</tr>
</tbody>
</table>

HTML Input Form

1. **SAS® Job Execution**
2. Hello World

   This sample uses a DATA Step with simple PUT statements to create the output.

3. Specify a name for the greeting: World

4. [Run code] [Show SAS Log]

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Hello World</title>
    <link rel="stylesheet" href="/SASJobExecution/theme">
</head>
<body>
    <!-- HTML content here -->
</body>
</html>
```
The DIV tag displays a page heading.

The name of the sample is displayed in the H1 tag.

The code prompts the user to specify a name used in the greeting. This field is required, and the default value for the name is World.

The Run Code button submits the job for execution. Select the Show SAS Log check box if you want to view the SAS log with the output.

Always specify /SASJobExecution/ for the value of the ACTION attribute of the FORM tag. This ensures that the SAS Job Execution Web Application processes the form data when the form is submitted. You can specify _tab for the value of the TARGET attribute of the FORM tag if you want output to display in a new tab within the application.

As the name implies, HIDDEN elements are not displayed on screen, but they pass data to the application specified in the ACTION attribute. Specify $PROGRAM$ as the value for the _PROGRAM input tag. When the HTML input form is displayed, the path and name of the program to execute are substituted for $PROGRAM$. A Please wait message is displayed while the program is running when the value for _ACTION is set to wait,execute. Specify only execute if you do not want to see this
message. The value of _OUTPUT_TYPE indicates that non-ODS HTML output is created by the job. The %JESBEGIN macro issues a FILENAME statement that supports HTML output.

When the form is submitted, the following global macro variables are defined just before SAS code execution:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ODSDEST</td>
<td>blank</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>html</td>
</tr>
<tr>
<td>_PROGRAM</td>
<td>/Folder/Hello World</td>
</tr>
<tr>
<td>myname</td>
<td>World</td>
</tr>
<tr>
<td>_DEBUG</td>
<td>log (if the check box is selected)</td>
</tr>
</tbody>
</table>

The _ODSDEST macro variable is derived from the value of _OUTPUT_TYPE. It is blank because ODS is not being used. The MYNAME macro variable is used in the program code.

Program

* Declare input parameter;

   $global MYNAME;

   * Write the custom HTML to _webout;

   data _null_;
   file _webout;
   put '<!DOCTYPE html>';
   put '<html lang=en'>;
   put '<head><title>Hello World!</title></head>';
   put '<body role=main>';
   put '<h1>Hello %sysfunc(htmlencode(&MYNAME))!</h1>';
   put '</body>';
   put '</html>';
   run;

Program Description

The %JESBEGIN macro assigns a FILENAME statement to return HTML output to the web browser because html is specified as the value for the _OUTPUT_TYPE input parameter.

The DATA step code writes simple HTML to the _WEBOUT FILEREFS that is assigned by the %JESBEGIN macro, and that HTML is rendered and displayed by the web browser. The value of the MYNAME macro variable used in the PUT statement is obtained from the value that was specified in the HTML input form. The HTMLENCODE function is used here to prevent execution of malicious code in the web browser.

The %JESEND macro executes after the last line of code, but it does not close all open ODS destinations because this sample does not use ODS.
Multiple Output Formats

This example uses ODS to create different output formats. An HTML input form, which accepts the
data set to display and the ODS destination and style, provides a basic user interface to the program.

Output

The following HTML output is one representation of the variety of formats that ODS can produce.

<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>69.0</td>
<td>112.5</td>
</tr>
<tr>
<td>Alice</td>
<td>F</td>
<td>13</td>
<td>66.5</td>
<td>64.0</td>
</tr>
<tr>
<td>Barbara</td>
<td>F</td>
<td>13</td>
<td>69.3</td>
<td>98.0</td>
</tr>
<tr>
<td>Carol</td>
<td>F</td>
<td>14</td>
<td>62.0</td>
<td>102.5</td>
</tr>
<tr>
<td>Henry</td>
<td>M</td>
<td>14</td>
<td>63.5</td>
<td>102.5</td>
</tr>
<tr>
<td>James</td>
<td>M</td>
<td>12</td>
<td>57.3</td>
<td>83.0</td>
</tr>
<tr>
<td>Jane</td>
<td>F</td>
<td>12</td>
<td>59.8</td>
<td>84.5</td>
</tr>
<tr>
<td>Janet</td>
<td>F</td>
<td>15</td>
<td>62.5</td>
<td>112.5</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>M</td>
<td>13</td>
<td>62.5</td>
<td>84.0</td>
</tr>
<tr>
<td>John</td>
<td>M</td>
<td>12</td>
<td>59.0</td>
<td>99.5</td>
</tr>
<tr>
<td>Joyce</td>
<td>F</td>
<td>11</td>
<td>51.3</td>
<td>50.5</td>
</tr>
<tr>
<td>Judy</td>
<td>F</td>
<td>14</td>
<td>64.3</td>
<td>90.0</td>
</tr>
<tr>
<td>Louise</td>
<td>F</td>
<td>12</td>
<td>56.3</td>
<td>77.0</td>
</tr>
<tr>
<td>Mary</td>
<td>F</td>
<td>15</td>
<td>66.5</td>
<td>112.0</td>
</tr>
<tr>
<td>Philip</td>
<td>M</td>
<td>16</td>
<td>72.0</td>
<td>150.0</td>
</tr>
<tr>
<td>Robert</td>
<td>M</td>
<td>12</td>
<td>64.8</td>
<td>128.0</td>
</tr>
<tr>
<td>Ronald</td>
<td>M</td>
<td>15</td>
<td>67.0</td>
<td>133.0</td>
</tr>
<tr>
<td>Thomas</td>
<td>M</td>
<td>11</td>
<td>57.5</td>
<td>85.0</td>
</tr>
<tr>
<td>William</td>
<td>M</td>
<td>15</td>
<td>66.5</td>
<td>112.0</td>
</tr>
</tbody>
</table>

N = 19
Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,prompts,execute</td>
<td>Displays the HTML input form or prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td>DATASET</td>
<td>SASHELP.CLASS</td>
<td>Specifies the SAS table to display</td>
</tr>
</tbody>
</table>

HTML Input Form

```html
<!DOCTYPE html>
<html lang="en">
<head>
<title>Multiple Output Formats</title>
<link rel="stylesheet" href="/SASJobExecution/theme">
</head>
<body role="main" class="jobexec_body">
<form class="jobexec_form" action="/SASJobExecution/" target="_tab">
<input type="hidden" name="_program" value="\$PROGRAM\$"/>
<input type="hidden" name="_action" value="execute"/>
<div class="jobexec_sample_header">SAS® Job Execution</div>
<h1 class="jobexec_sample_name">Multiple Output Formats</h1>
<p>This sample shows different output formats supported by the Output Delivery System (ODS). Sample data sets can be printed to any of the output formats listed below.</p>
</form>
</body>
</html>
```
1 <label for="dataset">Data set:</label>
<select name="dataset" id="dataset" class="jobexec_sample_select">
   <option value="" selected>&#160;</option>
   <option value="sashelp.retail">SASHELP.RETAIL</option>
   <option value="sashelp.class">SASHELP.CLASS</option>
   <option value="sashelp.revhub2">SASHELP.REVHUB2</option>
   <option value="does_not_exist">does_not_exist</option>
</select>
<br/>
<br/>
<br/>
2 <label for="_output_type">Output format:</label>
<select name="_output_type" id="_output_type" class="jobexec_sample_select">
   <option value="" selected>(default)</option>
   <option value="ods_html">HTML</option>
   <option value="ods_html5">HTML5</option>
   <option value="ods_pdf">Portable Document Format (PDF)</option>
   <option value="ods_rtf">Rich Text Format (RTF)</option>
   <option value="ods_csv">Comma-separated Value (CSV)</option>
   <option value="ods_xml">Extensible Markup Language (XML)</option>
   <option value="ods_tagsets.rtf">RTF Tagset</option>
   <option value="ods_tagsets.excelxp">Excel (XML)</option>
   <option value="ods_ps">Postscript (PS)</option>
   <option value="ods_latex">LaTeX</option>
</select>
<br/>
<br/>
<br/>
3 <label for="_odsstyle">ODS style:</label>
<select name="_odsstyle" id="_odsstyle" class="jobexec_sample_select">
   <option value="" selected>(default)</option>
   <!-- More values here -->
</select>
<br/>
<br/>
<br/>
<br/>
<br/>
<input type="submit" value="Run code" class="jobexec_sample_input_submit"/>
<input type="checkbox" name="_debug" id="_debug" value="log" class="jobexec_sample_input_checkbox" />
<label for="_debug">Show SAS Log</label>
</form>
</body>
</html>
1 The first SELECT tag creates a drop-down list with the names of the data set to be displayed, and the first item is selected by default. A macro variable named DATASET is created with the corresponding value in the VALUE attribute of the OPTION tag when you select an item. The first and last items are used to test the cases when no data set is specified and when a non-existent data set is selected, respectively. The SAS program handles these two error conditions.

2 The second SELECT tag creates a drop-down list that enables you to select the ODS output format. The first item is selected by default and results in a blank value for the _OUTPUT_TYPE macro variable. The JESBEGIN macro handles this case, and the ODS HTML5 destination creates the output.

3 The final SELECT tag creates a drop-down list that enables you to choose the ODS style to apply to your output. The first item is selected by default and results in a blank value for the _ODSSTYLE macro variable. The JESBEGIN macro handles this case, and the output is created using the HTMLBlue ODS style.

The following table contains the display values for the first drop-down list and the values for the DATASET global macro variable:

<table>
<thead>
<tr>
<th>Display Value</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>blank</td>
<td>blank</td>
</tr>
<tr>
<td>SASHELP.RETAIL</td>
<td>sashelp.retail</td>
</tr>
<tr>
<td>SASHELP.CLASS</td>
<td>sashelp.class</td>
</tr>
<tr>
<td>SASHELP.REVHUB2</td>
<td>sashelp.revhub2</td>
</tr>
<tr>
<td>does_not_exist</td>
<td>does_not_exist</td>
</tr>
</tbody>
</table>

The following table contains the display values for the first drop-down list and the values for the _OUTPUT_TYPE global macro variable:

<table>
<thead>
<tr>
<th>Display Value</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(default)</td>
<td>ods_html</td>
</tr>
<tr>
<td>HTML</td>
<td>ods_html</td>
</tr>
<tr>
<td>HTML5</td>
<td>ods_html5</td>
</tr>
<tr>
<td>Portable Document Format (PDF)</td>
<td>ods_pdf</td>
</tr>
<tr>
<td>Rich Text Format (RTF)</td>
<td>ods_rtf</td>
</tr>
<tr>
<td>Comma-Separated Value (CSV)</td>
<td>ods_csv</td>
</tr>
<tr>
<td>Extensible Markup Language (XML)</td>
<td>ods_xml</td>
</tr>
<tr>
<td>RTF_Tagset</td>
<td>ods_tagsets.rtf</td>
</tr>
<tr>
<td>Excel (XML)</td>
<td>ods_tagsets.excelxp</td>
</tr>
</tbody>
</table>
When the form is submitted, the following global macro variables are defined just before SAS code execution, but only the _ODSDEST and DATASET variables are used by the program:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>execute</td>
</tr>
<tr>
<td>_ODSDEST</td>
<td>Depends on selection (for example, tagsets.rtf)</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>Depends on selection (for example, HTMLBlue)</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>Depends on selection (see previous table)</td>
</tr>
<tr>
<td>_PROGRAM</td>
<td>//Folder/Multiple Output Formats</td>
</tr>
<tr>
<td>_DEBUG</td>
<td>log (if the check box is selected)</td>
</tr>
<tr>
<td>DATASET</td>
<td>Depends on selection (see previous table)</td>
</tr>
</tbody>
</table>

The _ODSDEST macro variable is derived from the value of _OUTPUT_TYPE and indicates the ODS destination that is used.

The HTML in this form uses some of the same fields as “Simple ODS HTML” on page 99. See that section for more information.

Program

* Declare input parameter;

%global DATASET;

%macro setup;

%local ERRORTEXT RC;

%* Verify that a valid data set was specified;

%if (%qcmpres(&DATASET) eq )
  %then %let ERRORTEXT=ERROR: You must specify a data set.;
%else %if not %sysfunc(exist(&DATASET))
  %then %let ERRORTEXT=ERROR: Data set "%sysfunc(htmlencode(&DATASET))" not found.;

%if (%bquote(&ERRORTEXT) ne ) %then %do;
  %* Close the currently open destination and write message to the browser;
ods _all_ close;

%let RC=%sysfunc(fdelete(_webout));

filename _webout filesrvc parenturi="&SYS_JES_JOB_URI"
   name='_webout.htm';

title;

ods html5 file=_webout
   text="&ERRORTEXT";
ods html5 close;

data _null_; abort cancel;
run;
%end;
%mend setup;

%setup

title "%sysfunc(htmlencode(%qupcase(&DATASET))) Table in
   %sysfunc(htmlencode(%qupcase(&_ODSDEST))) Format";

proc print data=&DATASET noobs label n; run; quit;

Program Description

The %JESBEGIN macro performs several tasks before executing the code. A FILENAME statement is issued to return the type of output that is specified in the _OUTPUT_TYPE input parameter to the web browser.

An ODS statement for the appropriate destination is also issued. If a blank value is specified for the _OUTPUT_TYPE input parameter, then a FILENAME statement for HTML output is issued, and the ODS HTML5 destination is used.

ODS writes the output to the _WEBOUT FILEREF assigned by the %JESBEGIN macro, and the style specified in the _ODSSYLE input parameter controls the appearance of the output.

The SETUP macro checks the validity of the value specified for the DATASSET input parameter. If no value is specified or if the specified table does not exist, then an error message is created and returned to the web browser using the HTML5 ODS destination. The HTMLENCODE function is used here, and later in the TITLE statement, to prevent execution of malicious code in the web browser.

All ODS destinations are closed, and the current content of the _WEBOUT FILEREF is deleted to ensure that only the error message is returned to the web browser. A FILENAME statement is issued to return HTML output to the web browser, and an ODS statement for the HTML5 destination writes the error message to the _WEBOUT FILEREF. See “Assigning a FILEREF for HTML Output” on page 42 for more information about this type of FILENAME statement. No additional program statements are executed after the ABORT statement executes.

The PRINT procedure displays the data in the specified table.

The %JESEND macro executes after the last line of code and closes all open ODS destinations.

HTML and PDF output are displayed by the web browser, and all other output is downloaded so that it can be saved and opened with an appropriate application.
Note: The %JESBEGIN and %JESEND macros are included by default and do not usually need to be added.

ODS Output with Embedded Graphics

This example creates HTML, PDF, or RTF output using ODS and returns it to the web browser. An HTML input form, which accepts the ODS destination, style, and graphic output type, provides a basic user interface to the program.

Output

![Bar Chart](chart.png)
### Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,prompts,execute</td>
<td>Displays the HTML input form or prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>ods_html5</td>
<td>Specifies that ODS HTML5 output is created by the job</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
<td>Specifies the name of the ODS style</td>
</tr>
</tbody>
</table>

### Student Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>69.0</td>
<td>112.5</td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
<td>F</td>
<td>13</td>
<td>60.5</td>
<td>84.0</td>
</tr>
<tr>
<td>3</td>
<td>Barbara</td>
<td>F</td>
<td>13</td>
<td>65.3</td>
<td>98.0</td>
</tr>
<tr>
<td>4</td>
<td>Carol</td>
<td>F</td>
<td>14</td>
<td>62.8</td>
<td>102.5</td>
</tr>
<tr>
<td>5</td>
<td>Henry</td>
<td>M</td>
<td>14</td>
<td>63.5</td>
<td>102.5</td>
</tr>
<tr>
<td>6</td>
<td>James</td>
<td>M</td>
<td>12</td>
<td>67.0</td>
<td>83.0</td>
</tr>
<tr>
<td>7</td>
<td>Jane</td>
<td>F</td>
<td>12</td>
<td>59.8</td>
<td>84.6</td>
</tr>
<tr>
<td>8</td>
<td>Janet</td>
<td>F</td>
<td>16</td>
<td>62.5</td>
<td>112.5</td>
</tr>
<tr>
<td>9</td>
<td>Jeffrey</td>
<td>M</td>
<td>13</td>
<td>62.5</td>
<td>84.0</td>
</tr>
<tr>
<td>10</td>
<td>John</td>
<td>M</td>
<td>12</td>
<td>69.0</td>
<td>69.5</td>
</tr>
<tr>
<td>11</td>
<td>Joyce</td>
<td>F</td>
<td>11</td>
<td>61.3</td>
<td>50.0</td>
</tr>
<tr>
<td>12</td>
<td>Judy</td>
<td>F</td>
<td>14</td>
<td>64.3</td>
<td>90.0</td>
</tr>
<tr>
<td>13</td>
<td>Louise</td>
<td>F</td>
<td>12</td>
<td>59.3</td>
<td>77.0</td>
</tr>
<tr>
<td>14</td>
<td>Mary</td>
<td>F</td>
<td>15</td>
<td>69.5</td>
<td>112.0</td>
</tr>
<tr>
<td>15</td>
<td>Philip</td>
<td>M</td>
<td>16</td>
<td>72.0</td>
<td>160.0</td>
</tr>
<tr>
<td>16</td>
<td>Robert</td>
<td>M</td>
<td>12</td>
<td>64.8</td>
<td>128.0</td>
</tr>
<tr>
<td>17</td>
<td>Ronald</td>
<td>M</td>
<td>18</td>
<td>67.0</td>
<td>133.0</td>
</tr>
<tr>
<td>18</td>
<td>Thomas</td>
<td>M</td>
<td>11</td>
<td>57.5</td>
<td>85.0</td>
</tr>
<tr>
<td>18</td>
<td>William</td>
<td>M</td>
<td>15</td>
<td>65.6</td>
<td>112.0</td>
</tr>
</tbody>
</table>
The SGPLOT procedure creates a bar chart using the SASHELP.CLASS table, followed by a display of the data using the PRINT procedure.

The SVG image format is not supported for RTF output.

Output format: HTML5

Graphic image format: (default based on output format)

ODS style: HTMLBlue
The first SELECT tag creates a drop-down list with output format values. A macro variable named _OUTPUT_TYPE is created with the corresponding value in the VALUE attribute of the OPTION tag.

The second SELECT tag creates a drop-down list that enables you to select the ODS graphic image format. The first item is initially selected and results in a blank value for the _ODS_DEVICE macro variable.

The final SELECT tag creates a drop-down list that enables you to select the ODS style to apply to your output. The HTMLBlue style is selected by default. The selected value is stored in the _ODSSTYLE global macro variable.

Specify $PROGRAM$ as the value for the _PROGRAM input tag. When the HTML input form is displayed, the path and name of the program to execute are substituted for $PROGRAM$.

The following table lists the display values for the first drop-down list and the corresponding values for the _OUTPUT_TYPE global macro variable:

<table>
<thead>
<tr>
<th>Display Value</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML5</td>
<td>ods_html5</td>
</tr>
</tbody>
</table>
The following table lists the display values for the second drop-down list and the values for the _ODS_DEVICE global macro variable:

<table>
<thead>
<tr>
<th>Display Value</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(default based on the output format)</td>
<td>blank</td>
</tr>
<tr>
<td>SVG</td>
<td>svg</td>
</tr>
<tr>
<td>PNG</td>
<td>png</td>
</tr>
</tbody>
</table>

When the form is submitted, the following global macro variables are defined just before SAS code execution, but they are not used by the program:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>execute</td>
</tr>
<tr>
<td>_ODS_DEVICE</td>
<td>Depends on selection (blank, svg, or png)</td>
</tr>
<tr>
<td>_ODSDEST</td>
<td>Depends on selection (html5, pdf, or rtf)</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>Depends on selection (for example, HTMLBlue)</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>Depends on selection (ods_html5, ods_pdf, or ods_rtf)</td>
</tr>
<tr>
<td>_PROGRAM</td>
<td>/Folder/ODS Output with Embedded Graphics</td>
</tr>
<tr>
<td>_DEBUG</td>
<td>log (if the check box is selected)</td>
</tr>
</tbody>
</table>

The _ODSDEST macro variable is derived from the value of _OUTPUT_TYPE and indicates the ODS destination that is used.

The HTML in this form uses some of the same fields as “Simple ODS HTML” on page 99. See that section for more information.

Program

```
title 'Student Data - SGLOT';
proc sgplot data=sashelp.class; hbar age; run; quit;

  title 'Student Data';
  proc print  data=sashelp.class; run; quit;
```
Program Description

The %JESBEGIN macro performs several tasks before executing the code. A FILENAME statement is issued to return the type of output that is specified in the _OUTPUT_TYPE input parameter to the web browser.

An ODS statement using the HTML5, PDF, or RTF destination is issued based on the value of _OUTPUT_TYPE. ODS writes the output to the _WEBOUT FILEREF assigned by the %JESBEGIN macro, and the style specified in the _ODSSTYLE input parameter controls the appearance of the output.

An ODS GRAPHICS statement that specifies the graphic image format is issued if a value is specified for the _ODS_DEVICE input parameter. If no value is specified for _ODS_DEVICE, then the best format is used, based on the ODS destination.

The SGPLOT procedure creates the graphic image using data from the SASHELP.CLASS table, and the PRINT procedure displays the data.

The %JESEND macro executes after the last line of code and closes all open ODS destinations.

The HTML output and the PDF output are displayed by the web browser, and the RTF output is downloaded so that it can be saved and opened with an appropriate application, such as Microsoft Word.

Note: The %JESBEGIN and %JESEND macros are included by default and do not usually need to be added.

Report with Download Links

This example creates a report in HTML format and provides links to download the report in Excel Spreadsheet XML and PDF formats. An HTML input form provides a basic user interface to the program.
### Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,prompts,execute</td>
<td>Displays the HTML input form or prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>none</td>
<td>Suppresses automatic issuing of FILENAME and ODS statements</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
<td>Specifies the name of the ODS style</td>
</tr>
</tbody>
</table>

---

### The CLASS Table

**Gender=Female**

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Judy</td>
<td>14</td>
<td>84.3</td>
<td>60.0</td>
</tr>
<tr>
<td>Jane</td>
<td>12</td>
<td>52.9</td>
<td>54.5</td>
</tr>
<tr>
<td>Joyce</td>
<td>11</td>
<td>51.3</td>
<td>55.2</td>
</tr>
<tr>
<td>Barbara</td>
<td>13</td>
<td>55.3</td>
<td>88.0</td>
</tr>
<tr>
<td>Carol</td>
<td>14</td>
<td>52.6</td>
<td>102.5</td>
</tr>
<tr>
<td>Mary</td>
<td>15</td>
<td>50.0</td>
<td>112.0</td>
</tr>
<tr>
<td>Louise</td>
<td>12</td>
<td>59.3</td>
<td>77.0</td>
</tr>
<tr>
<td>Alice</td>
<td>13</td>
<td>56.5</td>
<td>84.0</td>
</tr>
<tr>
<td>Janet</td>
<td>15</td>
<td>52.5</td>
<td>112.6</td>
</tr>
</tbody>
</table>

N = 9

**Gender=Male**

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Philip</td>
<td>16</td>
<td>72.0</td>
<td>103.0</td>
</tr>
<tr>
<td>James</td>
<td>12</td>
<td>67.0</td>
<td>83.0</td>
</tr>
<tr>
<td>Henry</td>
<td>14</td>
<td>63.6</td>
<td>102.6</td>
</tr>
<tr>
<td>John</td>
<td>12</td>
<td>60.0</td>
<td>90.8</td>
</tr>
<tr>
<td>William</td>
<td>16</td>
<td>65.6</td>
<td>112.0</td>
</tr>
<tr>
<td>Alfred</td>
<td>14</td>
<td>60.0</td>
<td>112.6</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>13</td>
<td>62.5</td>
<td>84.0</td>
</tr>
<tr>
<td>Thomas</td>
<td>11</td>
<td>67.0</td>
<td>85.0</td>
</tr>
<tr>
<td>Ronald</td>
<td>15</td>
<td>67.0</td>
<td>133.0</td>
</tr>
<tr>
<td>Robert</td>
<td>12</td>
<td>64.6</td>
<td>129.0</td>
</tr>
</tbody>
</table>

N = 10
The PRINT procedure displays data in the SASHELP.CLASS table in HTML format. Links are provided to download the output in the Excel and PDF formats.

**ODS style:**

```html
<input type="hidden" name="_odsstyle" value="HTMLBlue" />
```

---

<input type="submit" value="Run code" class="jobexec_sample_input_submit"/>
<input type="checkbox" name="_debug" id="_debug" value="log" class="jobexec_sample_input_checkbox"/>
<label for="_debug">Show SAS Log</label>
The HTML in this form uses some of the same fields as “Simple ODS HTML” on page 99. See that section for more information.

Specify $\textit{PROGRAM}$ as the value for the \_\texttt{PROGRAM} input tag. When the HTML input form is displayed, the path and name of the program to execute are substituted for $\textit{PROGRAM}$.

The FILENAME and ODS statements issued by the %JESBEGIN macro do not meet the needs of this program. Specifying \texttt{none} for _\texttt{OUTPUT_TYPE} suppresses the issuing of these statements. The appropriate statements are issued within the program.

When the form is submitted, the following global macro variables are defined just before SAS code execution, but only _\texttt{ODSSTYLE} is used by the program:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_\texttt{ACTION}</td>
<td>\texttt{execute}</td>
</tr>
<tr>
<td>_\texttt{ODSDEST}</td>
<td>\texttt{blank}</td>
</tr>
<tr>
<td>_\texttt{ODSSTYLE}</td>
<td>Depends on selection (for example, HTMLBlue)</td>
</tr>
<tr>
<td>_\texttt{OUTPUT_TYPE}</td>
<td>\texttt{none}</td>
</tr>
<tr>
<td>_\texttt{PROGRAM}</td>
<td>/\texttt{Folder}/Report with Download Links</td>
</tr>
<tr>
<td>_\texttt{DEBUG}</td>
<td>\texttt{log} (if the check box is selected)</td>
</tr>
</tbody>
</table>

Program

* Close all open destinations;

\begin{verbatim}
ods _all_ close;

options nodate nonumber;

* Declare input parameter;

%global _\texttt{ODSSTYLE};

* Define the escape character for ODS inline formatting;

ods escapechar='^';

* Create a format for the student gender;

proc format;
  value $\texttt{gender}'F' = 'Female'
\end{verbatim}
'M' = 'Male';
run; quit;

* Prepare the data;

proc sql;
  create view work.class as
  select name label   = 'Name',
         sex  label   = 'Gender' format=$gender.,
         age  label   = 'Age',
         height label = 'Height',
         weight label = 'Weight'
  from sashelp.class
  order by sex;
quit;

* Create an ODS document with the report results;

  title1 'The CLASS Table';
  footnote;

  1ods document name=work.mydoc(write);

  proc print data=work.class noobs n label;
    by sex;
    var name age height weight;
  run; quit;

  ods document close;

* Create the Excel XML and PDF output and associate with the job;

  2filename f_xlxp filesrvc parenturi="&SYS_JES_JOB_URI"
     name='Class.xml'
     contenttype='application/vnd.ms-excel'
     contentdisp='attachment; filename="Class.xml"';

  3filename f_pdf filesrvc parenturi="&SYS_JES_JOB_URI"
     name='Class.pdf'
     contenttype='application/pdf';

  ods pdf file=f_pdf style=&_ODSSTYLE;
  ods tagsets.ExcelXP file=f_xlxp style=&_ODSSTYLE
    options(embedded_titles='yes'
             suppress_bylines='yes'
             sheet_name='#byval(sex) Students'
             print_header=&C&A);

  proc document name=work.mydoc;
    replay;
  run; quit;

  ods pdf close;
  ods tagsets.ExcelXP close;
* Create download links;

4 %let EXCEL_LINK=%bquote(<a href="&_FILESRVC_F_XLXP_URI/content" target="&_SASDLResults">Excel</a>);
%let PDF_LINK=%bquote(<a href="&_FILESRVC_F_PDF_URI/content" target="&_SASDLResults">PDF</a>);

* Create the HTML output for display in the Web browser;

5 filename f_htm filesrvc parenturi="&_SYS_JES_JOB_URI" name='_webout.htm';

6 ods html5 file=f_htm style=&_ODSSTYLE
   text="&nbsp;&nbsp;";
   proc document name=work.mydoc;
      replay;
   run; quit;
ods html5 close;

The ODS DOCUMENT statement stores the output components from the PRINT procedure in a document named MYDOC. You can later use PROC DOCUMENT to display the results using any ODS destination. This technique is useful when you need to display procedure output several times, but you do not want to rerun the procedure.

FILENAME and ODS statements are not issued by the %JESBEGIN macro because _OUTPUT_TYPE=none is specified. FILENAME statements are issued to store the Microsoft XML results and the PDF results, following the general format discussed in “Assigning a FILEREF for Other Types of Output” on page 43.

The next FILENAME statement stores PDF output generated by the ODS PDF destination. The CONTENTTYPE option creates a MIME header that informs the web browser that the content is intended for a client capable of rendering PDF output. Most web browsers can display PDF output, so the CONTENTDISP option is not needed.

As discussed in “Assigning a FILEREF for HTML Output” on page 42, the FILESRVC access method creates global macro variables of the form _FILESRVC_filerref_URI, where FILEREF is the fileref used in the FIENAME statement. This macro variable provides a relative URL that can be used to reference and retrieve the file using the Files service. These macro variables are used to create the EXCEL_LINK and PDF_LINK macro variables. Later they are used to create download links.

A FILENAME statement is issued to return HTML content to the web browser. See “Assigning a FILEREF for HTML Output” on page 42 for more information about this statement.

The ODS HTML5 destination creates the HTML output using the ODS style specified in the _ODSSTYLE input parameter. The macro variables created earlier provide download links to the Excel XML and the PDF versions of the report. The NBSPACE inline formatting function provides extra blank space between the link text in the output. See SAS Output Delivery System: User’s Guide for more information about the NBSPACE inline formatting function.

Program Description

The %JESBEGIN macro normally performs several tasks before executing your code. In this case, only a list of global macro variables is displayed because _OUTPUT_TYPE=none is specified.

See SAS Output Delivery System: User’s Guide for more information about ODS ESCAPECHAR.
The SQL procedure selects the data of interest, applies a user-defined format, and then sorts the data.

PROC DOCUMENT executes and replays the PROC PRINT output created earlier and stores it in the XML and PDF files. The ODS destinations are closed after PROC DOCUMENT creates the output files. See SAS Output Delivery System: Procedures Guide for more information about PROC DOCUMENT.

PROC DOCUMENT then executes and replays the PROC PRINT output created earlier, and the HTML output is displayed in the web browser. The %JESEND macro executes after the last line of code, but it does not close all open ODS destinations because _OUTPUT_TYPE=none is specified. The HTML5 destination must be explicitly closed.

---

Note: The %JESBEGIN and %JESEND macros are included by default and do not usually need to be added.

---

Return GCHART Image Output

This example uses the GCHART procedure to create a bar chart using the SASHELP.CLASS table, and returns the image data to the client. An HTML input form provides a basic user interface to the program, but a more appropriate use is including SAS-generated graphic output in another web application, such as a dashboard. In this case, you can specify the URL in an IMG tag: `<img src="http://host:port/SASJobExecution/?_program=/Folder/Return GCHART Image Output&_action=execute">`
### Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ACTION</em></td>
<td>form,prompts,execute</td>
<td>Displays the HTML input form or prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td><em>OUTPUT_TYPE</em></td>
<td>png</td>
<td>Specifies that binary PNG output is created by the job</td>
</tr>
<tr>
<td><em>ODSSTYLE</em></td>
<td>HTMLBlue</td>
<td>Specifies the name of the ODS style</td>
</tr>
</tbody>
</table>
HTML Input Form

SAS® Job Execution

Return GCHART Image Output

The GCHART procedure creates a bar chart using the SASHELP.CLASS table, and returns the binary image data to the client.

Output format: PNG

ODS style: HTMLBlue

Run code □ Show SAS Log

<!DOCTYPE html>
<html lang="en">
<head>
<title>Return GCHART Image Output</title>
<link rel="stylesheet" href="/SASJobExecution/theme">
</head>
<body role="main" class="jobexec_body">
<form class="jobexec_form" action="/SASJobExecution/" target="_tab">
<input type="hidden" name="_program" value="$PROGRAM$"/>
<input type="hidden" name="_action" value="execute"/>
<div class="jobexec_sample_header">SAS<sup>®</sup> Job Execution</div>
<h1 class="jobexec_sample_name">Return GCHART Image Output</h1>
<p>The GCHART procedure creates a bar chart using the SASHELP.CLASS table, and returns the binary image data to the client.</p>
<label for="_output_type">Output format:</label>
<select name="_output_type" id="_output_type" class="jobexec_sample_select">
<option value="gif">GIF</option>
<option value="jpeg">JPEG</option>
<option value="svg">SVG</option>
<option value="png" selected>PNG</option>
</select>
<br/>
<br/>
<label for="_odsstyle">ODS style:</label>
<select name="_odsstyle" id="_odsstyle" class="jobexec_sample_select">
[More values here]
<option value="HTMLBlue" selected>HTMLBlue</option>
</select>
<br/>
Specify $\text{PROGRAM}$ as the value for the _PROGRAM input tag. When the HTML input form is displayed, the path and name of the program to execute are substituted for $\text{PROGRAM}$.

2 The first SELECT tag creates a drop-down list that enables you to select the SAS/GRAPH graphic image format. PNG is selected by default. Specifying an image format as the value for _OUTPUT_TYPE creates image data.

3 The second SELECT tag creates a drop-down list that enables you to select the ODS style to apply to the graphic output. The HTMLBlue style is selected by default. The selected value is stored in the _ODSSTYLE global macro variable.

When the form is submitted, the following global macro variables are defined just before SAS code execution, but only _ODSSTYLE and _OUTPUT_TYPE are used by the program:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>execute</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>Depends on selection (for example, HTMLBlue)</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>Depends on selection (gif, jpeg, svg, or png)</td>
</tr>
<tr>
<td>_PROGRAM</td>
<td>/Folder/Return GCHART Image Output</td>
</tr>
<tr>
<td>_DEBUG</td>
<td>log (if the check box is selected)</td>
</tr>
</tbody>
</table>

The HTML in this form uses some of the same fields as “Simple ODS HTML” on page 99. See that section for more information.

Program

* Declare input parameters;

%global _ODSSTYLE
 _OUTPUT_TYPE;
Program Description

The %JESBEGIN macro issues a FILENAME statement to return the type of output that is specified in the _OUTPUT_TYPE input parameter.

The ODS style specified in the _ODSSTYLE parameter controls the graphic output appearance.

The ODS Graphics option to specify the output format is specified.

The GCHART procedure creates the graphic image using data from the SASHELP.CLASS table.

The %JESEND macro executes after the last line of code and closes all open ODS destinations. The image output is displayed by the web browser, or within an IMG tag.

Note: The %JESBEGIN and %JESEND macros are included by default and do not usually need to be added.

---

Return SGPLOT Image Output

This example uses the SGPLOT procedure to create a bar chart using the SASHELP.CLASS table, and returns the image data to the client. An HTML input form provides a basic user interface to the program, but a more appropriate use is including SAS-generated graphic output in another web application, such as a dashboard. In this case, you can specify the URL in an IMG tag: `<img src="http://host:port/SASJobExecution/?_program=/Folder/Return SGPLOT Image Output&_action=execute">"
Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,prompts,execute</td>
<td>Displays the HTML input form or prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>png</td>
<td>Specifies that binary PNG output is created by the job</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
<td>Specifies the name of the ODS style</td>
</tr>
</tbody>
</table>
The SGPLOT procedure creates a bar chart using the SASHELP.CLASS table, and returns the binary image data to the client.

Output format: PNG

ODS style: HTMLBlue

The SGPLOT procedure creates a bar chart using the SASHELP.CLASS table, and returns the binary image data to the client.
Specify $\textit{PROGRAM}$ as the value for the \_PROGRAM input tag. When the HTML input form is displayed, the path and name of the program to execute are substituted for $\textit{PROGRAM}$. 

The first SELECT tag creates a drop-down list that enables you to select the SAS/GRAPH graphic image format. PNG is selected by default. Specifying an image format as the value for \_OUTPUT_TYPE creates image data.

The second SELECT tag creates a drop-down list that enables you to select the ODS style to apply to the graphic output. The HTMLBlue style is selected by default. The selected value is stored in the \_ODSTYLE global macro variable.

When the form is submitted, the following global macro variables are defined just before SAS code execution, but only \_ODSTYLE and \_OUTPUT_TYPE are used by the program:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>execute</td>
</tr>
<tr>
<td>_ODSTYLE</td>
<td>Depends on selection (for example, HTMLBlue)</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>Depends on selection (gif, jpeg, svg, or png)</td>
</tr>
<tr>
<td>_PROGRAM</td>
<td>/Folder/Return SGPLOT Image Output</td>
</tr>
<tr>
<td>_DEBUG</td>
<td>log (if the check box is selected)</td>
</tr>
</tbody>
</table>

The HTML in this form uses some of the same fields as “Simple ODS HTML” on page 99. See that section for more information.

Program

* Declare input parameters;

\%
\global \_ODSTYLE
\global \_OUTPUT_TYPE;
* Specify the ODS style and graphic output format;

ods listing style=&_ODSSTYLE;
ods graphics / outputfmt=&_OUTPUT_TYPE;

* Create the graphic output;

proc sgplot data=sashelp.class; hbar age; run; quit;

Program Description

The %JESBEGIN macro issues a FILENAME statement to return the type of output that is specified in the _OUTPUT_TYPE input parameter.
The ODS style specified in the _ODSSTYLE parameter controls the graphic output appearance.
The ODS Graphics option to specify the output format is specified.
The SGPLOT procedure creates the graphic image using data from the SASHELP.CLASS table.
The %JESEND macro executes after the last line of code and closes all open ODS destinations. The image output is displayed by the web browser, or within an IMG tag.

Note: The %JESBEGIN and %JESEND macros are included by default and do not usually need to be added.

Simple JSON

This example uses PROC JSON to display the SASHELP.CLASS table in JSON format. Use this technique and execute the job using direct URL access if you have an application that requires data in JSON format. An HTML input form provides a basic user interface to the program.

Output

```
[  
   {  
      "Name": "Alfred",  
      "Sex": "M",  
      "Age": 14,  
      "Height": 69,  
      "Weight": 112.5  
   },  
   {  
      "Name": "Alice",  
      "Sex": "F",  
      "Age": 13,  
      "Height": 56.5,  
      "Weight": 84  
   },  
   {  
      "Name": "Barbara",  
      
```
"Sex": "F",
"Age": 13,
"Height": 65.3,
"Weight": 98
}

**Job Definition Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,prompts,execute</td>
<td>Displays the HTML input form or prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>json</td>
<td>Specifies that JSON (non-ODS) output is created by the job</td>
</tr>
</tbody>
</table>

**HTML Input Form**

---

**SAS® Job Execution**

Simple JSON

The JSON procedure creates simple JSON content that returns the data in the SASHELP.CLASS table.

---

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Simple JSON</title>
<link rel="stylesheet" href="/SASJobExecution/theme">
</head>

<body role="main" class="jobexec_body">

<form class="jobexec_form" action="/SASJobExecution/" target="_tab">
  <input type="hidden" name="_program" value="$PROGRAM$"/>
  <input type="hidden" name="_action" value="execute"/>
  <input type="hidden" name="_output_type" value="json"/>
  <div class="jobexec_sample_header">SAS<sup>&#174;</sup> Job Execution</div>
  <h1 class="jobexec_sample_name">Simple JSON</h1>
  <p>The JSON procedure creates simple JSON content that returns the data in the SASHELP.CLASS table.</p>
</form>
```

---

The JSON procedure creates simple JSON content that returns the data in the SASHELP.CLASS table.
The HTML in this form uses some of the same fields as “Simple ODS HTML” on page 99. See that section for more information.

Specify $\text{PROGRAM}$ as the value for the _PROGRAM input tag. When the HTML input form is displayed, the path and name of the program to execute are substituted for $\text{PROGRAM}$.

The value of _OUTPUT_TYPE indicates that non-ODS JSON output is created by the job. The %JESBEGIN macro issues a FILENAME statement that supports JSON output.

Program

```sas
proc json out=_webout nosastags pretty;
  export sashelp.class;
run; quit;
```

Program Description

The %JESBEGIN macro assigns a FILENAME statement to return JSON output to the web browser because json is specified as the value for the _OUTPUT_TYPE input parameter.

The JSON procedure converts the SASHELP.CLASS table to JSON format and then writes JSON data to the _WEBOUT FILEREF that is assigned by the %JESBEGIN macro. The JSON is displayed by the web browser.

The %JESEND macro executes after the last line of code, but it does not close all open ODS destinations because this sample does not use ODS.

This technique is useful if you have an application such as a JavaScript grid or chart object that requires data in JSON format. In your application, specify the following URL to retrieve the data in JSON format:

```
http://host:port/SASJobExecution/?_program=/Folder/Simple JSON&_action=execute
```

Use the concepts in this example if your application requires data in another format, such as XML or CSV.

Note: The %JESBEGIN and %JESEND macros are included by default and do not usually need to be added.
## Simple ODS HTML

This example creates HTML output using ODS and returns it to the web browser. An HTML input form provides a basic user interface to the program.

### Output

<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>66.0</td>
<td>112.5</td>
</tr>
<tr>
<td>Alice</td>
<td>F</td>
<td>13</td>
<td>56.5</td>
<td>84.0</td>
</tr>
<tr>
<td>Barbara</td>
<td>F</td>
<td>13</td>
<td>50.3</td>
<td>66.0</td>
</tr>
<tr>
<td>Carol</td>
<td>F</td>
<td>14</td>
<td>62.5</td>
<td>102.5</td>
</tr>
<tr>
<td>Henry</td>
<td>M</td>
<td>14</td>
<td>63.5</td>
<td>102.5</td>
</tr>
<tr>
<td>James</td>
<td>M</td>
<td>12</td>
<td>67.3</td>
<td>80.0</td>
</tr>
<tr>
<td>Jane</td>
<td>F</td>
<td>12</td>
<td>56.5</td>
<td>54.5</td>
</tr>
<tr>
<td>Janet</td>
<td>F</td>
<td>15</td>
<td>62.5</td>
<td>112.5</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>M</td>
<td>13</td>
<td>82.5</td>
<td>84.0</td>
</tr>
<tr>
<td>John</td>
<td>M</td>
<td>12</td>
<td>56.0</td>
<td>98.0</td>
</tr>
<tr>
<td>Joyce</td>
<td>F</td>
<td>11</td>
<td>51.3</td>
<td>50.5</td>
</tr>
<tr>
<td>Judy</td>
<td>F</td>
<td>14</td>
<td>64.3</td>
<td>80.0</td>
</tr>
<tr>
<td>Louise</td>
<td>F</td>
<td>12</td>
<td>56.3</td>
<td>77.0</td>
</tr>
<tr>
<td>Mary</td>
<td>F</td>
<td>15</td>
<td>66.5</td>
<td>112.0</td>
</tr>
<tr>
<td>Philip</td>
<td>M</td>
<td>18</td>
<td>72.0</td>
<td>150.0</td>
</tr>
<tr>
<td>Robert</td>
<td>M</td>
<td>12</td>
<td>84.8</td>
<td>128.0</td>
</tr>
<tr>
<td>Ronald</td>
<td>M</td>
<td>15</td>
<td>67.0</td>
<td>133.0</td>
</tr>
<tr>
<td>Thomas</td>
<td>M</td>
<td>11</td>
<td>57.5</td>
<td>85.0</td>
</tr>
<tr>
<td>William</td>
<td>M</td>
<td>13</td>
<td>66.5</td>
<td>112.0</td>
</tr>
</tbody>
</table>

### Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,prompts,execute</td>
<td>Displays the HTML input form or prompts if available; otherwise, executes the job</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>ods_html5</td>
<td>Specifies that ODS HTML5 output is created by the job</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
<td>Specifies the name of the ODS style</td>
</tr>
</tbody>
</table>
HTML Input Form

1. **SAS® Job Execution**

2. **Simple ODS HTML**

   The PRINT procedure creates a simple HTML page that displays the data in the SASHELP.CLASS table.

3. Run code □ Show SAS Log

```html
<!DOCTYPE html>
<html lang="en">
<head>
<title>Simple ODS HTML</title>
<link rel="stylesheet" href="/SASJobExecution/theme">
</head>
<body role="main" class="jobexec_body">
<form class="jobexec_form" action="/SASJobExecution/" target="_tab">
  <input type="hidden" name="_program" value="$PROGRAM$"/>
  <input type="hidden" name="_action" value="execute"/>
  <input type="hidden" name="_output_type" value="ods_html5"/>
  <div class="jobexec_sample_header">SAS<sup>®</sup> Job Execution</div>
  <h1 class="jobexec_sample_name">Simple ODS HTML</h1>
  <p>The PRINT procedure creates a simple HTML page that displays the data in the SASHELP.CLASS table.</p>
  <hr/>
  <input type="submit" value="Run code" class="jobexec_sample_input_submit"/>
  <input type="checkbox" name="_debug" id="_debug" value="log" class="jobexec_sample_input_checkbox"/>
  <label for="_debug">Show SAS Log</label>
</form>
</body>
</html>
```

1. This HTML is similar to the HTML in the Hello World input form.
2. This HTML is similar to the HTML in the Hello World input form.
3. Specify $PROGRAM$ as the value for the _PROGRAM input tag. When the HTML input form is displayed, the path and name of the program to execute are substituted for $PROGRAM$. 
   _ACTION is set to execute to ensure that the program executes when you click **Run Code**. The value of _OUTPUT_TYPE indicates that ODS HTML5 output is created by the job. The
%JESBEGIN macro issues a FILENAME statement that supports HTML output, and it issues an ODS statement using the HTML5 destination.

When the form is submitted, the following global macro variables are defined just before SAS code execution, but they are not used by the program:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>execute</td>
</tr>
<tr>
<td>_ODSDEST</td>
<td>html5</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>ods_html5</td>
</tr>
<tr>
<td>_PROGRAM</td>
<td>/Folder/Simple ODS HTML</td>
</tr>
<tr>
<td>_DEBUG</td>
<td>log (if the check box is selected)</td>
</tr>
</tbody>
</table>

The _ODSDEST macro variable is derived from the value of _OUTPUT_TYPE, and it indicates the ODS destination that is used.

Program

```sas
    title 'Student Data';
    proc print data=sashelp.class noobs;
        var name sex age height weight;
    run; quit;
```

Program Description

The %JESBEGIN macro performs several tasks before executing the code. A FILENAME statement is issued to return HTML output to the web browser because ods_html5 is specified as the value for the _OUTPUT_TYPE input parameter. An ODS statement for the HTML5 destination is also issued.

The HTML is written to the _WEBOUT FILEREF assigned by the %JESBEGIN macro, and the HTMLBlue style, specified in the _ODSSTYLE input parameter, controls the appearance of the output.

The PRINT procedure displays the SASHELP.CLASS table.

The %JESEND macro executes after the last line of code and closes all open ODS destinations.

Note: The %JESBEGIN and %JESEND macros are included by default and do not usually need to be added.
Upload a CSV File

This example uploads a Comma-Separated Value (CSV) file to the server, imports it into a SAS table, and then displays the first 10 records of the SAS table. An HTML input form, which enables you to select a file to upload, provides a basic user interface to the program.

Output

The following output is an example of a CSV file that could be uploaded.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>59</td>
<td>112.5</td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
<td>F</td>
<td>13</td>
<td>50.5</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>Nathan</td>
<td>F</td>
<td>13</td>
<td>65.3</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>Carol</td>
<td>F</td>
<td>14</td>
<td>62.8</td>
<td>102.5</td>
</tr>
<tr>
<td>5</td>
<td>Henry</td>
<td>M</td>
<td>14</td>
<td>62.5</td>
<td>102.5</td>
</tr>
<tr>
<td>6</td>
<td>James</td>
<td>M</td>
<td>12</td>
<td>57.3</td>
<td>88</td>
</tr>
<tr>
<td>7</td>
<td>Jane</td>
<td>F</td>
<td>12</td>
<td>56.6</td>
<td>64.5</td>
</tr>
<tr>
<td>8</td>
<td>Janet</td>
<td>F</td>
<td>15</td>
<td>62.5</td>
<td>112.5</td>
</tr>
<tr>
<td>9</td>
<td>Jeffrey</td>
<td>M</td>
<td>13</td>
<td>62.6</td>
<td>94</td>
</tr>
<tr>
<td>10</td>
<td>John</td>
<td>M</td>
<td>12</td>
<td>50</td>
<td>62.5</td>
</tr>
</tbody>
</table>

Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,execute</td>
<td>Displays the HTML input form before the job is executed</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>ods_html5</td>
<td>Specifies that ODS HTML5 output is created by the job</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
<td>Specifies the name of the ODS style</td>
</tr>
</tbody>
</table>
HTML Input Form

SAS® Job Execution

Upload a CSV File

Use this page to upload a CSV file from your local machine to the SAS server machine. The program imports the file into a SAS table and then uses PROC PRINT to display the first 10 records.

Optionally, you can choose to download the file from the SAS server machine by clicking a link in the output.

Choose a CSV file to upload: [Choose File] No file chosen

Run code  Show SAS Log

<!-- DOCTYPE html -->
<html lang="en">
<head>
<title>Upload a CSV File</title>
<link rel="stylesheet" href="/SASJobExecution/theme">
</head>
<body role="main" class="jobexec_body">
<form class="jobexec_form" action="/SASJobExecution/" method="post" target="_tab" enctype="multipart/form-data">
<input type="hidden" name="_program" value="$PROGRAM$" />
<input type="hidden" name="_action" value="execute" />
<input type="hidden" name="_output_type" value="ods_html5"/>
<input type="hidden" name="_csrf" value="$CSRF$"/>
<div class="jobexec_sample_header">SAS<sup>®</sup> Job Execution</div>
<h1 class="jobexec_sample_name">Upload a CSV File</h1>
<p>Use this page to upload a CSV file from your local machine to the SAS server machine. The program imports the file into a SAS table and then uses PROC PRINT to display the first 10 records.</p>
<p>Optionally, you can choose to download the file from the SAS server machine by clicking a link in the output.</p>
<hr/>
This input form is functionally equivalent to the form used in “Upload a File” on page 106.

The file that you choose to upload should be a CSV file with the .csv file extension.

When the form is submitted, the following global macro variables are defined just before SAS code execution, but they are not used by the program:

<table>
<thead>
<tr>
<th>Macro Variable Name</th>
<th>Macro Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_CSRF</td>
<td>Cross-Site Request Forgery token for this request</td>
</tr>
<tr>
<td>_ACTION</td>
<td>execute</td>
</tr>
<tr>
<td>_ODSDEST</td>
<td>html5</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>ods_html5</td>
</tr>
<tr>
<td>_PROGRAM</td>
<td>/Folder/Upload a CSV File</td>
</tr>
<tr>
<td>_DEBUG</td>
<td>log (if the check box is selected)</td>
</tr>
</tbody>
</table>

Additional macro variables with information about the uploaded file are created. See “Upload a File” on page 106 for more information.

Program

* Check the file extension to verify that it is a CSV file;

data _null_;  
length filename $1024;  
filename = htmlencode(strip(&WEBIN_FILENAME));  
call symputx('_WEBIN_FILENAME', filename);
if (upcase("&_WEBIN_FILEEXT") ne 'CSV') then do;
    rc = dosubl('ods all close;');
    file _webout;
    put '<!DOCTYPE html>';
    put '<html lang="en">';
    put '<head><title>Program Error</title></head>';  
    put '<body role="main">';
    put '<h1>ERROR: Uploaded file "' filename +(-1) '" is not a CSV file.</h1>';  
    put '</body>';  
    put '</html>';  
    abort cancel;
end;
run;

* Create a FILEREF for the uploaded file;
filename upload filesrvc parenturi="&SYS_JES_JOB_URI"  
    name="&_WEBIN_FILENAME"  
    contenttype="&_WEBIN_CONTENT_TYPE";

* Set options to support non-SAS name;
options validvarname=any validmemname=extend;

* Import the uploaded CSV file;
proc import datafile=upload  
    out=work.mydata  
    dbms=csv  
    replace;  
    getnames=yes;  
    run; quit;

title 'First 10 Records of Uploaded File ' ""&_WEBIN_FILENAME"";
footnote link="&_WEBIN_FILEURI/content" 'Click here to download file';

proc print data=work.mydata(obs=10)  
    style(header)=[just=center]  
    style(column)=[verticalalign=middle];  
    run; quit;

Program Description
The %JESBEGIN macro performs several tasks before executing the code. A FILENAME statement is issued to return HTML output to the web browser because ods_html5 is specified as the value for the _OUTPUT_TYPE input parameter. An ODS statement for the HTML5 destination is also issued. The HTML is written to the _WEBOUT FILEREF that is assigned by the %JESBEGIN macro, and the HTMLBlue style, specified in the _ODSSTYLE input parameter, controls the appearance of the output.

Additional macro variables with information about the uploaded file are created during the upload process. See “Upload a File” on page 106 for more information.

The HTMLENCODE function encodes the values of input parameters to prevent execution of malicious code in the web browser.
If the value of the _WEBIN_FILEEXT macro variable indicates that a CSV file was not uploaded, then action is taken to prevent further code execution. The DOSUBL function closes all open ODS destinations, and an error message is returned to the web browser. No additional program statements are executed after the ABORT statement executes.

The FILENAME statement creates a reference to the temporary location of the uploaded file. The file can be used with PROC IMPORT or with any other code that accepts a FILEREF. For example, you can use the FCOPY function to make a copy of the file.

PROC IMPORT creates the WORK.MYDATA table from the uploaded CSV file.

The PRINT procedure displays the first 10 records of the SAS table that is created by the IMPORT procedure, and a link to download the CSV file is created by the FOOTNOTE statement.

The %JESEND macro executes after the last line of code and closes all open ODS destinations.

Note: The %JESBEGIN and %JESEND macros are included by default and do not usually need to be added.

Upload a File

This example uploads an arbitrary file to the server and then displays information about the file. An HTML input form, which enables you to select a file to upload, provides a basic user interface to the program.

Output

The following output is an example of a file that could be uploaded.

<table>
<thead>
<tr>
<th>SAS Macro Variables Generated for Uploaded File “Capture.PNG”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

Click here to download file
Job Definition Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>form,execute</td>
<td>Displays the HTML input form before the job is executed</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>ods_html5</td>
<td>Specifies that ODS HTML5 output is created by the job</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>HTMLBlue</td>
<td>Specifies the name of the ODS style</td>
</tr>
</tbody>
</table>

HTML Input Form

**SAS® Job Execution**

**Upload a File**

Use this page to upload a file from your local machine to the SAS server machine. The program displays the macro variables with information about the file.

Optionally, you can choose to download the file from the SAS server machine by clicking a link in the output.

Choose a file to upload: [Choose File] No file chosen

Run code  Show SAS Log

```html
<!DOCTYPE html>
<html lang="en">
<head>
<title>Upload a File</title>
<link rel="stylesheet" href="/SASJobExecution/theme">
</head>
<body role="main" class="jobexec_body">

1
<form class="jobexec_form" action="/SASJobExecution/" method="post" target="_tab"

type="hidden" name="_program" value="$PROGRAM$" />
<input type="hidden" name="_action" value="execute" />
<input type="hidden" name="_output_type" value="ods_html5" />
2
<input type="hidden" name="_csrf" value="$CSRF$" />
<div class="jobexec_sample_header">SAS<sup>®</sup> Job Execution</div>
</form>
<h1 class="jobexec_sample_name">Upload a File</h1>
```
```
Use this page to upload a file from your local machine to the SAS server machine. The program displays the macro variables with information about the file.

Optionally, you can choose to download the file from the SAS server machine by clicking a link in the output.

When you upload a file, be sure to specify the METHOD and ENCTYPE attributes exactly as shown.

When you make a POST request, you must specify the _CSRF input tag exactly as shown. This tag ensures the request is considered non-malicious by sending a Cross-Site Request Forgery token to the server. When the HTML input form is displayed, the Cross-Site Request Forgery token for this request is substituted for $CSRF$.

A special form of the INPUT tag displays a file selector dialog box when you click a button. Navigate to the file that you want to upload, select it, and then run the code. The file is transferred to the SAS server and stored in the SAS Infrastructure Data Server using the Files service.

Specify $PROGRAM$ as the value for the _PROGRAM input tag. When the HTML input form is displayed, the path and name of the program to execute are substituted for $PROGRAM$.

The HTML in this form has some of the same elements and features of the form used in “Simple ODS HTML” on page 99. See that section for more information.

When the form is submitted, the following global macro variables are defined just before SAS code execution, but they are not used by the program:
Macro Variable Name | Macro Variable Value
---|---
_CSRF | Cross-Site Request Forgery token for this request
_ACTION | execute
_ODSDEST | html5
_ODSSTYLE | HTMLBlue
_OUTPUT_TYPE | ods_html5
_PROGRAM | /Folder/Upload a File
_DEBUG | log (if the check box is selected)

The _ODSDEST macro variable is derived from the value of _OUTPUT_TYPE and indicates the ODS destination that is used.

Additional macro variables with information about the uploaded file are created. See the Output and Program Description sections for more information.

Program

* Create a data set with information about the upload;

data work.upload_info;

    length varname $25 value $1024 description $256;

    varname     = '_WEBIN_CONTENT_LENGTH';
    value       = symget('_WEBIN_CONTENT_LENGTH');
    description = 'Specifies the size of the file that was uploaded in bytes  
                   (supplied automatically by the Web browser).';
    output;

    varname     = '_WEBIN_CONTENT_TYPE';
    value       = resolve(symget('_WEBIN_CONTENT_TYPE'));
    description = 'Specifies the content type that corresponds to the file that was uploaded  
                   (supplied automatically by the Web browser).';
    output;

    varname     = '_WEBIN_FILE_COUNT';
    value       = symget('_WEBIN_FILE_COUNT');
    description = 'Specifies the number of files that were uploaded.';
    output;

    varname     = '_WEBIN_FILEEXT';
    value       = resolve(symget('_WEBIN_FILEEXT'));
    description = 'Specifies the extension of the file that was uploaded.';
    output;
Program Description

The %JESBEGIN macro performs several tasks before executing the code. A FILENAME statement is issued to return HTML output to the web browser because ods_html5 is specified as the value for the _OUTPUT_TYPE input parameter. An ODS statement for the HTML5 destination is also issued.

The HTML is written to the _WEBOUT FILEREF assigned by the %JESBEGIN macro, and the HTMLBlue style, specified in the _ODSSTYLE input parameter, controls the appearance of the output.

Additional global macro variables with information about the file are created as part of the upload process. This information is retrieved and stored in the WORK.UPLOAD_INFO table.

The _WEBIN_FILEURI macro variable is of special interest because it provides a reference to the temporary location of the uploaded file. This location is associated with the job execution object and deleted when the job expires.

The FOOTNOTE statement shows an example of downloading the file by adding /content to the end of the _WEBIN_FILE_URI macro variable. The "Upload a CSV File" on page 102 provides an example of referencing the uploaded file using a FILENAME statement.

The PRINT procedure displays information about the uploaded file.

The %JESEND macro executes after the last line of code and closes all open ODS destinations.

Note: The %JESBEGIN and %JESEND macros are included by default and do not usually need to be added.
Reference

%JESBEGIN and %JESEND AutoCall Macros

Operation of the %JESBEGIN and %JESEND macros is controlled by global macro variables that are created from job input parameters of the same name. All input parameters are optional.

Table 7  Global Macro Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>_CONTDISP</td>
<td>Any valid value</td>
<td>Specifies a value for the CONTENTDISP option in the FILENAME statement.</td>
</tr>
<tr>
<td>_CONTDISP_FILEEXT</td>
<td></td>
<td>Specifies the file extension to use in the CONTENTDISP option in the FILENAME statement if _CONTDISP is not specified. The default value is derived based on the type of output being generated (the value of the _OUTPUT_TYPE parameter).</td>
</tr>
<tr>
<td>_CONTDISP_FILENAME</td>
<td></td>
<td>Specifies the file name to use in the CONTENTDISP option in the FILENAME statement if _CONTDISP is not specified. The default value is SASResults.</td>
</tr>
<tr>
<td>_CONTTYPE</td>
<td>Any valid value</td>
<td>Specifies a value for the CONTENTTYPE option in the FILENAME statement.</td>
</tr>
<tr>
<td>_DEBUG</td>
<td>Any valid value for _DEBUG</td>
<td>If one of the values for _DEBUG is trace, then MPRINT is turned on, LS is set to max, and additional debug messages are printed in the log.</td>
</tr>
<tr>
<td>_ENCODING</td>
<td>Any valid SAS encoding</td>
<td>Specifies a value for the ENCODING option in the ODS statement.</td>
</tr>
<tr>
<td>_FILEREF_OPTIONS</td>
<td>Any valid value for _FILEREF_OPTIONS</td>
<td>Specifies a value to add to the end of the FILENAME statement.</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Action</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>_GOPT_DEVICE</td>
<td>Any valid value for the DEVICE graphic option</td>
<td>Specifies a value to add to a GOPTIONS statement.</td>
</tr>
<tr>
<td>_GOPT_HSIZE</td>
<td>Any valid value for the HSIZE graphic option</td>
<td>Specifies a value to add to a GOPTIONS statement.</td>
</tr>
<tr>
<td>_GOPT_VSIZE</td>
<td>Any valid value for the VSIZE graphic option</td>
<td>Specifies a value to add to a GOPTIONS statement.</td>
</tr>
<tr>
<td>_GOPT_XPIXELS</td>
<td>Any valid value for the XPIXELS graphic option</td>
<td>Specifies a value to add to a GOPTIONS statement.</td>
</tr>
<tr>
<td>_GOPT_YPIXELS</td>
<td>Any valid value for the YPIXELS graphic option</td>
<td>Specifies a value to add to a GOPTIONS statement.</td>
</tr>
<tr>
<td>_GOPTIONS</td>
<td>Any value that is valid in a GOPTIONS statement</td>
<td>Specifies a value to add to a GOPTIONS statement.</td>
</tr>
<tr>
<td>_ODSDEVICE</td>
<td>Any valid graphics device</td>
<td>Specifies a value for the OUTPUTFMT option of the ODS GRAPHICS statement.</td>
</tr>
<tr>
<td>_ODS_EMBED_GRAPHICS</td>
<td>N, NO (case–insensitive)</td>
<td>Specifies ODS options for embedding graphics. This macro is valid only for the HTML5 destination. By default, these options are turned on.</td>
</tr>
<tr>
<td>_ODSOPTIONS</td>
<td>Any valid ODS options</td>
<td>Specifies a value to add to the ODS statement.</td>
</tr>
<tr>
<td>_ODSTYLE</td>
<td>Any valid ODS style</td>
<td>Specifies a value for the STYLE option in the ODS statement.</td>
</tr>
<tr>
<td>_ODSTYLESHEET_URL</td>
<td>Any valid value</td>
<td>Specifies a value for the URL suboption for the STYLESHEET option in the ODS statement. Note: If you are accessing a style sheet outside of the domain, you must use SAS Environment Manager to add a new content security policy. See SAS Viya Administration: Configuration Properties for more information.</td>
</tr>
</tbody>
</table>
**_OUTPUT_TYPE**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>_OUTPUT_TYPE</td>
<td>NONE, ODS _ods-destination, html, pdf, json, and so on (case-insensitive)</td>
<td>Specifies a value for the OUTPUT option in the ODS statement. Specifying NONE is almost the same as omitting the macros from the user code. Global macro variable values are displayed by the %JESBEGIN macro and then the macro exits. The %JESEND macro exits without performing any tasks. The default value is ods_html5.</td>
</tr>
</tbody>
</table>

**_SUPPRESS_MVARS**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>_SUPPRESS_MVARS</td>
<td>Y, YES (case-insensitive)</td>
<td>Suppresses the display of macro variables (by the %JESBEGIN macro) before executing user code.</td>
</tr>
</tbody>
</table>

The %JESBEGIN macro creates the following global macro variables:

**Table 8  Global Macro Variables Created by %JESBEGIN**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>_JOBERROR</td>
<td>0 for success, nonzero for failure</td>
<td>Indicates whether the %JESBEGIN macro executed successfully.</td>
</tr>
<tr>
<td>_ODSDEST</td>
<td>Value to the right of ods_ in the value of the _OUTPUT_TYPE parameter</td>
<td>Specifies which ODS destination is used, if any.</td>
</tr>
<tr>
<td>_STATUS_MESSAGE</td>
<td>Any plain text</td>
<td>Passes an error message to the SAS Job Execution Web Application.</td>
</tr>
</tbody>
</table>

**Reserved Macro Variables**

**Table 9  Reserved Macro Variables**

<table>
<thead>
<tr>
<th>SAS Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ACTION</td>
<td>Specifies the _ACTION job input parameter, if any.</td>
</tr>
<tr>
<td>_ADDJESBEGINENDMACROS</td>
<td>Specifies whether to add the %JESBEGIN and %JESEND macros to the code.</td>
</tr>
<tr>
<td>_APSLIST</td>
<td>Specifies a list of job input parameters.</td>
</tr>
<tr>
<td>_CONTEXTNAME</td>
<td>Specifies the Compute service context name.</td>
</tr>
<tr>
<td>SAS Variable Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>_CONTDISP</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_CONTDISP_FILEEXT</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_CONTDISP_FILENAME</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_CONTTYPE</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_CSRF</td>
<td>Specifies the Cross-Site Request Forgery token for this request.</td>
</tr>
<tr>
<td>_DEBUG</td>
<td>Specifies the _DEBUG job input parameter, if any.</td>
</tr>
<tr>
<td>_ENCODING</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_FILEREF_OPTIONS</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_FILESRV_fileref_URI</td>
<td>Specifies the URI of a file created by a FILEREF using the FILESRVC engine.</td>
</tr>
<tr>
<td>_GOPT_DEVICE</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_GOPT_HSIZE</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_GOPT_VSIZE</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_GOPT_XPIXELS</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_GOPT_YPIXELS</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_OPTIONS</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_HTUA</td>
<td>Specifies the name of the user agent.</td>
</tr>
<tr>
<td>_JOB</td>
<td>Specifies a globally unique identifier.</td>
</tr>
<tr>
<td>_JOBERROR</td>
<td>Acts as an output parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_ODS_DEVICE</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_ODS_EMBED_GRAPHICS</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_ODSDEST</td>
<td>Acts as an output parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_ODSOPTIONS</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_ODSSTYLE</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>SAS Variable Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>_ODSSSTYLESHEET_URL</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_OMITJSONLISTING</td>
<td>Specifies whether an internal JSON listing file is returned.</td>
</tr>
<tr>
<td>_OMITJSONLOG</td>
<td>Specifies whether an internal JSON log file is returned.</td>
</tr>
<tr>
<td>_OMITSESSIONRESULTS</td>
<td>Specifies whether any results are returned.</td>
</tr>
<tr>
<td>_OMITTEXTLISTING</td>
<td>Specifies whether an internal text listing file is returned.</td>
</tr>
<tr>
<td>_OMITTEXTLOG</td>
<td>Specifies whether an internal text log is returned.</td>
</tr>
<tr>
<td>_OUTPUT_TYPE</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_PROGRAM</td>
<td>Specifies the path and name of the job.</td>
</tr>
<tr>
<td>_REPLAY</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>_RESULTFILE</td>
<td>Specifies the output result files to be returned.</td>
</tr>
<tr>
<td>_RMTADDR</td>
<td>Specifies the Internet Protocol (IP) address of the client that sent the request.</td>
</tr>
<tr>
<td>_RMTHOST</td>
<td>Specifies the fully qualified name of the client that sent the request or the IP address of the client if the name cannot be determined.</td>
</tr>
<tr>
<td>_SAVEFILE</td>
<td>Specifies the name of the file for saved output.</td>
</tr>
<tr>
<td>_SAVEFOLDER</td>
<td>Specifies the name of the folder for saved output.</td>
</tr>
<tr>
<td>_STATUS_MESSAGE</td>
<td>Specifies the message text that is displayed by the client after a job executes.</td>
</tr>
<tr>
<td>_SUPPRESS_MVARS</td>
<td>Acts as an input parameter for the %JESBEGIN macro.</td>
</tr>
<tr>
<td>_URL</td>
<td>Specifies the URL of the web server middle tier that is used to access the job.</td>
</tr>
<tr>
<td>_USERLOCALE</td>
<td>Specifies the locale for the user that was set in the user preferences. If this value was not set, it contains the locale sent in the HTTP request Accept-Language header.</td>
</tr>
<tr>
<td>_VERSION</td>
<td>Specifies the SAS Job Execution Web Application version number.</td>
</tr>
<tr>
<td>_XFORWARD</td>
<td>Specifies the host and port of the original HTTP request.</td>
</tr>
<tr>
<td>SAS Variable Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>_WEBIN_CONTENT_LENGTH</td>
<td>Contain properties of the file that is being uploaded. See the Upload a File and Upload a CSV File sections in “Samples” on page 51 for more information.</td>
</tr>
<tr>
<td>_WEBIN_CONTENT_TYPE</td>
<td></td>
</tr>
<tr>
<td>_WEBIN_FILE_COUNT</td>
<td></td>
</tr>
<tr>
<td>_WEBIN_FILEEXT</td>
<td></td>
</tr>
<tr>
<td>_WEBIN_FILENAME</td>
<td></td>
</tr>
<tr>
<td>_WEBIN_FILEURI</td>
<td></td>
</tr>
<tr>
<td>_WEBIN_NAME</td>
<td></td>
</tr>
<tr>
<td>SYS_COMPUTE_JOB_ID</td>
<td>Specifies the Compute service job ID.</td>
</tr>
<tr>
<td>SYS_COMPUTE_SESSION_ID</td>
<td>Specifies the Compute service session ID.</td>
</tr>
<tr>
<td>SYS_COMPUTE_SESSION_OWNER</td>
<td>Specifies the user name that submitted the job.</td>
</tr>
<tr>
<td>SYS_JES_JOB_URI</td>
<td>Specifies the Job Execution service object URI.</td>
</tr>
</tbody>
</table>

**_ACTION Input Parameter Values**

The following values are supported by the _ACTION parameter:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>Executes the job in the background.</td>
</tr>
<tr>
<td>execute</td>
<td>Executes the job.</td>
</tr>
<tr>
<td>form</td>
<td>Displays an HTML input form file stored in the folder structure before job execution.</td>
</tr>
<tr>
<td>json</td>
<td>Returns a list of unexpired jobs or sample jobs in JSON format.</td>
</tr>
<tr>
<td>lastjob</td>
<td>Displays output from a previous job execution if it has not yet expired.</td>
</tr>
<tr>
<td>prompts</td>
<td>Displays a prompt file stored in the folder structure before job execution.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>schedule</td>
<td>Indicates that a job is to be scheduled using SAS Environment Manager.</td>
</tr>
<tr>
<td>wait</td>
<td>Displays a wait screen with informational text while the job is executing.</td>
</tr>
</tbody>
</table>

PARAM_LIST Macro

The param_list macro is used to convert the parameter list generated by a multiple–value prompt into a form that is useful in your SAS code.

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvar</td>
<td>Required. Specifies the name of the macro variable that corresponds to the prompt name.</td>
</tr>
<tr>
<td>outvar</td>
<td>Optional. Specifies the name of the macro variable that contains the converted parameter list. If the name is not specified, an underscore (_) is added to the beginning of the value specified in mvar.</td>
</tr>
<tr>
<td>dlm</td>
<td>Optional. Specifies a character that is used to delimit values in the converted parameter list. A blank space is used by default. If a character is specified, the delimiter followed by a blank space is used.</td>
</tr>
<tr>
<td>quote</td>
<td>Optional. Specify y (case-insensitive) to quote the individual values in the converted parameter list. By default, the value is n.</td>
</tr>
</tbody>
</table>

The value of the macro variable specified in the OUTVAR argument is valid when one or more values are selected in the prompt. If no values are selected, then the macro variable is assigned a blank value.

Example: Using Prompt Values in a VAR Statement

This example assumes that a prompt is used to specify one or more column names in the SASHELP.CLASS table using the PRINT procedure:

```sas
%param_list(mvar=prompt_vals, outvar=column_list)
proc print data=sashelp.class;
  var &COLUMN_LIST;
run; quit;
```
The code fails if the prompt_vals macro variable is blank because the column_list macro variable does not have a value. This might happen if no values were selected in the prompt. One way to avoid this problem is to use the IFC function:

```sas
proc print data=sashelp.class;
   %sysfunc(ifc(%sysfunc(length(&COLUMN_LIST)) gt 0,
          "var &COLUMN_LIST",
          ));
run; quit;
```

Example: Using Prompt Values in a SELECT Statement

This example assumes that a prompt is used to specify one or more column names in the SASHELP.CLASS table using the SQL procedure:

```sas
%param_list(mvar=prompt_vals, outvar=column_list, dlm=%str(,))
```

```sas
proc sql;
   create table work.class as
   select &COLUMN_LIST
   from sashelp.class;
run; quit;
```

See the previous example for information about handling missing prompt values.

Example: Using Prompt Values in a WHERE Statement

This example assumes that a prompt is used to select one or more age values (for example, 12, 14, and 16):

```sas
%param_list(mvar=prompt_vals, outvar=value_list, dlm=%str(,))
```

```sas
proc print data=sashelp.class;
   where age in (&VALUE_LIST);
run; quit;
```

This example assumes that a prompt is used to select M, F, or both values (case-sensitive). Specify y for the quote argument because the values of character variables must be quoted:

```sas
%param_list(mvar=prompt_vals, outvar=value_list, dlm=%str(,), quote=y)
```

```sas
proc print data=sashelp.class;
   where sex in (&VALUE_LIST);
run; quit;
```

Source Code

```sas
%macro param_list(mvar=, outvar=, dlm=, quote=n);
%local I PARAMLIST;
%if (%bquote(&MVAR) eq ) %then %do;
   %put ERROR: You must specify a value for the MVAR argument.;
   data _null_;
   abort cancel nolist;
   run;
   %goto exit;
%end;
```

```sas
%macrorange
```
%if (%symexist(&MVAR) ne 1) %then %do;
  %put ERROR: Macro variable "&MVAR" does not exist.;
  data _null_;
  abort cancel nolist;
  run;
  %goto exit;
%end;
%goto exit;
%end;

%if (%bquote(%upcase(&QUOTE)) ne Y) and
   (%bquote(%upcase(&QUOTE)) ne N) %then %do;
  %put ERROR: You must specify either Y or N for the QUOTE argument.;
  data _null_;
  abort cancel nolist;
  run;
  %goto exit;
%end;

%let QUOTE=%upcase(&QUOTE);

%let OUTVAR=%upcase(&OUTVAR);

%global &OUTVAR;
%global &MVAR.0;

%if (%bquote(&&MVAR.0) eq ) %then %do;
  %if (&QUOTE eq Y)
    %then %let PARAMLIST=%sysfunc(quote(%bquote(&&MVAR)));%else %let PARAMLIST=%bquote(&&MVAR);
  %end;
  %else %do I = 1 %to &&MVAR.0;
    %if (&I eq 1) %then %do;
      %if (&QUOTE eq Y)
        %then %let PARAMLIST=%sysfunc(quote(%bquote(&&MVAR&I)));%else %let PARAMLIST=%bquote(&&MVAR&I);
      %end;
      %else %do;
        %if (&QUOTE eq Y)
          %then %let PARAMLIST=&PARAMLIST.&DLM %sysfunc(quote(%bquote(&&MVAR&I)));%else %let PARAMLIST=&PARAMLIST.&DLM %bquote(&&MVAR&I);
        %end;
      %end;
    %end;
  %end;
%end;

%let &OUTVAR=&PARAMLIST;

119
Modifying Your Settings

You can use the Settings window to edit user preferences or customize accessibility settings. Changing these settings does not impact other users. To access these settings, click the user button in the application bar and select Settings.

General

The General section includes settings that enable users to change the appearance of the web application, enable warning and information messages to be displayed, and choose a profile picture. Here are the settings:

- You can change the appearance of the web application by using the Theme setting. The default theme is set by an administrator. The theme specifies the collection of colors, graphics, and fonts that appear in the application. You can also apply other themes, including both SAS themes and custom themes, if available.

  Select Choose a theme, and then select another theme from the drop-down list to change the look of the application. The theme change takes effect after you close the Settings window.

  SAS themes:
  
  **High Contrast**
  
  Presents a dark background with high-contrast foreground elements to meet the needs of users with low vision.

  **Illuminate**
  
  Presents a clean and uncomplicated color palette that is easy to use.

  **Inspire**
  
  Presents vibrant and cohesive colors that shift the emphasis from the application to the content.

- If you want messages to display that you previously asked not to display, click Reset Messages. By default, all warning and information messages are displayed.

- You can select a profile picture to display in the application bar, as well as in other places within the application that use profile pictures. The default is the first initial or character of your first name.

  Click Choose Picture and then select an image file to upload. The image file’s size can be up to 1 MB. The valid file types are BMP, GIF, JPEG, JPG, and PNG.

Region and Language

The Region and Language section includes the following settings that enable users to specify the locale for regional formats and sorting, as well as for offline processes:
The **Locale for regional formats and sorting** setting specifies the locale that is used for sorting data and formatting values such as dates, times, numbers, and currency. By default, the browser locale is used. Changes take effect after you sign out and sign back in.

The **Locale for offline processes** setting specifies the locale that is used for offline jobs or background processes such as report distributions or notifications. By default, the locale of the Java Runtime Environment is used.

### Accessibility

Several settings in the **Accessibility** section can assist people who rely on assistive technologies:

- **Select Enable sounds** to hear audio indicators for events that occur within the user interface.
- **Select Enable visual effects** to show visual effects that indicate state changes. For example, when this setting is selected, you will see a subtle movement in the user interface if you delete an item.
- **Select Adjust the display duration for pop-up notifications** to specify how long temporary pop-up notification messages are displayed. Increasing the amount of time that notifications are displayed can help users discover and read messages that disappear automatically.
- **Select Invert application colors** to make the user interface easier to see for users with sensitivity to certain bright colors (for example, a black-on-white display). You can also use the Ctrl+` (Ctrl +back quote) keyboard shortcut to invert the application colors.
- **Select Display tooltips when using the keyboard to navigate** to enable keyboard users to access tooltips. When this option is selected, putting keyboard focus on a control also displays the tooltip on the screen. You can also select the location in the browser window to display the tooltip. By default, the tooltip is displayed in the bottom right corner of the browser window.
- The focus indicator is an outline that indicates which user interface component is active. You can make the focus indicator easier to see by selecting **Customize the focus indicator settings** and adjusting the color, thickness, and opacity.