SAS® Job Execution Web Application 2.1: User's Guide

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 HTML input form can be created to provide a user interface to the job. The job definitions and HTML input forms 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 HTML 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 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 HTML input form, select a link, or view an inline image that is displayed in a web browser.
- Web page authors create the HTML 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 HTML input form.

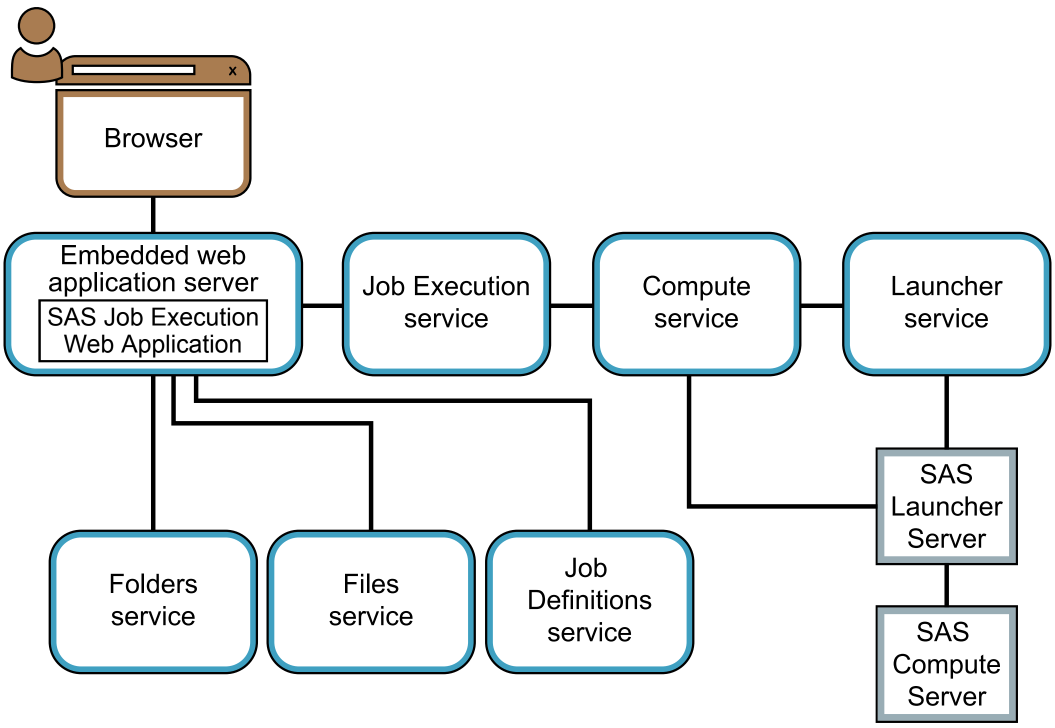
The SAS Job Execution Web Application executes a stored job when it receives a request with a parameter indicating the location of the job. The request is initiated by accessing a URL with the following general format:

```
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.

   Note: Each client user must have a user account on the Compute Server machine to run a request.

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.

8 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.

9 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

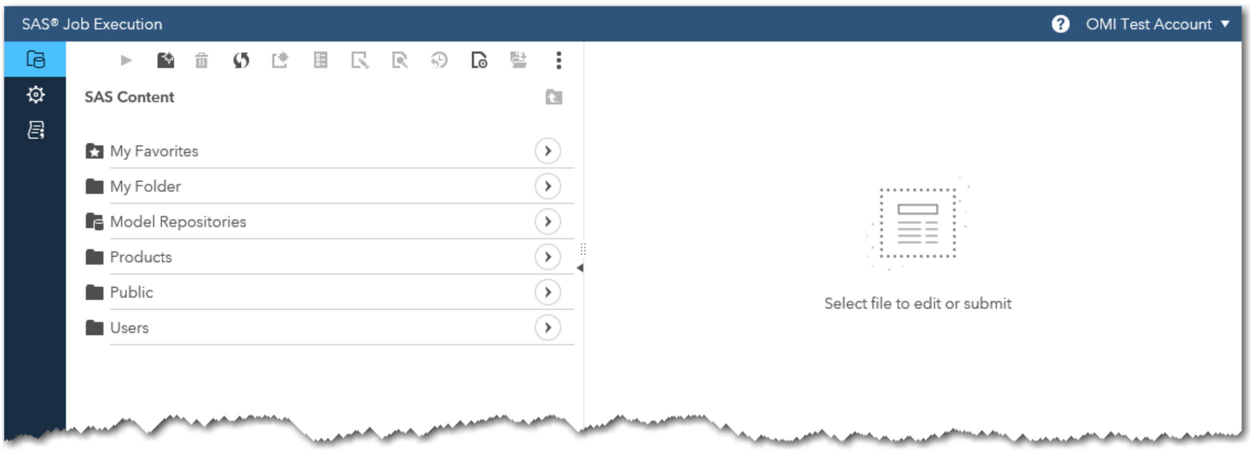
Access the SAS Job Execution Web Application using this URL:

`http://`*`host`*`:`*`port`*`/SASJobExecution`

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

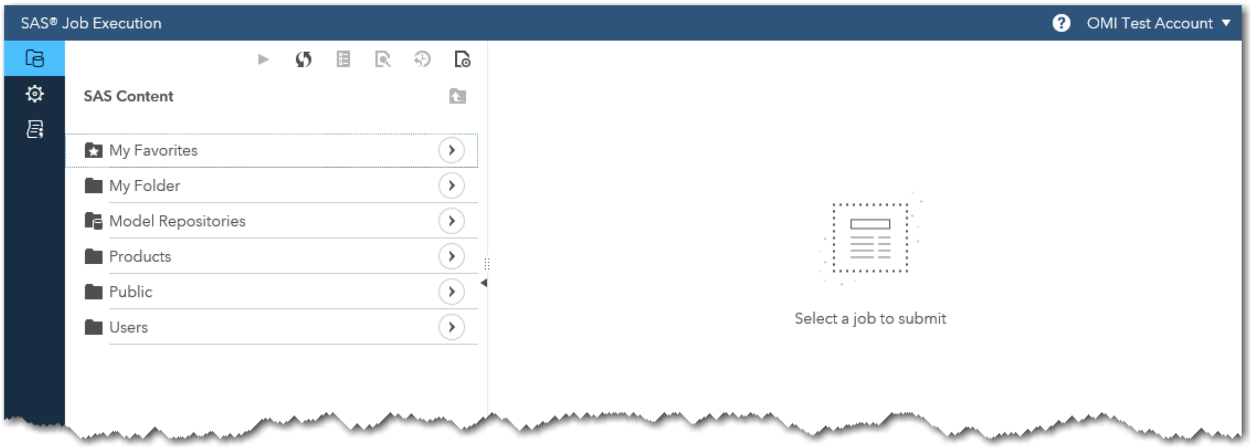
Accessing Content

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, you can click to access it. The left pane displays folders and their contents, and the right pane is used as a presentation and working area for the various job functions:



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 19 for more information about creating and editing files.

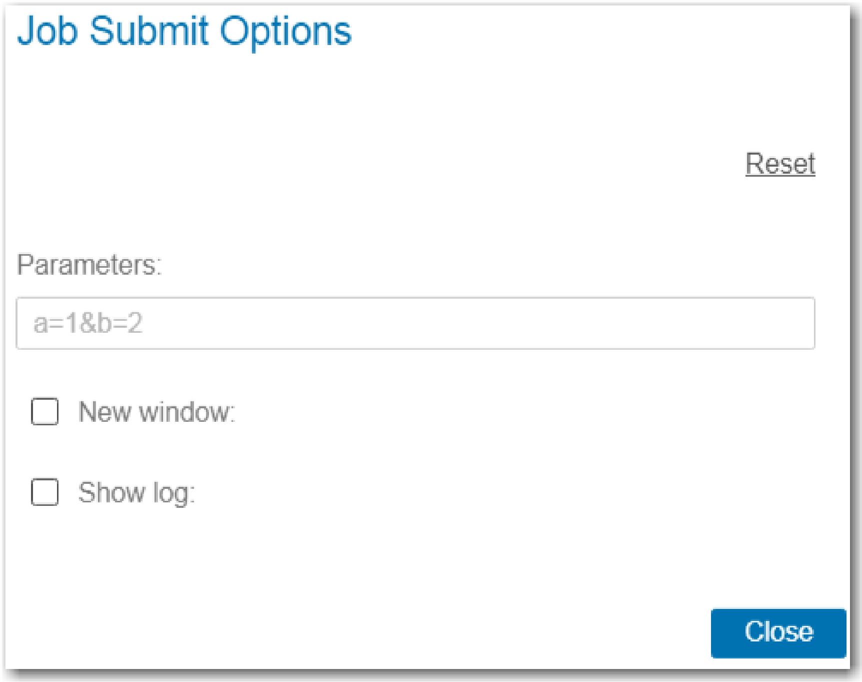
The full version of the SAS Content page is available to users who have authorization to develop jobs. It can be accessed using the `/developer` path. A more limited SAS Content page is available to users who are allowed only to execute existing jobs and view output. It can be accessed using the `/user` path. 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 17 for more information about how to assign users access to developer or user views.



Click ▤ to view general properties of a job or to add parameters to the job. See “Passing User Input to a Job Using a Job Definition Parameter” on page 23 for more information about setting parameters.

Click ⟲ to retrieve the output from the most recent execution of the job, if it is still available. The output is displayed in the right pane.

Select a job and then click ▶ to submit a job for execution. The following dialog box with execution options appears when you click 🗋:



You can add parameters using the format `name1=value1&name2=value2`. Select the **New window** option to display the job output in a new browser window. The **Show log** option adds the `_debug=log` parameter setting to the request. These options remain in effect for job submissions during the current browser session.

The following two URL parameters can be used to alter the folder list for the SAS Content page:

Parameter	Description
`_folder=/SomeFolder`	Opens the display at the indicated folder path
`_path=/SomeFolder`	Makes the root of the display the indicated folder path and displays items only in this folder and its subfolders

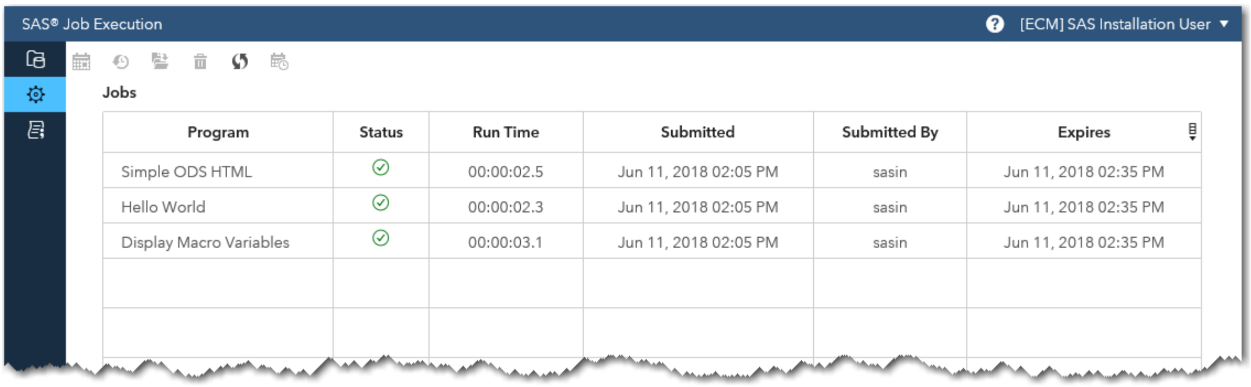
Here is a sample usage:

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

See "Accessing Application Pages through a URL" on page 6 for more information about accessing pages through a URL.

Managing Jobs

Click ⚙ to access the Jobs page, where you can display and maintain jobs that have run previously and have not yet expired.



Program	Status	Run Time	Submitted	Submitted By	Expires
Simple ODS HTML	✓	00:00:02.5	Jun 11, 2018 02:05 PM	sasin	Jun 11, 2018 02:35 PM
Hello World	✓	00:00:02.3	Jun 11, 2018 02:05 PM	sasin	Jun 11, 2018 02:35 PM
Display Macro Variables	✓	00:00:03.1	Jun 11, 2018 02:05 PM	sasin	Jun 11, 2018 02:35 PM

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 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.

SAS Environment Manager is used to schedule jobs. Click ▦ to open the Jobs page, where you can access the **Scheduling** tab. See *SAS Viya Administration: Jobs* for more information about scheduling. See "Scheduling a Job" on page 12 for more information about using the _ACTION parameter to schedule a job.

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.

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:

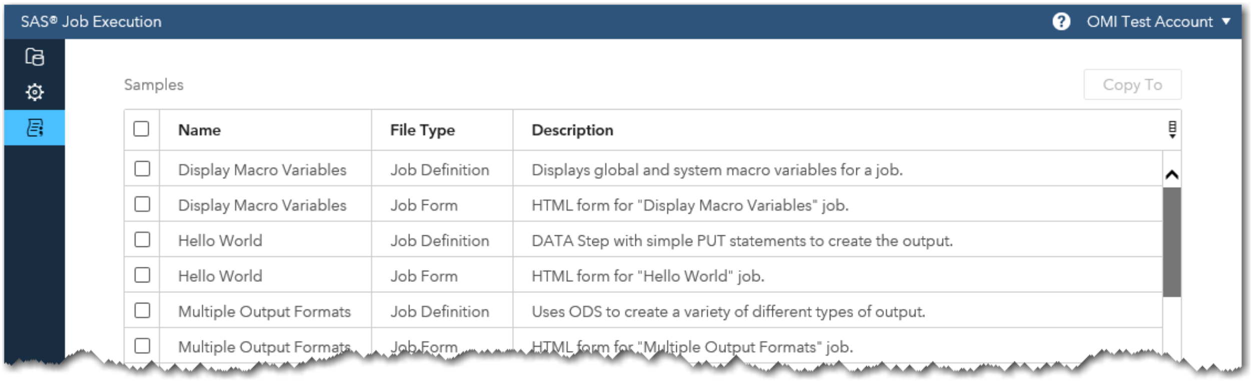
```
http://host:port/SASJobExecution/jobs?_program=/SomeFolder/Hello World
```

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

Installing the Samples

A set of sample jobs and HTML input forms 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.



Name	File Type	Description
Display Macro Variables	Job Definition	Displays global and system macro variables for a job.
Display Macro Variables	Job Form	HTML form for "Display Macro Variables" job.
Hello World	Job Definition	DATA Step with simple PUT statements to create the output.
Hello World	Job Form	HTML form for "Hello World" job.
Multiple Output Formats	Job Definition	Uses ODS to create a variety of different types of output.
Multiple Output Formats	Job Form	HTML form for "Multiple Output Formats" job.

To install samples and use them, copy the sample job definitions and HTML input forms 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 and HTML input forms 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 17 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:

Path	Page Description
`/developer`	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.
`/env`	Displays system environment information such as properties and parameters (restricted to SAS Administrators).
`/jobs`	Administers previous job runs.
`/logout`	Logs out of the application.
`/samples`	Administers sample jobs.

Path	Page Description
`/user`	Displays and executes job files. Depending on a user's authorization, this might be the default SAS Content page.

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 in the Folders service. 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 from the Job Definitions service.

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 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.

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`

Prompting for Input Parameters with HTML Input Forms

The location of the HTML input form file can be specified using the _FORM parameter. This 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
```

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 an HTML Input Form" on page 24 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 by the Files service 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, 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:

_RESULTFILE Value	Returns
*	All files

_RESULTFILE Value	Returns
`*.csv`	Only CSV files
`*.html,*.htm`	HTML and HTM files, in that order
`*.htm*,*.pdf,*.csv`	HTM, HTML, CSV, and PDF files, in that order
`state??.csv`	CSV files beginning with `state` and ending with any two characters (for example, state01.csv)
`retail.htm,retail.pdf,class.htm,class.pdf`	Only the files that are listed in the order in which they are listed

Use `_action=json` if you have an application that requires a list of result files in JSON format. Here is an example:

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

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

```
[
  {
    "name": "_webout.htm",
    "href": "/files/files/a680029d-da79-4d1a-83be-2440c6e6b89e/content"
  }
]
```

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, including the SAS log:

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

The following JSON is returned:

```
[
  {
    "name": "7BC34A5D-42BD-FD4C-AAF1-87D3A55AB8D9.log.txt",
    "href": "/files/files/bff66afe-eaf3-4af1-8261-b64ec9b72198/content"
  },
  {
    "name": "Class.pdf",
    "href": "/files/files/7ffad6f2-f6f4-4093-af4e-0548c4dfe8b0/content"
  },
  {
    "name": "Class.xml",
    "href": "/files/files/1987b95d-196c-4fad-9590-7afc2f6e8be3/content"
  },
  {
    "name": "_webout.htm",
    "href": "/files/files/6ded3921-0ae7-4e97-bf97-a654df996d27/content"
  }
]
```

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 "Sending ODS Output to an Email Recipient" on page 33 for an example of how to use _STATUS_MESSAGE.

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 14 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 HTML Input Forms" on page 7 for more information about how to use it to display HTML input forms. The `prompts` and `nobanner` values are 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:

Parameter Name	Value
%opt-myopt	**linesize=max**

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

Parameter Name	Value
%opt-myopt1	**linesize=max**
%opt-myopt2	**nobyline**

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 120 seconds for a job to complete. You can change this for a request by specifying the _TIMEOUT job parameter in the URL:

```
_timeout=60
```

The default value can be set using the EXECUTETIMEOUT configuration parameter.

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, this specifies that the job expires within 60 minutes after execution has completed:

```
_expiration=PT60M
```

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

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 ⟲ in 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/5d21f17b-5f85-4a49-9c34-319174eb741f/state
```

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

```
/SASJobExecution/?_jobexec=/jobExecution/jobs/5d21f17b-5f85-4a49-9c34-319174eb741f
```

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 parameter 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 an HTML Input Form” on page 24 for more information about creating an HTML file.

2. Follow the instructions in “Setting Configuration Properties” on page 14 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 parameter to specify the text of the wait message.

Note: The **wait** value works only when _ACTION is being used as a URL parameter.

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 the job definition and parameter values in the Scheduling table in SAS Environment Manager:

```
_action=schedule
```

Use SAS Environment Manager to complete the steps needed to schedule the job. See *SAS Viya Administration: Jobs* for more information about scheduling.

See “Managing Jobs” on page 5 for more information about manually performing this action.

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 FILEREF in “Development Concepts” on page 19 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.htm:

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

See “Managing Jobs” on page 5 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:

Value	Information Returned
fields	List of all input parameters. Those listed in the Arguments section become SAS global macro variables.
log	SAS job log.
time	Total time in seconds to execute the job; incremental and total time are reported when used in conjunction with `trace`.
trace	Information for each step of the job.

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 17 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:

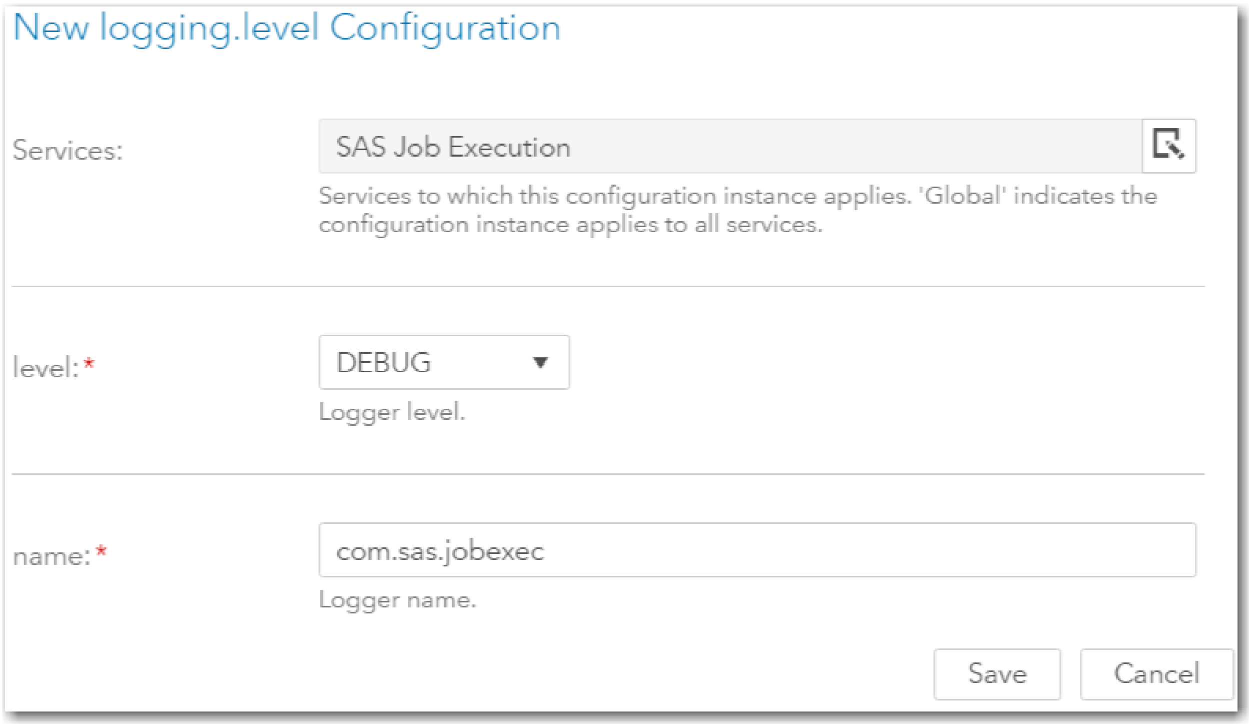
1 Click in SAS Environment Manager, and then select **Definitions** from the **View** drop-down list.

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**.


New logging.level Configuration

Services: SAS Job Execution

Services to which this configuration instance applies. 'Global' indicates the configuration instance applies to all services.

level:* DEBUG

Logger level.

name:* com.sas.jobexec

Logger name.

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 in the left pane. 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 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:

Property	Default Value	Description
actiondisallow		Comma-separated list of action values to disallow
background		SAS folder path and name of HTML file to display for background processing
contextname	SAS Job Execution compute context	Compute service context name

Property	Default Value	Description
debugdisallow		Comma-separated list of debug values to disallow
executetimeout	`120`	Job run time-out in seconds
expiration	`PT30M`	Job expiration duration (30 minutes) in the format defined in the W3C XML duration data type
indextitle	`SAS Job Execution`	Banner title for the application
maxfilecount	`5`	File upload maximum file count
maxfilesize	`100000000`	File upload maximum file size in bytes
waittext	`Please wait`	Message to display on job wait display
welcome		SAS folder path of HTML file to display if no parameters are specified

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 14 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 Properties**, and **HTTP Request Headers** sections displayed by the System Environment page. The following table lists some common request properties:

Recommended SAS Variable Name	Request Property Name	Description
_AUTHTYP	jobexec.auth.type	Name of the authentication scheme that is used (for example, BASIC, SSL, or blank if no protection).
	jobexec.character.encoding	Name of the character encoding that is used in the body of the request.
	jobexec.content.length	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.
	jobexec.content.type	MIME type of the body of the request. If the type is not known, the value is blank.
	jobexec.context.path	Portion of the request URL that indicates the context of the request.

Recommended SAS Variable Name	Request Property Name	Description
	jobexec.cookies	Cookie strings that the client sent with this request.
	jobexec.header	All HTTP request headers.
	jobexec.header.*name*	A particular HTTP request header line as it was received, where *name* is the header name.
_HTUA	jobexec.header.*user-agent*	A particular HTTP request header line as it was received, where *user-agent* is the name of the user agent.
	jobexec.jsessionid	Web application session ID.
_USERLOCALE	jobexec.locale	Preferred locale in which the client accepts content, based on the Accept-Language header.
	jobexec.method	HTTP method used when this request was made (for example, GET, POST, or PUT).
	jobexec.path	URL pathname.
	jobexec.protocol	Name and version of the protocol that the request uses in the form `protocol/majorVersion.minorVersion` (for example, HTTP/1.1).
	jobexec.query.string	Query string that is contained in the request URL after the path.
_RMTADDR	jobexec.remote.addr	Internet Protocol (IP) address of the client that sent the request.
_RMTHOST	jobexec.remote.host	Fully qualified name of the client that sent the request or the IP address of the client if the name cannot be determined.
	jobexec.request.inputencoding	Request input encoding. The default encoding is UTF-8.
	jobexec.scheme	Name of the scheme that was used to make this request (for example, HTTP, HTTPS, or FTP).
	jobexec.secure	A value of true or false, indicating whether this request was made using a secure channel, such as HTTPS.
	jobexec.user	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.

Recommended SAS Variable Name	Request Property Name	Description
_URL	jobexec.uri	Part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.
_VERSION	jobexec.version	SAS Job Execution Web Application version.
_XFORWARD	jobexec.header.x-forwarded-host	Request HTTP header value for *x-forwarded-host.*
_CLIENTNAME	SASJobExecution	Name of the client.

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:

Macro Variable Name	Value	Description
_SOME_TEXT	*This is some text*	Hardcoded text
_JAVA_VERSION	**`$java.version`**	Version of Java on the Java Application Server
_SCHEME	**`$jobexec.scheme`**	Name of the scheme for the request
_HTTP_HOST	**`$jobexec.header.host`**	Host and port number for the request

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.

Security for SAS Viya Jobs

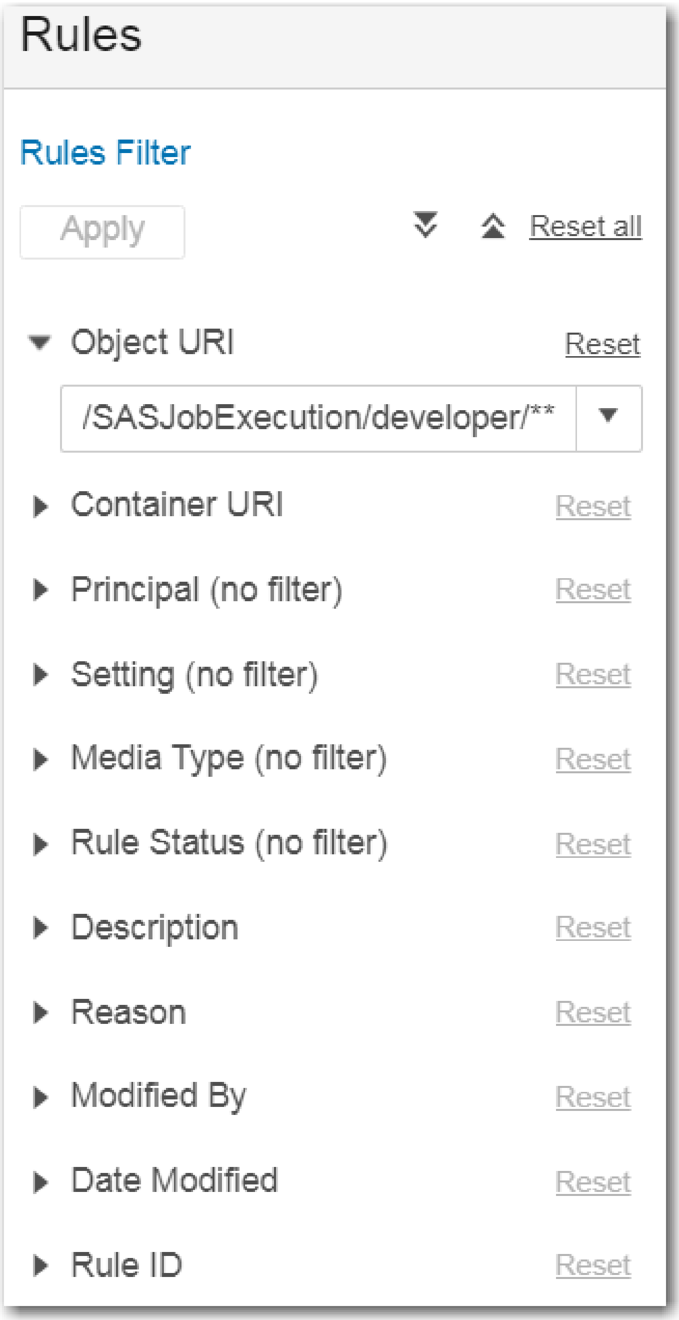
Setting Authorization for a Folder

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:


Object URI: /SASJobExecution/developer/**

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:

Function	Object URI	Default Principal
Web application root	/SASJobExecution/	Authenticated Users
Administration interface	/SASJobExecution/developer/**	Authenticated Users
_DEBUG job parameter	/SASJobExecution/debug/**	Authenticated Users
Display system environment	/SASJobExecution/env/**	SASAdministrators
Display an HTML input form before the job executes	/SASJobExecution/form/**	Authenticated Users
Standard interface to display and execute jobs	/SASJobExecution/user/**	Authenticated Users
Administration of previous job runs	/SASJobExecution/jobs/**	Authenticated Users
Administer sample jobs	/SASJobExecution/samples/**	Authenticated Users

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 42 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 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 42 and "%JESBEGIN and %JESEND AutoCall Macros" on page 84 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. The FILENAME statement shown in "Assigning a FILEREF for HTML Output" on page 33 is used.

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

`http://`*`host:port`*`/SASJobExecution`

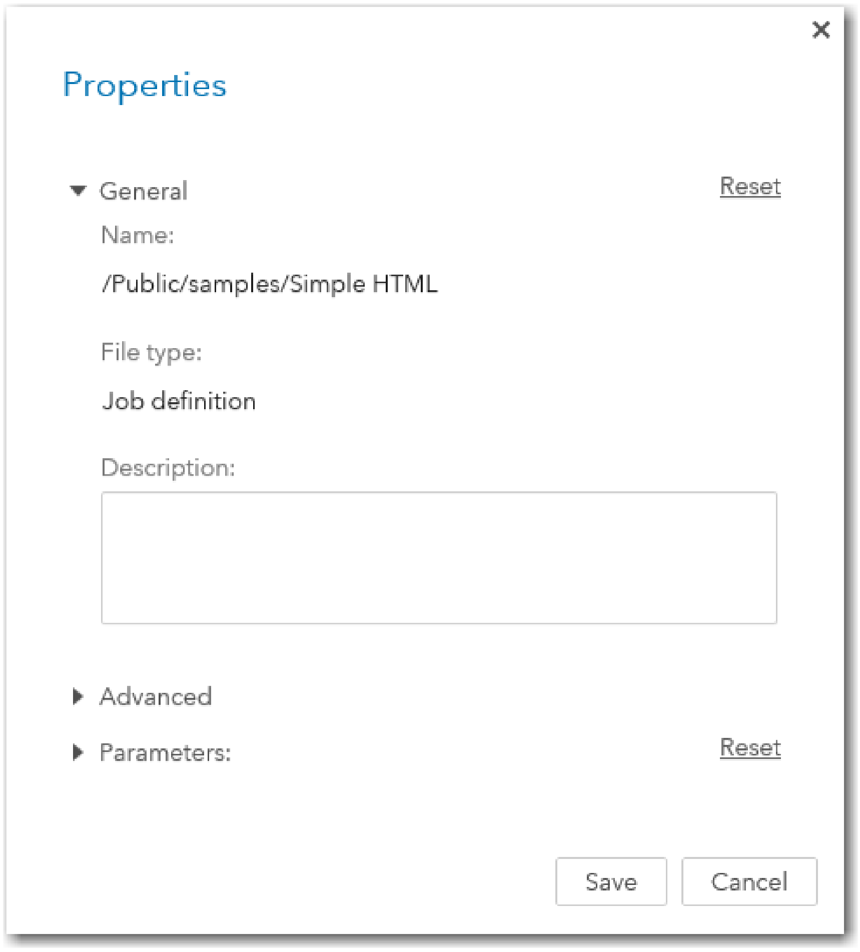
Note: You must have access to developer functionality to create jobs.

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 values for the **File type** field. Click **OK**.

3 Click the **Simple HTML** job and then click to display the Properties dialog box.



4 Expand the **Parameters** group and then click **Add a new parameter**. Specify the following properties for the parameter:

Property	Value
Name	`_OUTPUT_TYPE`
Field Type	`Character`
Default Value	`html`

Click **Save**.

This job creates HTML output but not HTML output generated by ODS. See "Creating Simple HTML Output Using ODS" on page 26 for more information about creating ODS output from a job.

5 Click .

6 Enter the following code into the editor window:

```
*  Write the custom HTML to _webout;

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 19 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 20 is:

```
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 ☰ to display the Properties dialog box.

2 Expand the **Advanced** group 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:

```
<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.)

Use the content selector pane to 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 22 is an example of an input 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:

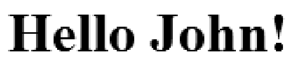
```
http://host:port/SASJobExecution/?_program=/SomeFolder1/Simple HTML&myname=John
```

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 20:

```
*  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:



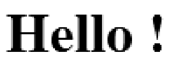
Hello John!

Passing User Input to a Job Using a Job Definition Parameter

Run the Simple HTML job that was created in the previous section 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 .

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

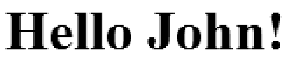
2 Expand the **Parameters** group and then click **Add a new parameter**. Specify the following properties for the parameter:

Property	Value
Name	`myname`
Field Type	`Character`
Default Value	`John`

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 output is now:



```
Hello John!
```

Passing User Input to a Job Using an HTML Input Form

You can use the SAS Content page to create HTML input forms 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 "Passing User Input to a Job Using the Query String" on page 23. The example creates an 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 input forms.

When you create an input form with the same name as the job and 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:

Property	Value
Name	`_ACTION`
Field Type	`Character`
Default Value	`form,execute`

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

Perform the following steps to create a job form named Simple HTML in the same folder as the job:

1 Navigate to the specified location and then click .

2 Specify **`Simple HTML`** for the name of the file. Select **Job form** from the **File type** drop-down list. Click **OK** to create the file.

3 Click the Simple HTML form file that you just created and then click .

4 Enter the following HTML code into the editor window, ensuring that you specify the appropriate folder location in the _PROGRAM attribute:

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

<h1>Simple HTML Example</h1>

<form action="/SASJobExecution/" target="_blank">
  <input type="hidden" name="_program" value="/SomeFolder/Simple HTML">
  <input type="hidden" name="_action"  value="execute">

<label>Specify a name for the greeting: </label>
<input type="text" name="myname" value="World" required>

<br/>
<br/>

<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 `_blank` in the TARGET attribute to force the output to always appear in a new browser window or tab. 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 **`/SomeFolder/Simple HTML`**. This input tag indicates the location and name of the program to execute. The value that you specify is case-sensitive.

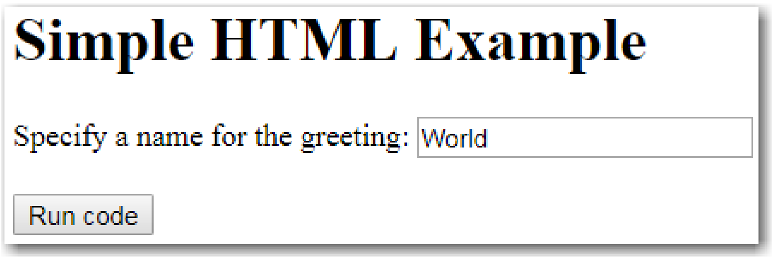
The second input tag specifies a value of **`execute`** for the _ACTION parameter, which overrides the default value specified in the job parameter and executes the job.

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 , and then click **Close** to save the input form and close the editor window.

5 Click the Simple HTML job to select it, and then click ►. The following HTML input form appears in the right pane:


Simple HTML Example

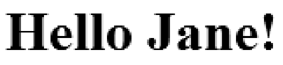
Specify a name for the greeting: World

6 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 23. 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.

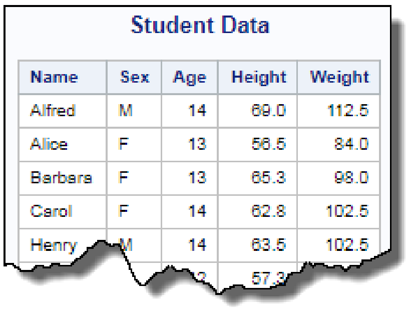
Many jobs create ODS HTML as their primary type of output. The %JESBEGIN macro can issue an ODS statement in addition to the _WEBOUT FILEREF. Add the following parameters to the job to use the ODS HTML5 destination and the HTMLBlue style:

Property	Value
Name	**_ODSSTYLE**
Field Type	**Character**
Default Value	**HTMLBlue**

This job uses PROC PRINT to display all of the data in the SASHELP.CLASS table:

```
*  Display the SASHELP.CLASS table;

title 'Student Data';

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

Here is a partial view of the output:



Student Data

Name	Sex	Age	Height	Weight
Alfred	M	14	69.0	112.5
Alice	F	13	56.5	84.0
Barbara	F	13	65.3	98.0
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
		2	57.3	

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 26 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.

The example uses a multiple selection list box to choose the columns to display from the SASHELP.CLASS table. Use the SAS Content page of the SAS Job Execution Web Application to create a job form named **`Multiple Input Values`** with the following HTML:

```
<!DOCTYPE html>
<head>
<title>Multiple Input Values Example</title>
</head>
<body>

<h1>Multiple Input Values Example</h1>

<form action="/SASJobExecution/" target="_blank">
<input type="hidden" name="_program" value="/SomeFolder/Multiple Input Values">
<input type="hidden" name="_action" value="execute">

<div>Use Ctrl+Click to choose columns to display: </div>

<br/>

<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>

<br/>
<br/>
```

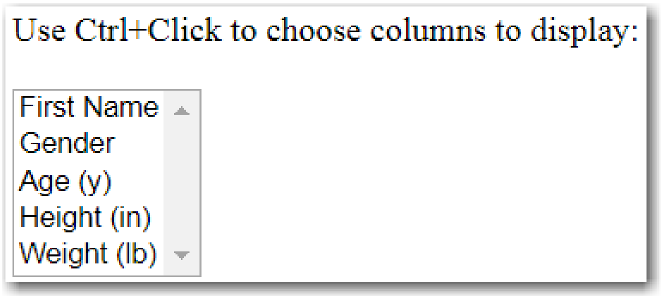
```
<input type="submit" value="Run code">
</form>

</body>
</html>
```

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:

Display Value	Macro Variable Value
First Name	`name`
Gender	`sex`
Age (y)	`age`
Height (in)	`height`
Weight (lb)	`weight`

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:

Macro Variable Name	Macro Variable Value	Description
COLS	`name`	Specifies the first value
COLS0	`3`	Specifies the number of values
COLS1	`name`	Specifies the first value
COLS2	`age`	Specifies the second value
COLS3	`weight`	Specifies the third value
COLS_COUNT	`3`	Specifies the number of values

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-name*0 and *param-name*_COUNT variables. The first value is set in the *param-name*1 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 90 to convert the user selections into a usable format.

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

Name	Field Type	Default Value
_ACTION	Character	`form, execute`
_OUTPUT_TYPE	Character	`ods_html5`
_ODSSTYLE	Character	`HTMLBlue`

Add the following code:

```
<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;
```

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, created by the PARAM_LIST macro, resolves to the following:

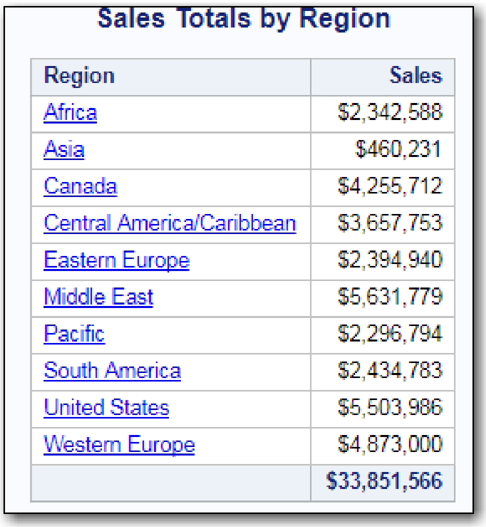
```
name age height
```

See "PARAM_LIST Macro" on page 90 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 to create the summarized information with the links and a second that displays the detailed data related to that link.

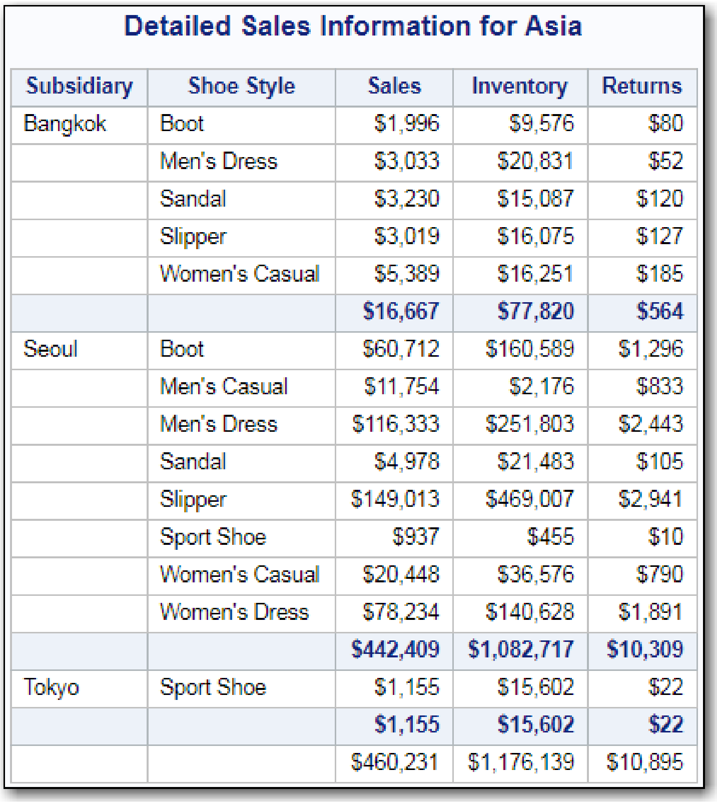
This example summarizes sales data in the SASHELP.SHOES table by sales region and then uses PROC PRINT to display it with links to the detailed data:



Sales Totals by Region

Region	Sales
Africa	$2,342,588
Asia	$460,231
Canada	$4,255,712
Central America/Caribbean	$3,657,753
Eastern Europe	$2,394,940
Middle East	$5,631,779
Pacific	$2,296,794
South America	$2,434,783
United States	$5,503,986
Western Europe	$4,873,000
	$33,851,566

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



Detailed Sales Information for Asia

Subsidiary	Shoe Style	Sales	Inventory	Returns
Bangkok	Boot	$1,996	$9,576	$80
	Men's Dress	$3,033	$20,831	$52
	Sandal	$3,230	$15,087	$120
	Slipper	$3,019	$16,075	$127
	Women's Casual	$5,389	$16,251	$185
		$16,667	**$77,820**	**$564**
Seoul	Boot	$60,712	$160,589	$1,296
	Men's Casual	$11,754	$2,176	$833
	Men's Dress	$116,333	$251,803	$2,443
	Sandal	$4,978	$21,483	$105
	Slipper	$149,013	$469,007	$2,941
	Sport Shoe	$937	$455	$10
	Women's Casual	$20,448	$36,576	$790
	Women's Dress	$78,234	$140,628	$1,891
		$442,409	**$1,082,717**	**$10,309**
Tokyo	Sport Shoe	$1,155	$15,602	$22
		$1,155	**$15,602**	**$22**
		$460,231	$1,176,139	$10,895

Create a job named Drilldown with the following job parameters:

Name	Field Type	Default Value
_OUTPUT_TYPE	Character	`ods_html5`
_ODSSTYLE	Character	`HTMLBlue`

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

```
*  Summarize the data;

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;

%let BASE_URL=&_URL.?_program=
   /SomeFolder/Drilldown2&_action=wait%nrstr(&region=);

*  Set the ODS escape character;

ods escapechar='^';

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

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:

```
^{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%2FCaribbean']
  Central America/Caribbean}
```

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

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

Name	Field Type	Default Value
_OUTPUT_TYPE	Character	**ods_html5**
_ODSSTYLE	Character	**HTMLBlue**

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, 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 `_ADDJESBEGINENDMACROS=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 named Email Report where the _OUTPUT_TYPE job 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 77 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 11 for more information about how to change the expiration time. See "Saving Job Output" on page 12 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 77 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 77 for an example that uses this format.

Advanced Programming

Sending JSON Data to a Job Using an Input Parameter

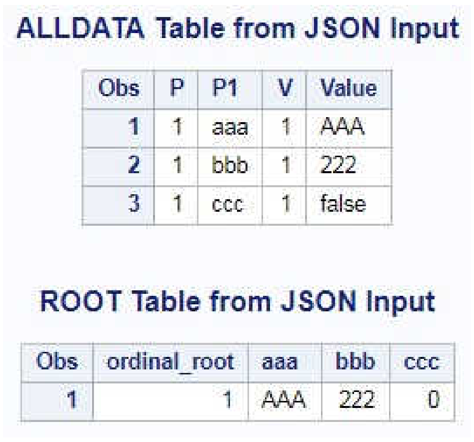
Data in JSON format can be sent to a job as an input parameter if the data contains fewer than 32,767 bytes. Data that is sent in this way can be accessed as a macro variable in the SAS job. In the following example, the MYJSON parameter contains JSON data that was sent to the JSON1 job. Note that %7B 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;

%global MYJSON;

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

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

Obs	P	P1	V	Value
1	1	aaa	1	AAA
2	1	bbb	1	222
3	1	ccc	1	false

ROOT Table from JSON Input

Obs	ordinal_root	aaa	bbb	ccc
1	1	AAA	222	0

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 “Upload a File” on page 68 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 35.

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:

```
<!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>

<!-- Other content of your web application here -->

<div id="JobResults"></div>

<script>submitForm();</script>

<!-- Other content of your web application here -->

</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 35 for the results and the SAS code used to process the data.

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:

```
<!DOCTYPE html>
<html>

<head>
<title>JavaScript Job Execution</title>

<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" , "$CSRF$");
 // 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>

<!-- 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 36. 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 82) 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:

```
<!DOCTYPE html>
<html>

<head>
<title>JavaScript Job Execution</title>

<script>
function submitForm() {
 var formData = new FormData();
 // Create input parameters
 formData.append("_program", "/Folder/Simple JSON");
 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) {
    // 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 77) 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, discussed in “Specifying Output Files” on page 8, to return a list of output files in JSON format:

```
[
 {
 "name": "Class.pdf",
 "href": "/files/files/3dba9d66-106b-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-8193-6e408dfa0a4f/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:

```
<!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); }
  }
 };
 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/ 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;

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

* FILEREFs for the response and the response headers;

filename resp     temp;
filename resp_hdr temp;

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;

libname resp json;

title 'Job Definitions';

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 is displayed in the right pane, and the output is displayed in a new browser tab or window.

Items Common to Most Samples

Many samples use the same or similar job input 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 Input Parameters

All sample jobs include the _ACTION input parameter to display their respective HTML input form. 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 22. The following table contains commonly used parameters:

Name	Value	Description
_ACTION	`form, execute`	Displays the HTML input form if one is available; otherwise, executes the job
_OUTPUT_TYPE	varies	Specifies the type of output created by the job, which can be `none`
_ODSSTYLE	`HTMLBlue`	Specifies the name of the ODS style if the job creates ODS output

%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.

Cascading Style Sheet Code in HTML Input Forms

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

```
<style type="text/css">

.pointer {
  cursor: pointer;
}
```

```
[Other Cascading Style Sheet code here]

</style>
```

The **`pointer`** class displays the web browser's pointer cursor when positioned over certain fields. You can provide your own CSS code or omit the STYLE element if you do not want to use CSS in your HTML input forms.

HTML Attributes to Support Accessibility

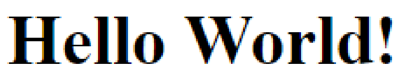
The HTML in the job forms that are included with the samples includes the following attributes, which support US government Section 508 accessibility standards:

```
<html lang="en">

<body role="main">

< ... for="element-ID" ... />

< ... aria-label="label-text" ... />

< ... aria-labelledby="checkboxfields element-ID" ... />
```

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

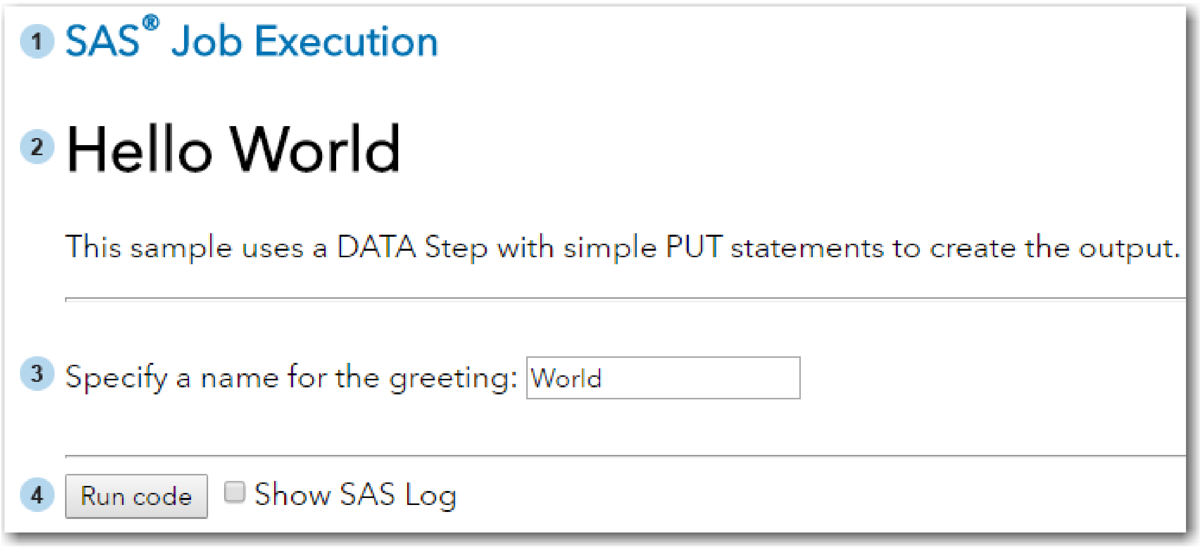


```
Hello World!
```

Job Input Parameters

Name	Value	Description
_ACTION	**`form, execute`**	Displays the HTML input form before the job is executed
_OUTPUT_TYPE	**`html`**	Specifies that non-ODS HTML output is created by the job
myname	**`World`**	Specifies the default value used in the greeting

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

```
<!DOCTYPE html>
<html lang="en">

<head>
<title>Hello World</title>
<style type="text/css">

.pointer {
  cursor: pointer;
}

[Other Cascading Style Sheet code here]

</style>

</head>

<body role="main">

1 <div>SAS<sup>®</sup> Job Execution</div>

2 <h1>Hello World</h1>

<p>
This sample uses a DATA Step with simple PUT statements to create the output.
</p>

<hr/>

<br/>

<form action="/SASJobExecution/" target="_SASResults">
<input type="hidden" name="_program" value="/Folder/Hello World"/>
<input type="hidden" name="_action" value="wait"/><input type="hidden" name="_output_type" value="html"/>
```

```
3 <label for="myname">Specify a name for the greeting:</label>
<input type="text" name="myname" id="myname" value="World" required/>

<br/>
<br/>

<hr/>

4 <input type="submit" value="Run code" class="pointer"/>

<input type="checkbox" name="_debug" id="_debug" value="log" class="pointer"/>
<label for="_debug">Show SAS Log</label>

</form>

</body>
</html>
```

1 The DIV tag displays a page heading.

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

3 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**.

4 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. A pointer cursor is displayed when positioned over these fields.

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.

As the name implies, HIDDEN elements are not displayed on screen, but they pass data to the application specified in the ACTION attribute. For the value of _PROGRAM, specify the path and name of the program to execute. 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 specifies 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:

Macro Variable Name	Macro Variable Value
_ODSDEST	blank
_OUTPUT_TYPE	**html**
_PROGRAM	*/Folder/***Hello World**
myname	**World**
_DEBUG	**log** (if the check box is selected)

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.

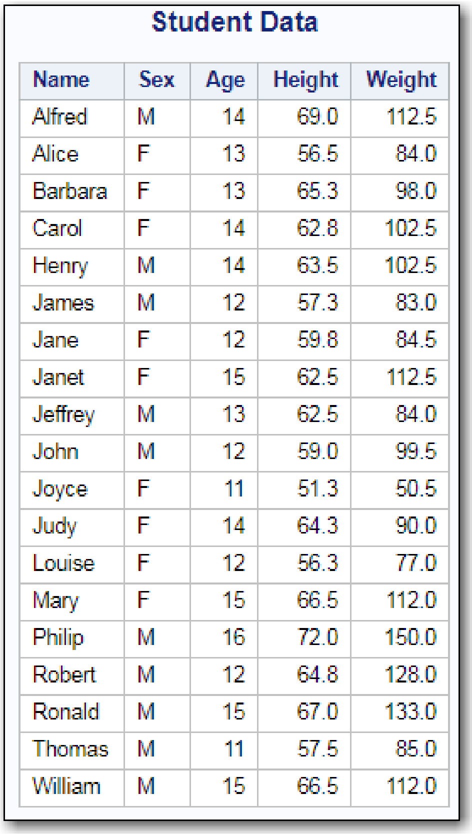
The DATA step code writes simple HTML to the _WEBOUT FILEREF that is assigned by the %JESBEGIN macro, and that HTML is rendered and displayed by the web browser. The value of the MYNAME macro variable used in the PUT statement is obtained from the value that was specified in the HTML input form. The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

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



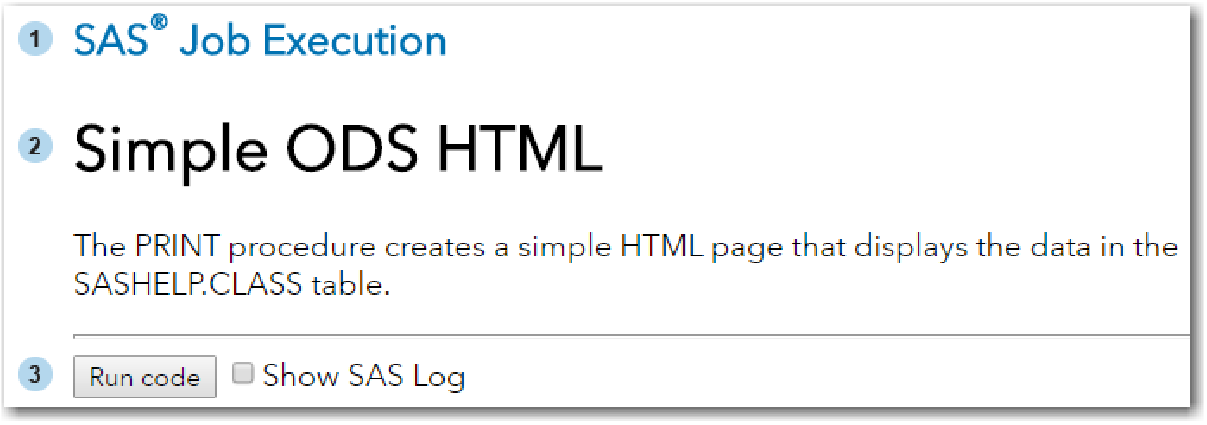
Student Data

Name	Sex	Age	Height	Weight
Alfred	M	14	69.0	112.5
Alice	F	13	56.5	84.0
Barbara	F	13	65.3	98.0
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83.0
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84.0
John	M	12	59.0	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90.0
Louise	F	12	56.3	77.0
Mary	F	15	66.5	112.0
Philip	M	16	72.0	150.0
Robert	M	12	64.8	128.0
Ronald	M	15	67.0	133.0
Thomas	M	11	57.5	85.0
William	M	15	66.5	112.0

Job Input Parameters

Name	Value	Description
_ACTION	`form, execute`	Displays the HTML input form before the job is executed
_OUTPUT_TYPE	`ods_html5`	Specifies that ODS HTML5 output is created by the job
_ODSSTYLE	`HTMLBlue`	Specifies the name of the ODS style

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.

```
<!DOCTYPE html>
<html lang="en">

<head>
<title>Simple ODS HTML</title>
<style type="text/css">

.pointer {
  cursor: pointer;
}

[Other Cascading Style Sheet code here]

</style>

</head>

<body role="main">

1 <div>SAS<sup>®</sup> Job Execution</div>

2 <h1>Simple ODS HTML</h1>

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

<hr/>

<form action="/SASJobExecution/" target="_SASResults">
3 <input type="hidden" name="_program" value="/Folder/Simple ODS HTML"/>
<input type="hidden" name="_action" value="execute"/>
<input type="hidden" name="_output_type" value="ods_html5"/>

<input type="submit" value="Run code" class="pointer"/>
<input type="checkbox" name="_debug" id="_debug" value="log" class="pointer"/>
<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 For the value of _PROGRAM, specify the path and name of the program to execute.

  _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:

Macro Variable Name	Macro Variable Value
_ACTION	`execute`
_ODSDEST	`html5`
_ODSSTYLE	`HTMLBlue`
_OUTPUT_TYPE	`ods_html5`
_PROGRAM	*/Folder/*`Simple ODS HTML`
_DEBUG	`log` (if the check box is selected)

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

Program

```
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 ODS HTML5 destination is used because it is the most up-to-date destination for creating HTML output. 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.

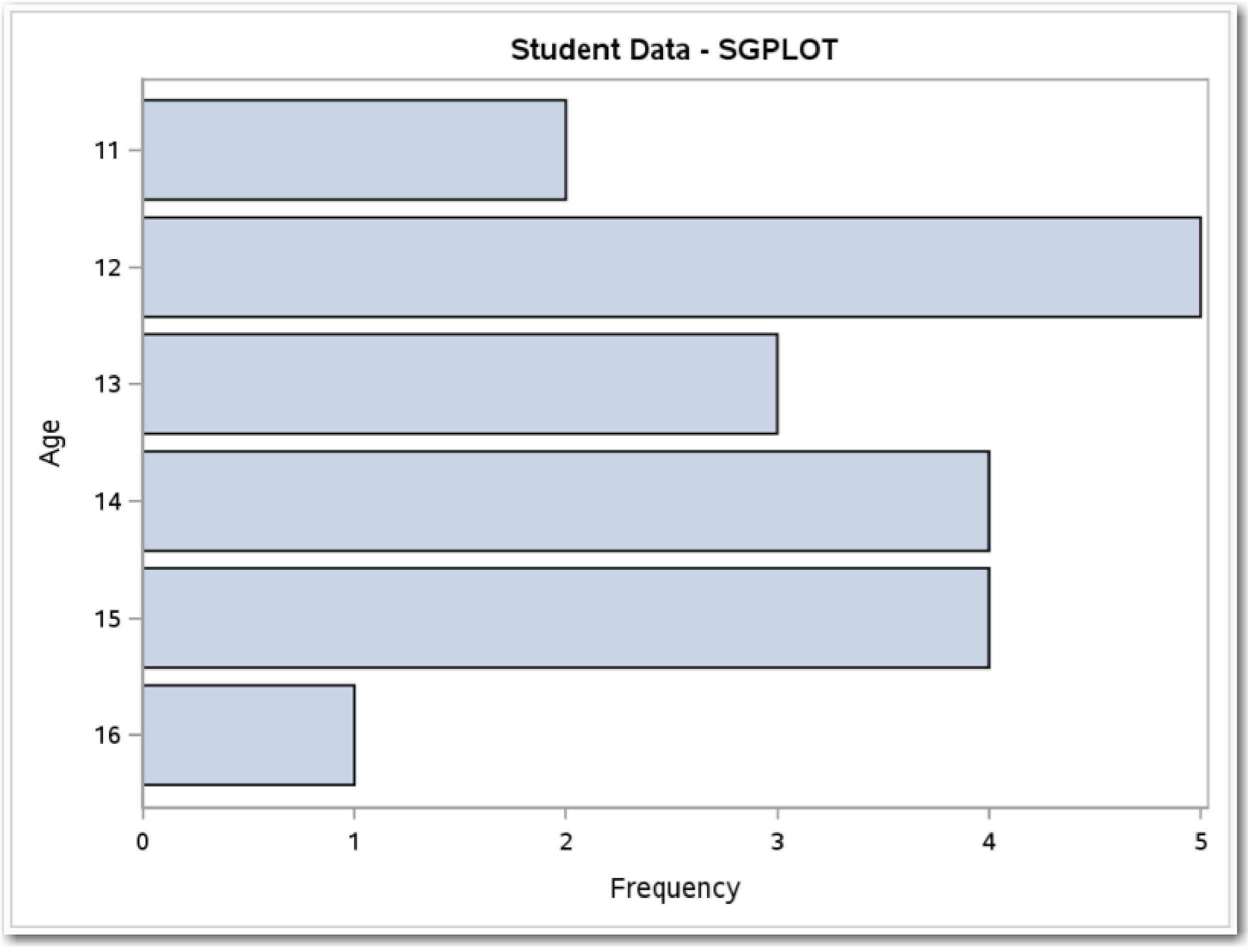
The PRINT procedure displays the SASHELP.CLASS table.

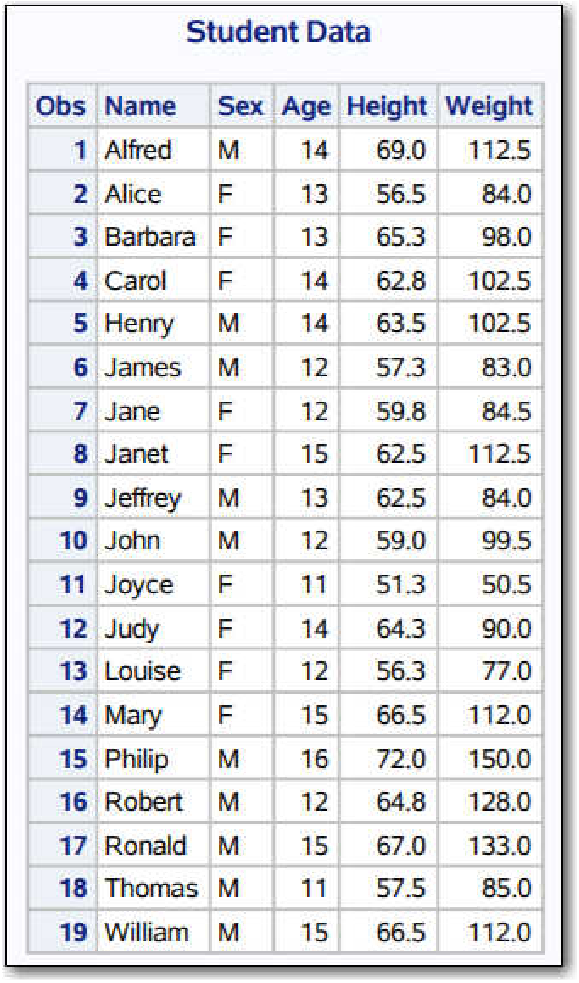
The %JESEND macro executes after the last line of code and closes all open ODS destinations.

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





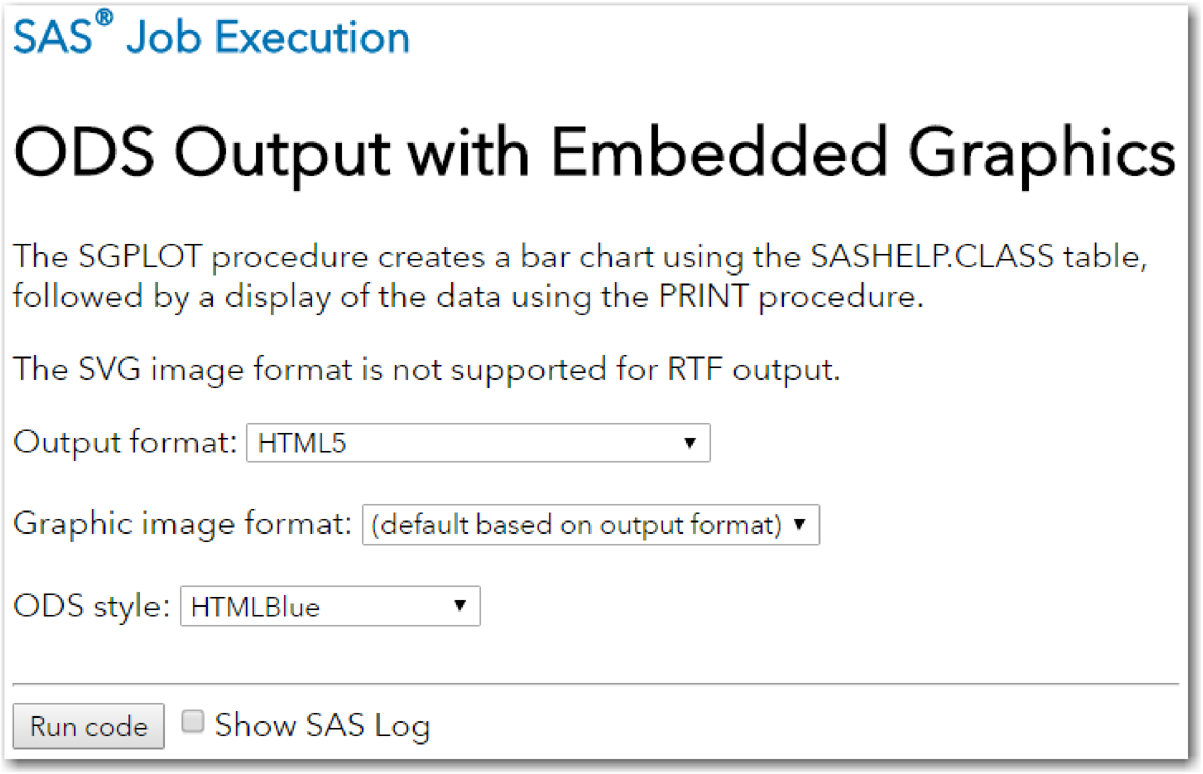
Student Data

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Job Input Parameters

Name	Value	Description
_ACTION	`form, execute`	Displays the HTML input form before the job is executed
_OUTPUT_TYPE	`ods_html5`	Specifies that ODS HTML5 output is created by the job
_ODSSTYLE	`HTMLBlue`	Specifies the name of the ODS style
_ODS_DEVICE	`png`	Specifies the ODS graphic image format

HTML Input Form



SAS® Job Execution

ODS Output with Embedded Graphics

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

```
<!DOCTYPE html>
<html lang="en">

<head>
<title>ODS Output with Embedded Graphics</title>
<style type="text/css">

.pointer {
  cursor: pointer;
}

[Other Cascading Style Sheet code here]

</style>

</head>

<body role="main">

<div>SAS<sup>®</sup> Job Execution</div>

<h1>ODS Output with Embedded Graphics</h1>

<p>
The SGPLOT procedure creates a bar chart using the SASHELP.CLASS
table, followed by a display of the data using the PRINT procedure.
</p>
```

```
<p>
The SVG image format is not supported for the RTF output.
</p>

<hr/>

<form action="/SASJobExecution/" target="_SASResults">
<input type="hidden" name="_program"
   value="/Folder/ODS Output with Embedded Graphics"/>

<input type="hidden" name="_action" value="execute"/>

<label for="_output_type">Output format:</label>
1 <select name="_output_type" id="_output_type" class="pointer">
  <option value="ods_html5">HTML5</option>
  <option value="ods_pdf">Portable Document Format (PDF) </option>
  <option value="ods_rtf">Rich Text Format (RTF) </option>
</select>

<br/>
<br/>

<label for="_ods_device">Graphic image format: </label>
2 <select name="_ods_device" id="_ods_device" class="pointer">
  <option value="" selected>(default based on output format)</option>
  <option value="svg">SVG</option>
  <option value="png">PNG</option>
</select>

<br/>
<br/>

<label for="_odsstyle">ODS style: </label>
3 <select name="_odsstyle" id="_odsstyle" class="pointer">
  [More values here]
  <option value="HTMLBlue" selected>HTMLBlue</option>
  [More values here]
</select>

<br/>
<br/>

<hr/>

<input type="submit" value="Run code" class="pointer"/>
<input type="checkbox" name="_debug" id="_debug" value="log" class="pointer"/>
<label for="_debug">Show SAS Log</label>

</form>

</body>

</html>
```

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

2 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.

3 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.

For the value of _PROGRAM, specify the path and name of the program to execute.

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

Display Value	**Macro Variable Value**
HTML5	**`ods_html5`**
Portable Document Format (PDF)	**`ods_pdf`**
Rich Text Format (RTF)	**`ods_rtf`**

The following table lists the display values for the second drop-down list and the values for the _ODS_DEVICE global macro variable:

Display Value	**Macro Variable Value**
(default based on the output format)	blank
SVG	**`svg`**
PNG	**`png`**

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**
_ACTION	**`execute`**
_ODS_DEVICE	Depends on selection (blank, **`svg`**, or **`png`**)
_ODSDEST	Depends on selection (**`html5`**, **`pdf`**, or **`rtf`**)
_ODSSTYLE	Depends on selection (for example, **`HTMLBlue`**)
_OUTPUT_TYPE	Depends on selection (**`ods_html5`**, **`ods_pdf`**, or **`ods_rtf`**)
_PROGRAM	*/Folder/*`ODS Output with Embedded Graphics`
_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.

The HTML in this form uses some of the same fields as "Simple ODS HTML" on page 46. See that section for more information.

Program

```
title 'Student Data - SGPLOT';
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. The HTML5 destination is used because it is the most up-to-date destination for creating HTML output. 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.

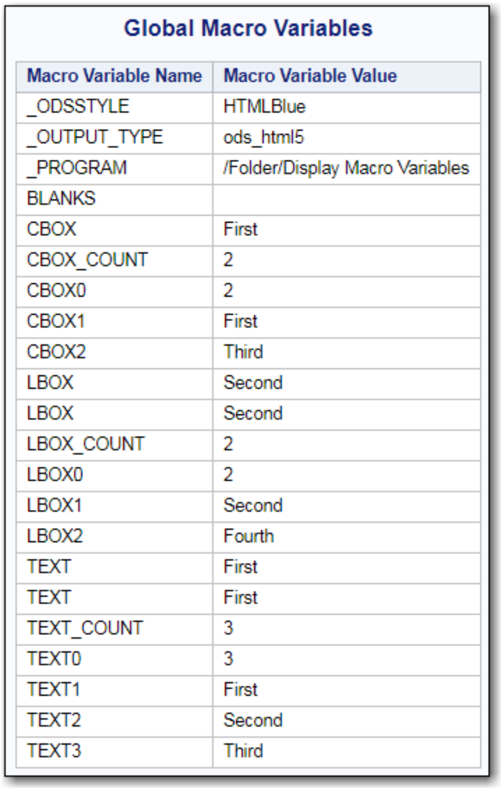
The HTML output and the PDF output are displayed by the web browser, and the RTF output is downloaded so that it can be saved and opened with an appropriate application, such as Microsoft Word.

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.



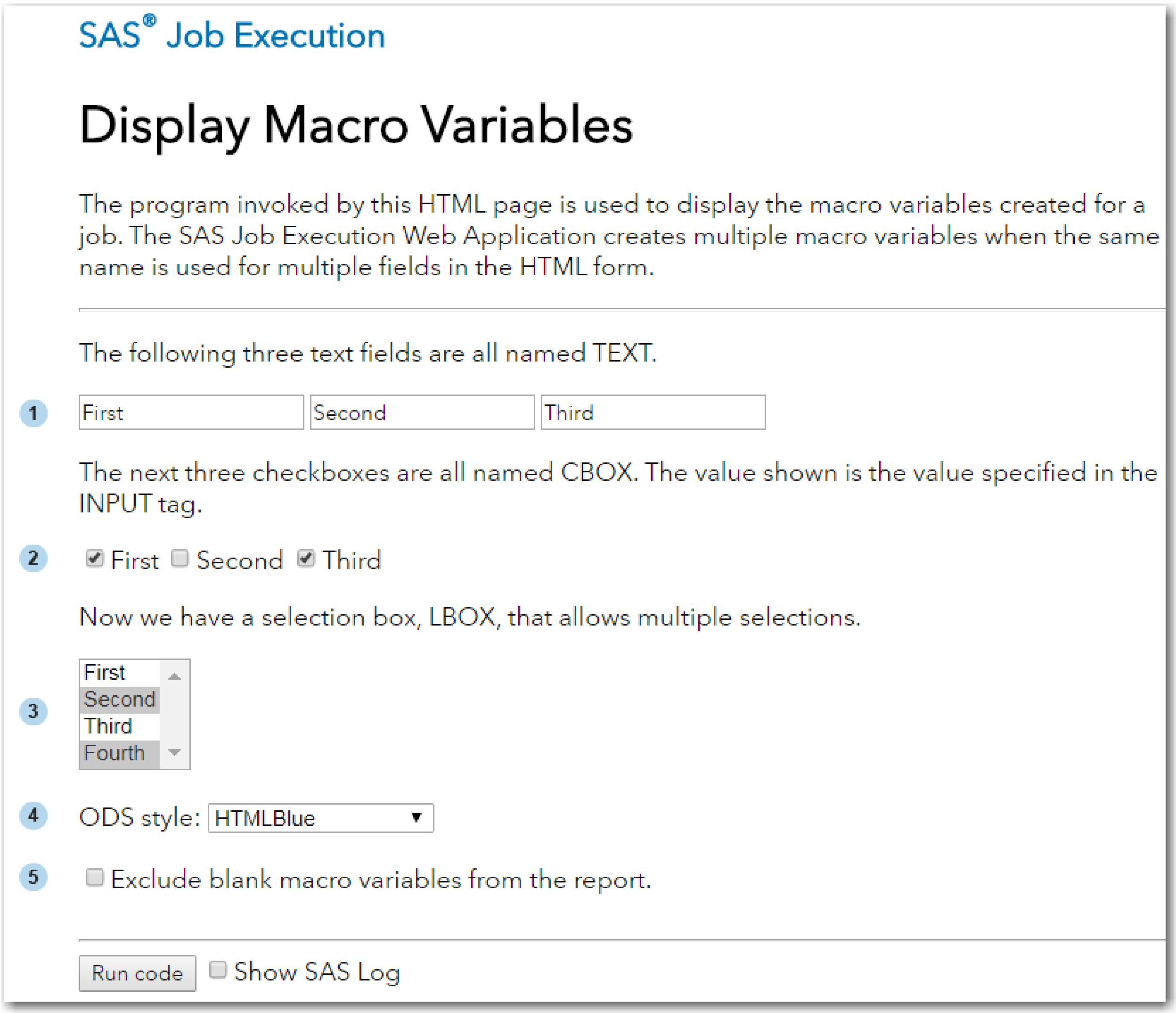
Global Macro Variables

Macro Variable Name	Macro Variable Value
_ODSSTYLE	HTMLBlue
_OUTPUT_TYPE	ods_html5
_PROGRAM	/Folder/Display Macro Variables
BLANKS	
CBOX	First
CBOX_COUNT	2
CBOX0	2
CBOX1	First
CBOX2	Third
LBOX	Second
LBOX	Second
LBOX_COUNT	2
LBOX0	2
LBOX1	Second
LBOX2	Fourth
TEXT	First
TEXT	First
TEXT_COUNT	3
TEXT0	3
TEXT1	First
TEXT2	Second
TEXT3	Third

Job Input Parameters

Name	Value	Description
_ACTION	`form, execute`	Displays the HTML input form before the job is executed
_OUTPUT_TYPE	`ods_html5`	Specifies that ODS HTML5 output is created by the job
_ODSSTYLE	`HTMLBlue`	Specifies the name of the ODS style

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.

1 First | Second | Third

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

2 [✓ First] Second [✓ Third]

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

3 First, [✓ Second], Third, [✓ Fourth]

4 ODS style: HTMLBlue

5 Exclude blank macro variables from the report.

```
<!DOCTYPE html>
<html lang="en">

<head>
<title>Display Macro Variables</title>
<style type="text/css">

.pointer {
  cursor: pointer;
}

[Other Cascading Style Sheet code here]

</style>

</head>

<body role="main">
```

```
<div>SAS<sup>®</sup> Job Execution</div>

<h1>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.
</p>

<hr/>

<form action="/SASJobExecution/" target="_SASResults">
<input type="hidden" name="_program" value="/Folder/Display Macro Variables"/>
<input type="hidden" name="_action" value="execute"/>
<input type="hidden" name="_output_type" value="ods_html5"/>

<p>
The following three text fields are all named TEXT.
</p>

1 <input type="text" name="text" value="First" aria-label="First text
field"/>
<input type="text" name="text" value="Second" aria-label="Second text
field"/>
<input type="text" name="text" value="Third" aria-label="Third text
field"/>

<p id="checkboxfields">
The next three checkboxes are all named CBOX. The value shown is the value
specified in the INPUT tag.
</p>

2 <input type="checkbox" name="cbox" id="cbox1" value="First" checked
  class="pointer" aria-labelledby="checkboxfields cbox1"/>
<label for="cbox1">First</label>

<input type="checkbox" name="cbox" id="cbox2" value="Second"
  class="pointer" aria-labelledby="checkboxfields cbox2"/>
<label for="cbox2">Second</label>

<input type="checkbox" name="cbox" id="cbox3" value="Third" checked
  class="pointer" aria-labelledby="checkboxfields cbox3"/>
<label for="cbox3">Third</label>

<p>
Now we have a selection box, LBOX, that allows multiple selections.
</p>

3 <select name="lbox" multiple class="pointer" aria-label="Selection box named LBOX">
  <option value="First">First</option>
  <option value="Second" selected>Second</option>
  <option value="Third">Third</option>
  <option value="Fourth" selected>Fourth</option>
</select>
```

```
<br/>
<br/>

4 <label for="_odsstyle">ODS style:</label>
<select name="_odsstyle" id="_odsstyle" class="pointer">
  [More values here]
  <option value="HTMLBlue" selected>HTMLBlue</option>
  [More values here]
</select>

<br/>
<br/>

5 <input type="checkbox" name="blanks" id="blanks" value="ExcludeBlanks"
  class="pointer"/>
<label for="blanks">Exclude blank macro variables from the report.</label>

<br/>
<br/>

<hr/>

<input type="submit" value="Run code" class="pointer"/>
<input type="checkbox" name="_debug" id="_debug" value="log" class="pointer"/>
<label for="_debug">Show SAS Log</label>

</form>

</body>

</html>
```

1 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 27. The output shows the values for the macro variables created from these elements when the default values are selected and the form is submitted.

2 This element enables you to specify multiple values.

3 This element enables you to specify multiple values.

4 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.

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

For the value of _PROGRAM, specify the path and name of the program to execute.

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 50. See that section for more information.

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;

title 'Global Macro Variables';
footnote;

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

title 'Automatic (System) Macro Variables';
footnote;
```

```
proc print data=work.systemvars label noobs
  style(header)=[just = center]
  style(column)=[protectspecialchars=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 ODS HTML5 destination is used because it is the most up-to-date destination for creating HTML output. 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.

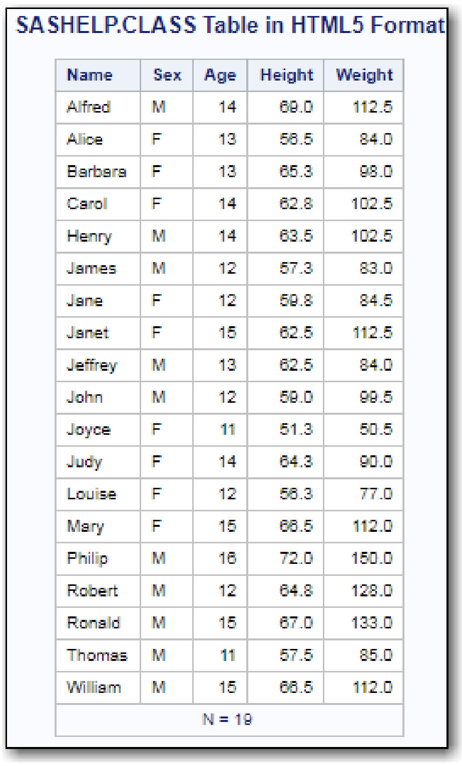
The %JESEND macro executes after the last line of code and closes all open ODS destinations.

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.



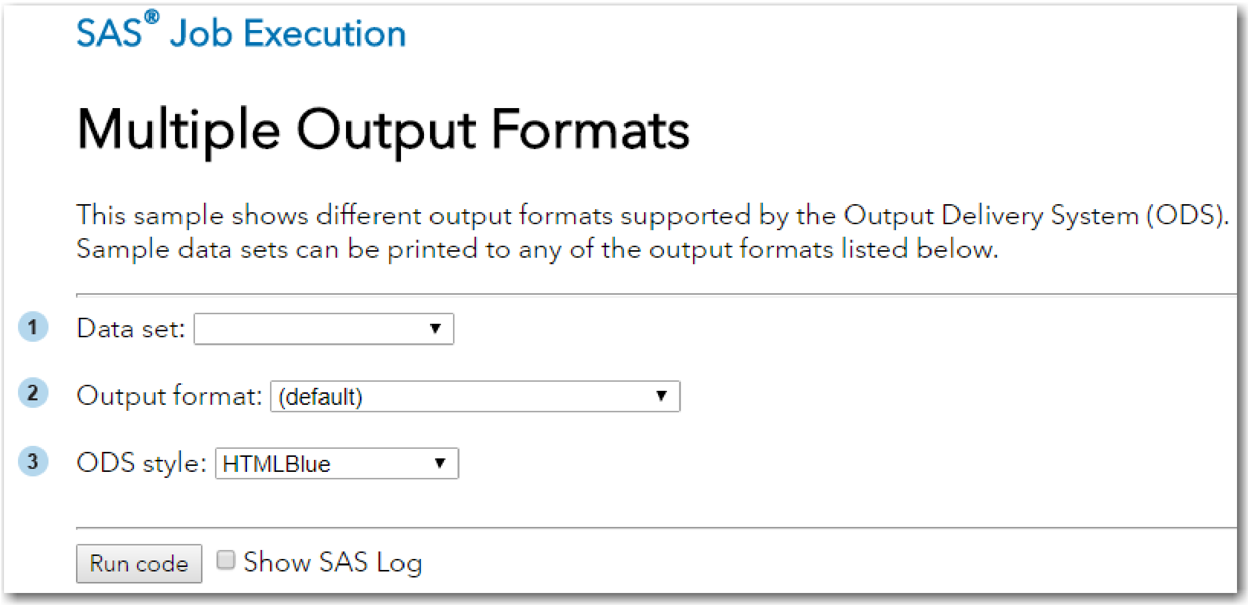
SASHELP.CLASS Table in HTML5 Format

Name	Sex	Age	Height	Weight
Alfred	M	14	69.0	112.5
Alice	F	13	56.5	84.0
Barbara	F	13	65.3	98.0
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83.0
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84.0
John	M	12	59.0	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90.0
Louise	F	12	56.3	77.0
Mary	F	15	66.5	112.0
Philip	M	16	72.0	150.0
Robert	M	12	64.8	128.0
Ronald	M	15	67.0	133.0
Thomas	M	11	57.5	85.0
William	M	15	66.5	112.0
N = 19				

Job Input Parameters

Name	Value	Description
_ACTION	`form, execute`	Displays the HTML input form before the job is executed
_OUTPUT_TYPE	`ods_html5`	Specifies that ODS HTML5 output is created by the job
_ODSSTYLE	`HTMLBlue`	Specifies the name of the ODS style
DATASET	`SASHELP.CLASS`	Specifies the SAS table to display

HTML Input Form


SAS® Job Execution
Multiple Output Formats
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.
1
Data set:
2
Output format: (default)
3
ODS style: HTMLBlue
Run code
Show SAS Log

```
<!DOCTYPE html>
<html lang="en">

<head>
<title>Multiple Output Formats</title>
<style type="text/css">

.pointer {
  cursor: pointer;
}

[Other Cascading Style Sheet code here]

</style>

</head>

<body role="main">

<div>SAS<sup>®</sup> Job Execution</div>

<h1>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>

<hr/>


<form action="/SASJobExecution/" target="_SASResults">
```

```
<input type="hidden" name="_program" value="/Folder/Multiple Output
  Formats"/>

<input type="hidden" name="_action"  value="execute"/>

1 <label for="dataset">Data set:</label>
<select name="dataset" id="dataset">
  <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/>

2 <label for="_output_type">Output format:</label>
<select name="_output_type" id="_output_type">
  <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/>

3 <label for="_odsstyle">ODS style:</label>
<select name="_output_style" id="_odsstyle">
  [More values here]
  <option value="HTMLBlue" selected>HTMLBlue</option>
  [More values here]
</select>

<br/>
<br/>

<hr/>

<input type="submit" value="Run code" class="pointer"/>
<input type="checkbox" name="_debug" id="_debug" value="log" class="pointer"/>
<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 SAS program 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 HTMLBlue style is selected by default. The selected value is stored in the _ODSSTYLE global macro variable.

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

Display Value	**Macro Variable Value**
blank	blank
SASHELP.RETAIL	`sashelp.retail`
SASHELP.CLASS	`sashelp.class`
SASHELP.REVHUB2	`sashelp.revhub2`
does_not_exist	`does_not_exist`

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

Display Value	**Macro Variable Value**
(default)	
HTML	`ods_html`
HTML5	`ods_html5`
Portable Document Format (PDF)	`ods_pdf`
Rich Text Format (RTF)	`ods_rtf`
Comma-Separated Value (CSV)	`ods_csv`
Extensible Markup Language (XML)	`ods_xml`
RTF_Tagset	`ods_tagsets.rtf`
Excel (XML)	`ods_tagsets.excelxp`
PostScript (PS)	`ods_ps`

Display Value	Macro Variable Value
LaTeX	**ods_latex**

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:

Macro Variable Name	Macro Variable Value
_ACTION	**execute**
_ODSDEST	Depends on selection (for example, **tagsets.rtf**)
_ODSSTYLE	Depends on selection (for example, **HTMLBlue**)
_OUTPUT_TYPE	Depends on selection (see previous table)
_PROGRAM	/*Folder*/**Multiple Output Formats**
_DEBUG	**log** (if the check box is selected)
DATASET	Depends on selection (see previous 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 46. 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. The ODS HTML5 destination is used because it is the most up-to-date destination for creating HTML output.

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 DATSASET 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 33 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.

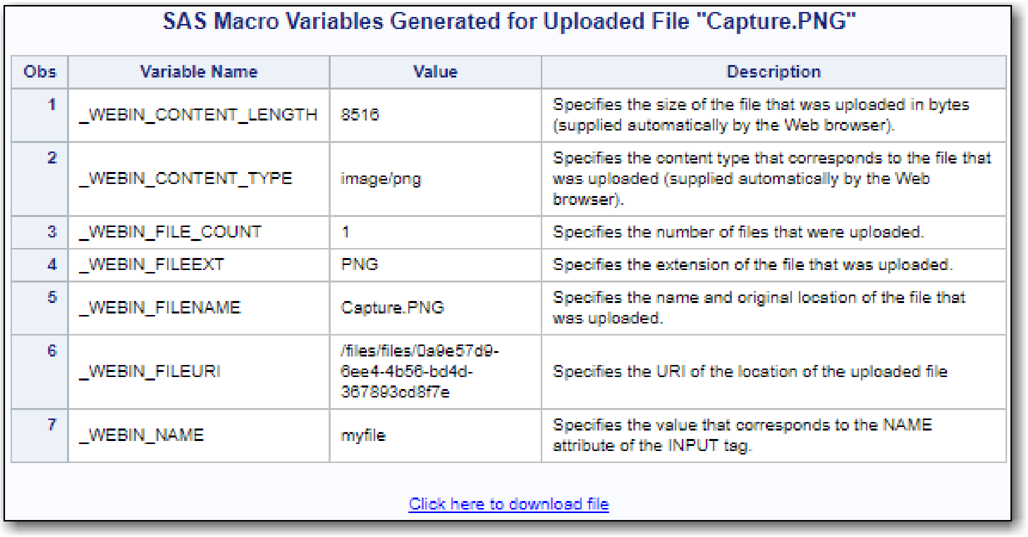
HTML and PDF output are displayed by the web browser, and all other output is downloaded so that it can be saved and opened with an appropriate application.

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.



SAS Macro Variables Generated for Uploaded File "Capture.PNG"

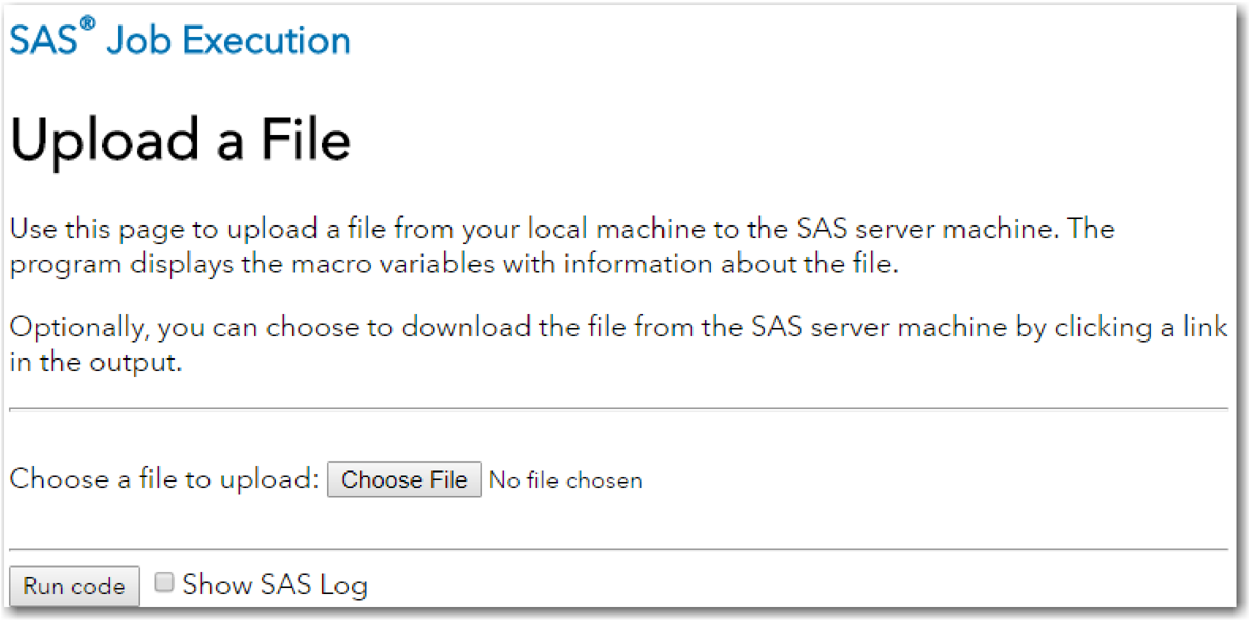
Obs	Variable Name	Value	Description
1	_WEBIN_CONTENT_LENGTH	8516	Specifies the size of the file that was uploaded in bytes (supplied automatically by the Web browser).
2	_WEBIN_CONTENT_TYPE	image/png	Specifies the content type that corresponds to the file that was uploaded (supplied automatically by the Web browser).
3	_WEBIN_FILE_COUNT	1	Specifies the number of files that were uploaded.
4	_WEBIN_FILEEXT	PNG	Specifies the extension of the file that was uploaded.
5	_WEBIN_FILENAME	Capture.PNG	Specifies the name and original location of the file that was uploaded.
6	_WEBIN_FILEURI	/files/files/0a9e57d9-6ee4-4b56-bd4d-367893cd8f7e	Specifies the URI of the location of the uploaded file
7	_WEBIN_NAME	myfile	Specifies the value that corresponds to the NAME attribute of the INPUT tag.

Click here to download file

Job Input Parameters

Name	Value	Description
_ACTION	`form, execute`	Displays the HTML input form before the job is executed
_OUTPUT_TYPE	`ods_html5`	Specifies that ODS HTML5 output is created by the job
_ODSSTYLE	`HTMLBlue`	Specifies the name of the ODS style

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.

```
<!DOCTYPE html>
<html lang="en">

<head>
<title>Upload a File</title>
<style type="text/css">

.pointer {
  cursor: pointer;
}

[Other Cascading Style Sheet code here]

</style>

</head>

<body role="main">

<div>SAS<sup>®</sup> Job Execution</div>

<h1>Upload a File</h1>

<p>
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.
</p>

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

```
<hr/>

<br/>

1 <form action="/SASJobExecution/" method="post" target="_SASResults"
  enctype="multipart/form-data">
<input type="hidden" name="_program" value="/Folder/Upload a File">
<input type="hidden" name="_action" value="execute"/>
<input type="hidden" name="_output_type" value="ods_html5"/>
2 <input type="hidden" name="_csrf" value="$CSRF$">

<label for="myfile">Choose a file to upload:</label>
3 <input type="file" name="myfile" id="myfile" required/>

<br/>
<br/>

<hr/>

<input type="submit" value="Run code" class="pointer"/>
<input type="checkbox" name="_debug" id="_debug" value="log" class="pointer"/>
<label for="_debug">Show SAS Log</label>

</form>

</body>

</html>
```

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

2 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.

3 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.

For the value of _PROGRAM, specify the path and name of the program to execute.

The HTML in this form has some of the same elements and features of the form used in "Simple ODS HTML" on page 46. 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**

Macro Variable Name	Macro Variable Value
_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;

varname     = '_WEBIN_FILENAME';
value       = resolve(symget('_WEBIN_FILENAME'));
description = 'Specifies the name and original location of the file that was uploaded.';
output;

varname     = '_WEBIN_FILEURI';
value       = resolve(symget('_WEBIN_FILEURI'));
description = 'Specifies the URI of the location of the uploaded file';
output;

varname     = '_WEBIN_NAME';
value       = resolve(symget('_WEBIN_NAME'));
description = 'Specifies the value that corresponds to the NAME attribute of the INPUT tag.';
```

```
output;

label varname     = 'Variable Name'
      value       = 'Value'
      description = 'Description';

run;

title 'SAS Macro Variables Generated for Uploaded File '
      """&_WEBIN_FILENAME""";
footnote link="&_WEBIN_FILEURI/content" 'Click here to download file';

proc print data=work.upload_info
  label
  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 ODS HTML5 destination is used because it is the most up-to-date destination for creating HTML output. 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 72 provides an example of referencing the uploaded file using a FILENAME statement.

The PRINT procedure displays information about the uploaded file.

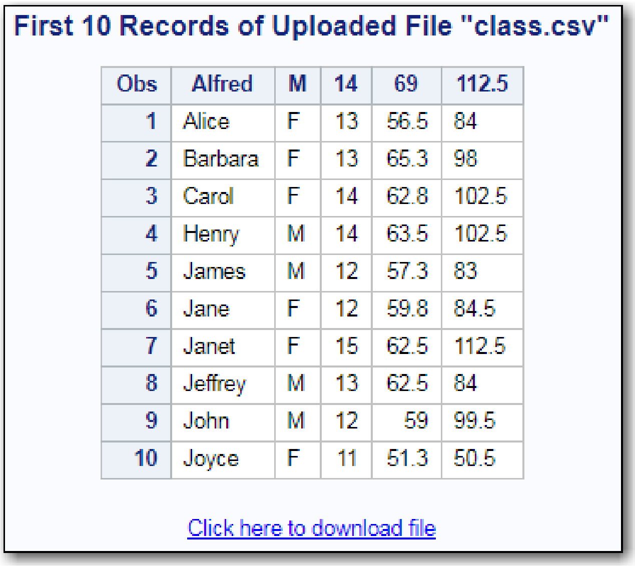
The %JESEND macro executes after the last line of code and closes all open ODS destinations.

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.



First 10 Records of Uploaded File "class.csv"

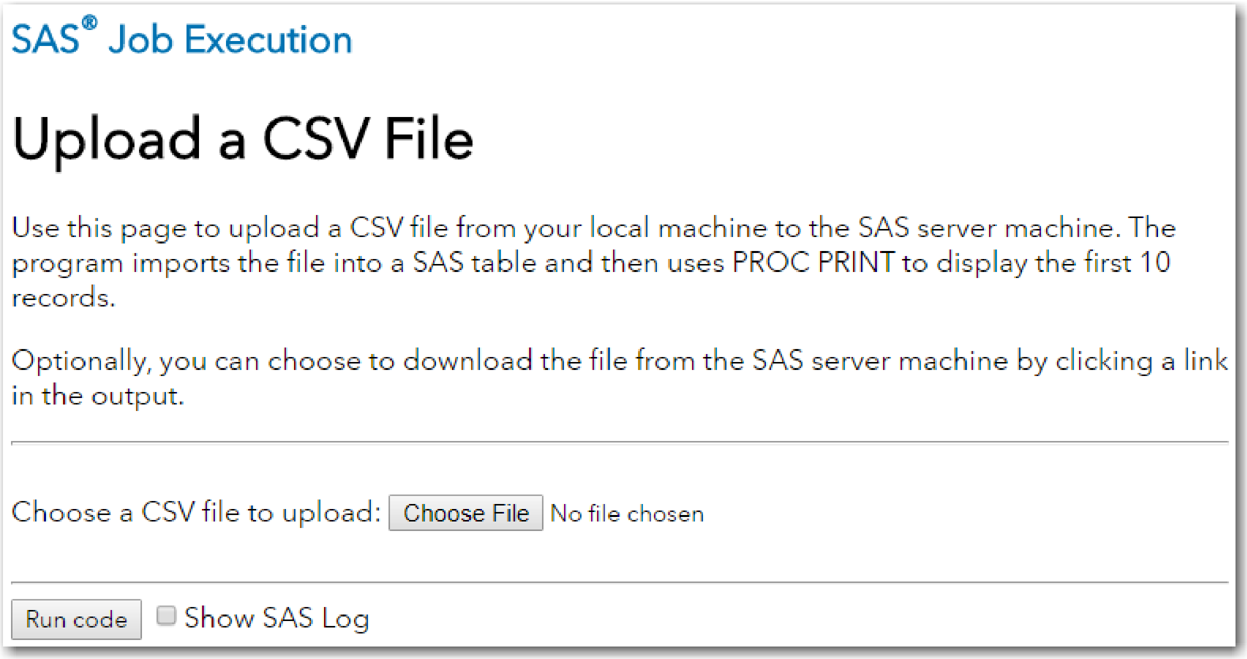
Obs	Alfred	M	14	69	112.5
1	Alice	F	13	56.5	84
2	Barbara	F	13	65.3	98
3	Carol	F	14	62.8	102.5
4	Henry	M	14	63.5	102.5
5	James	M	12	57.3	83
6	Jane	F	12	59.8	84.5
7	Janet	F	15	62.5	112.5
8	Jeffrey	M	13	62.5	84
9	John	M	12	59	99.5
10	Joyce	F	11	51.3	50.5

Click here to download file

Job Input Parameters

Name	Value	Description
_ACTION	`form, execute`	Displays the HTML input form before the job is executed
_OUTPUT_TYPE	`ods_html5`	Specifies that ODS HTML5 output is created by the job
_ODSSTYLE	`HTMLBlue`	Specifies the name of the ODS style

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:

```
<!DOCTYPE html>
<html lang="en">

<head>
<title>Upload a CSV File</title>
<style type="text/css">

.pointer {
  cursor: pointer;
}

[Other Cascading Style Sheet code here]

</style>

</head>

<body role="main">

<div>SAS<sup>®</sup> Job Execution</div>

<h1>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/>

<form action="/SASJobExecution/" method="post" target="_SASResults"
  enctype="multipart/form-data">
<input type="hidden" name="_program" value="/Folder/Upload a CSV File">
<input type="hidden" name="_action" value="execute"/>
<input type="hidden" name="_output_type" value="ods_html5"/>
<input type="hidden" name="_csrf" value="$CSRF$">

<br/>

<label for=" myfile">Choose a CSV file to upload:</label>
<input type="file" name="myfile" id=" myfile" required class="pointer"/>

<br/>
<br/>

<hr/>

<input type="submit" value="Run code" class="pointer"/>
<input type="checkbox" name="_debug" id="_debug" value="log" class="pointer"/>
```

```
<label for="_debug">Show SAS Log</label>

</form>

</body>

</html>
```

This input form is functionally equivalent to the form used in "Upload a File" on page 68.

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:

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 CSV File**
_DEBUG	**log** (if the check box is selected)

Additional macro variables with information about the uploaded file are created. See "Upload a File" on page 68 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 filesvc 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 ODS HTML5 destination is used because it is the most up-to-date destination for creating HTML output. 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 68 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.

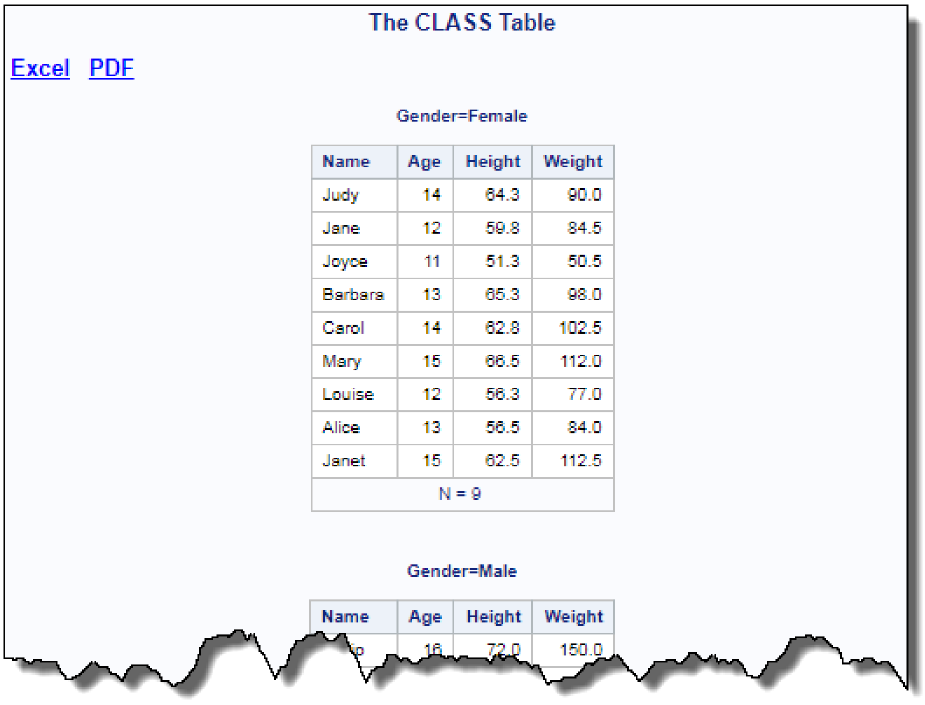
The PRINT procedure displays the first 10 records of the SAS table that is created by the IMPORT procedure, and a link to download the CSV file is created by the FOOTNOTE statement.

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

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.

Output



The CLASS Table

Excel PDF

Gender=Female

Name	Age	Height	Weight
Judy	14	64.3	90.0
Jane	12	59.8	84.5
Joyce	11	51.3	50.5
Barbara	13	65.3	98.0
Carol	14	62.8	102.5
Mary	15	66.5	112.0
Louise	12	56.3	77.0
Alice	13	56.5	84.0
Janet	15	62.5	112.5
N = 9			

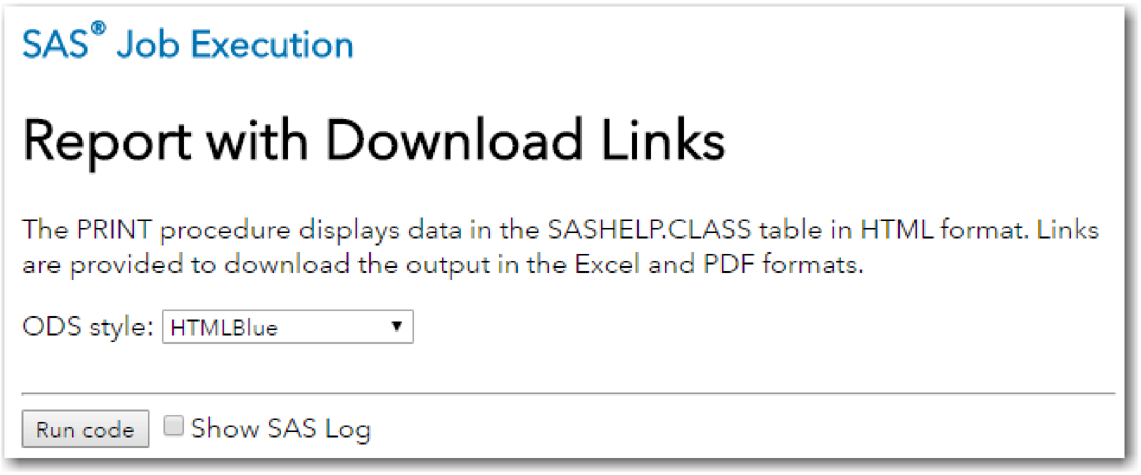
Gender=Male

Name	Age	Height	Weight
[illegible]	16	72.0	150.0

Job Input Parameters

Name	Value	Description
_ACTION	`form, execute`	Displays the HTML input form before the job is executed
_OUTPUT_TYPE	`none`	Suppresses automatic issuing of FILENAME and ODS statements
_ODSSTYLE	`HTMLBlue`	Specifies the name of the ODS style

HTML Input


SAS® Job Execution

Report with Download Links

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: HTMLBlue

```
<!DOCTYPE html>
<html lang="en">

<head>
<title>Report with Download Links</title>
<style type="text/css">

.pointer {
  cursor: pointer;
}

[Other Cascading Style Sheet code here]

</style>

</head>

<body role="main">

<div>SAS<sup>&#174;</sup> Job Execution</div>

<h1>Report with Download Links</h1>

<p>
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.
</p>

<form action="/SASJobExecution/" target="_SASResults">
<input type="hidden" name="_program" value="/Folder/Report with Download Links"/>
<input type="hidden" name="_action" value="execute"/>
<input type="hidden" name="_output_type" value="none"/>

<label for="_odsstyle">ODS style:</label>
<select name="_odsstyle" id="_odsstyle" class="pointer">
  [More values here]
  <option value="HTMLBlue" selected>HTMLBlue</option>
  [More values here]
```

```
</select>

<br/>
<br/>

<hr/>

<input type="submit" value="Run code" class="pointer"/>
<input type="checkbox" name="_debug" id="_debug" value="log" class="pointer"/>
<label for="_debug">Show SAS Log</label>

</form>

</body>

</html>
```

The HTML in this form uses some of the same fields as "Simple ODS HTML" on page 46. See that section for more information.

For the value of _PROGRAM, specify the path and name of the program to execute.

The FILENAME and ODS statements issued by the %JESBEGIN macro do not meet the needs of this program. Specifying **`none`** for _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 _ODSSTYLE is used by the program:

Macro Variable Name	Macro Variable Value
_ACTION	**`execute`**
_ODSDEST	blank
_ODSSTYLE	Depends on selection (for example, **`HTMLBlue`**)
_OUTPUT_TYPE	**`none`**
_PROGRAM	*/Folder/***`Report with Download Links`**
_DEBUG	**`log`** (if the check box is selected)

Program

```
*  Close all open destinations;

ods _all_ close;

options nodate nonumber;

*  Declare input parameter;

%global _ODSSTYLE;

*  Define the escape character for ODS inline formatting;
```

```
ods escapechar='^';

*  Create a format for the student gender;

proc format;
  value $gender 'F' = 'Female'
                '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;

1 ods 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;

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

3 filename 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="<span>^{style systemtitle &EXCEL_LINK^{nbspace 3
  &PDF_LINK}</span>";

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

ods html5 close;
```

1 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.

2 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 34.

3 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.

4 As discussed in "Assigning a FILEREF for HTML Output" on page 33, the FILESRVC access method creates global macro variables of the form _FILESRVC_fileref_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.

5 A FILENAME statement is issued to return HTML content to the web browser. See "Assigning a FILEREF for HTML Output" on page 33 for more information about this statement.

6 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.

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

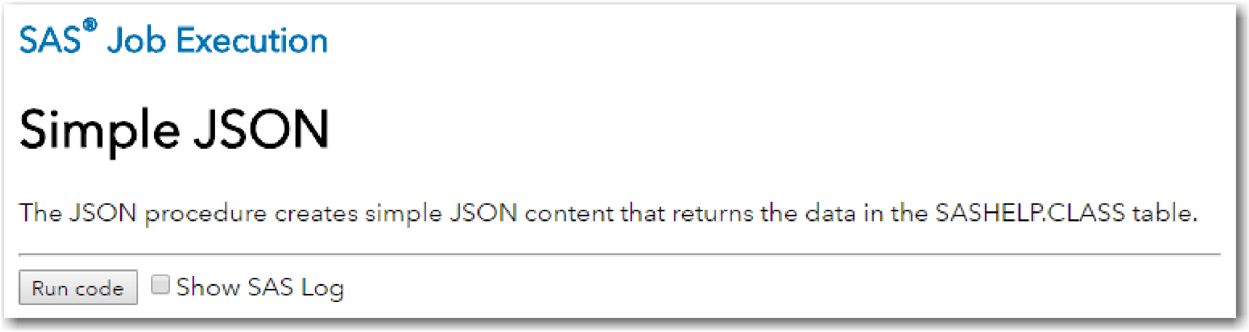
```
[
  {
    "Name": "Alfred",
    "Sex": "M",
    "Age": 14,
    "Height": 69,
    "Weight": 112.5
  },
  {
    "Name": "Alice",
    "Sex": "F",
    "Age": 13,
    "Height": 56.5,
    "Weight": 84
  },
  {
    "Name": "Barbara",
    "Sex": "F",
    "Age": 13,
    "Height": 65.3,
    "Weight": 98
 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

Job Input Parameters

Name	Value	Description
_ACTION	**`form, execute`**	Displays the HTML input form before the job is executed

Name	Value	Description
_OUTPUT_TYPE	`json`	Specifies that JSON (non-ODS) output is created by the job

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>
<style type="text/css">

.pointer {
  cursor: pointer;
}

[Other Cascading Style Sheet code here]

</style>

</head>

<body role="main">

<div>SAS<sup>&#174;</sup> Job Execution</div>

<h1>Simple JSON</h1>

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

</p>

<hr/>

<form action="/SASJobExecution/" target="_SASResults">
<input type="hidden" name="_program" value="/Folder/Simple JSON"/>
<input type="hidden" name="_action" value="execute"/>
<input type="hidden" name="_output_type" value="json"/>

<input type="submit" value="Run code" class="pointer"/>
```

```
<input type="checkbox" name="_debug" id="_debug" value="log" class="pointer"/>
<label for="_debug">Show SAS Log</label>

</form>

</body>

</html>
```

The HTML in this form uses some of the same fields as "Simple ODS HTML" on page 46. See that section for more information.

For the value of _PROGRAM, specify the path and name of the program to execute.

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

```
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.

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.

Name	Value	Action
_CONTDISP	Any valid value	Specifies a value for the CONTENTDISP option in the FILENAME statement.

Name	Value	Action
_CONTDISP_FILEEXT		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).
_CONTDISP_FILENAME		Specifies the file name to use in the CONTENTDISP option in the FILENAME statement if _CONTDISP is not specified. The default value is `SASResults`.
_CONTTYPE	Any valid value	Specifies a value for the CONTENTTYPE option in the FILENAME statement.
_DEBUG	Any valid value for _DEBUG	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.
_ENCODING	Any valid SAS encoding	Specifies a value for the ENCODING option in the ODS statement.
_FILEREF_OPTIONS	Any valid value for the FILESRVC FILENAME statement	Specifies a value to add to the end of the FILENAME statement.
_GOPT_DEVICE	Any valid value for the DEVICE graphic option	Specifies a value to add to a GOPTIONS statement.
_GOPT_HSIZE	Any valid value for the HSIZE graphic option	Specifies a value to add to a GOPTIONS statement.
_GOPT_VSIZE	Any valid value for the VSIZE graphic option	Specifies a value to add to a GOPTIONS statement.
_GOPT_XPIXELS	Any valid value for the XPIXELS graphic option	Specifies a value to add to a GOPTIONS statement.
_GOPT_YPIXELS	Any valid value for the YPIXELS graphic option	Specifies a value to add to a GOPTIONS statement.
_GOPTIONS	Any value that is valid in a GOPTIONS statement	Specifies a value to add to a GOPTIONS statement.
_ODS_DEVICE	Any valid graphics device	Specifies a value for the OUTPUTFMT option of the ODS GRAPHICS statement.
_ODS_EMBED_GRAPHICS	`N`, `NO` (case–insensitive)	Specifies ODS options for embedding graphics. This macro is valid only for the HTML5 destination. By default, these options are turned on.

Name	Value	Action
_ODSOPTIONS	Any valid ODS options	Specifies a value to add to the ODS statement.
_ODSSTYLE	Any valid ODS style	Specifies a value for the STYLE option in the ODS statement.
_ODSSTYLESHEET_URL	Any valid value	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.
_OUTPUT_TYPE	`NONE`, `ODS_`*ods-destination*, `html`, `pdf`, `json`, and so on (case–insensitive)	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`.
_SUPPRESS_MVARS	`Y`, `YES` (case–insensitive)	Suppresses the display of macro variables (by the %JESBEGIN macro) before executing user code.

The %JESBEGIN macro creates the following global macro variables:

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

Reserved Macro Variables

SAS Variable Name	Description
_ACTION	Specifies the _ACTION job input parameter, if any.

SAS Variable Name	Description
_APSLIST	Specifies a list of job input parameters.
_CONTEXTNAME	Specifies the Compute service context name.
_CONTDISP	Acts as an input parameter for the %JESBEGIN macro.
_CONTDISP_FILEEXT	Acts as an input parameter for the %JESBEGIN macro.
_CONTDISP_FILENAME	Acts as an input parameter for the %JESBEGIN macro.
_CONTTYPE	Acts as an input parameter for the %JESBEGIN macro.
_CSRF	Specifies the Cross-Site Request Forgery token for this request.
_DEBUG	Specifies the _DEBUG job input parameter, if any.
_ENCODING	Acts as an input parameter for the %JESBEGIN macro.
_FILEREF_OPTIONS	Acts as an input parameter for the %JESBEGIN macro.
_FILESRV_*fileref*_URI	Specifies the URI of a file created by a FILEREF using the FILESRVC engine.
_GOPT_DEVICE	Acts as an input parameter for the %JESBEGIN macro.
_GOPT_HSIZE	Acts as an input parameter for the %JESBEGIN macro.
_GOPT_VSIZE	Acts as an input parameter for the %JESBEGIN macro.
_GOPT_XPIXELS	Acts as an input parameter for the %JESBEGIN macro.
_GOPT_YPIXELS	Acts as an input parameter for the %JESBEGIN macro.
_GOPTIONS	Acts as an input parameter for the %JESBEGIN macro.
_HTUA	Specifies the name of the user agent.
_JOB	Specifies a globally unique identifier.
_JOBERROR	Acts as an output parameter for the %JESBEGIN macro.
_ODS_DEVICE	Acts as an input parameter for the %JESBEGIN macro.
_ODS_EMBED_GRAPHICS	Acts as an input parameter for the %JESBEGIN macro.
_ODSDEST	Acts as an output parameter for the %JESBEGIN macro.
_ODSOPTIONS	Acts as an input parameter for the %JESBEGIN macro.
_ODSSTYLE	Acts as an input parameter for the %JESBEGIN macro.
_ODSSTYLESHEET_URL	Acts as an input parameter for the %JESBEGIN macro.

SAS Variable Name	Description
_OMITJSONLISTING	Specifies whether an internal JSON listing file is returned.
_OMITJSONLOG	Specifies whether an internal JSON log file is returned.
_OMITSESSIONRESULTS	Specifies whether any results are returned.
_OMITTEXTLISTING	Specifies whether an internal text listing file is returned.
_OMITTEXTLOG	Specifies whether an internal text log is returned.
_OUTPUT_TYPE	Acts as an input parameter for the %JESBEGIN macro.
_PROGRAM	Specifies the path and name of the job.
_REPLAY	Reserved for future use.
_RESULTFILE	Specifies the output result files to be returned.
_RMTADDR	Specifies the Internet Protocol (IP) address of the client that sent the request.
_RMTHOST	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.
_SAVEFILE	Specifies the name of the file for saved output.
_SAVEFOLDER	Specifies the name of the folder for saved output.
_STATUS_MESSAGE	Specifies the message text that is displayed by the client after a job executes.
_SUPPRESS_MVARS	Acts as an input parameter for the %JESBEGIN macro.
_URL	Specifies the URL of the web server middle tier that is used to access the job.
_USERLOCALE	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.
_VERSION	Specifies the SAS Job Execution Web Application version number.
_XFORWARD	Specifies the host and port of the original HTTP request.

SAS Variable Name	Description
_WEBIN_CONTENT_LENGTH	Contain properties of the file that is being uploaded. See the Upload a File and Upload a CSV File sections in “Samples” on page 42 for more information.
_WEBIN_CONTENT_TYPE	
_WEBIN_FILE_COUNT	
_WEBIN_FILEEXT	
_WEBIN_FILENAME	
_WEBIN_FILEURI	
_WEBIN_NAME	
SYS_COMPUTE_JOB_ID	Specifies the Compute service job ID.
SYS_COMPUTE_SESSION_ID	Specifies the Compute service session ID.
SYS_JES_JOB_URI	Specifies the Job Execution service object URI.

_ACTION Input Parameter Values

The following values are supported by the _ACTION parameter:

Value	Description
background	Executes the job in the background.
execute	Executes the job.
form	Displays an HTML input form file stored in the folder structure before job execution.
json	Returns a list of unexpired jobs or sample jobs in JSON format.
lastjob	Displays output from a previous job execution if it has not yet expired.
schedule	Indicates that a job is to be scheduled using SAS Environment Manager.
wait	Displays a wait screen with informational text while the job is executing. This value works only when _ACTION is used as a URL parameter.

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

Name	Description
mvar	Required. Specifies the name of the macro variable that corresponds to the prompt name.
outvar	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.
dlm	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.
quote	Optional. Specify **`y`** (case-insensitive) to quote the individual values in the converted parameter list. By default, the value is **`n`**.

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:

```
%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:

```
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:

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

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):

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

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:

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

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

Source Code

```
%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 return;
  run;
  %goto exit;
%end;

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

%let QUOTE=%upcase(&QUOTE);
```

```
%if (%bquote(&OUTVAR) eq )
  %then %let OUTVAR = _&MVAR;

%if (%sysfunc(nvalid(%bquote(&OUTVAR), v7)) ne 1) %then %do;
  %put ERROR: Please specify a valid macro variable name for the OUTVAR argument.;
  data _null_;
  abort return;
  run;
  %goto exit;
%end;

%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;

%let &OUTVAR=&PARAMLIST;

%exit:
%mend param_list;
```

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 your name 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 confirmation 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 the system administrator. The theme specifies the collection of colors, graphics, and fonts that appear in the application. You can choose from SAS themes or custom themes, if available.

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

SAS themes:

Illuminate
This theme has a clean and uncomplicated color palette that is easy to use.

Inspire
This theme consists of vibrant and cohesive colors that shift the emphasis from the application to the content.

High Contrast
This theme presents a dark background with high-contrast foreground elements to meet the needs of users with low vision.

- If you want messages to display that you previously asked not to display, click **Reset Messages**. By default, all warnings and confirmation messages are displayed.
- You can select a profile picture to display as an avatar in the application bar, as well as in other places within the application that use avatars. An *avatar* is the graphical representation of the user or the user's alter ego or character.

  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 settings that enable users to specify the locale for regional formats and sorting, as well as for offline processes. Here are the settings:

- 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 affect 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 enabled, you see a subtle movement in the user interface if you delete an item.
- 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.
- 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.

Landmarks are references to the primary areas of an application’s user interface. They provide a quick and easy way for keyboard users to navigate to these areas of the application. You can access a list of landmarks by using one of the following keyboard shortcuts:

- For Microsoft Windows, press Ctrl+F6.

- For Mac, press Command+F6.

Use the arrow keys to select a landmark, and then press Enter to navigate to that area of the application.