SAS1670-2018

Data Management in SAS® Viya®: A Deep Dive

Wilbram Hazejager and Nancy Rausch, SAS Institute Inc.

ABSTRACT

This paper provides an in-depth look into the new SAS® data management capabilities in support of SAS® Viya® and SAS® Cloud Analytic Services. The paper includes an overview of SAS® Data Management in SAS Viya, and contains details about how the feature set integrates with SAS Cloud Analytic Services. Examples and usage scenarios of how to best leverage the technology are also included.

INTRODUCTION

Self-service data preparation from SAS, on SAS Viya, helps you access, profile, cleanse, and transform data from an intuitive interface. SAS® Data Explorer copies data to SAS Cloud Analytic Services (CAS) and enables you to navigate and manage that data. SAS® Data Studio builds and executes collections of transformations on data that has been loaded to CAS.

This paper takes you through some common usage scenarios, and for each of those scenarios explains in more detail how the different architecture components are being used.

HIGH-LEVEL ARCHITECTURE

SAS® Data Preparation is built on SAS Viya, and encompasses a number of web applications, including the following (which is not an exhaustive list):

- SAS Data Explorer to manage data
- SAS Data Studio to build and execute collections of transformations on data
- SAS® Job Monitor to monitor the status of data management jobs

A number of additional web applications are available to the user as part of SAS Viya, including the following (which is not an exhaustive list):

- SAS® Environment Manager for managing a SAS Viya environment. It includes a dashboard view, which provides a quick overall look of your environment's health and status, as well as detailed views that enable you to examine and manage your environment in detail.
- SAS® Lineage Viewer to better understand the relationships between objects in your SAS Viya applications. These objects include data, transformation processes, reports, and visualizations.

SAS web applications talk to SAS Viya services, often referred to as microservices. SAS Viya includes services such as Audit, Identities, and Monitoring.

SAS Data Preparation uses SAS Cloud Analytics Services (CAS), as the run-time environment to interact with data sources and execute data transformations. CAS uses SAS® Data Connectors and SAS® Data Connector Accelerators to read from and write to data sources. Here is a high-level architecture diagram.

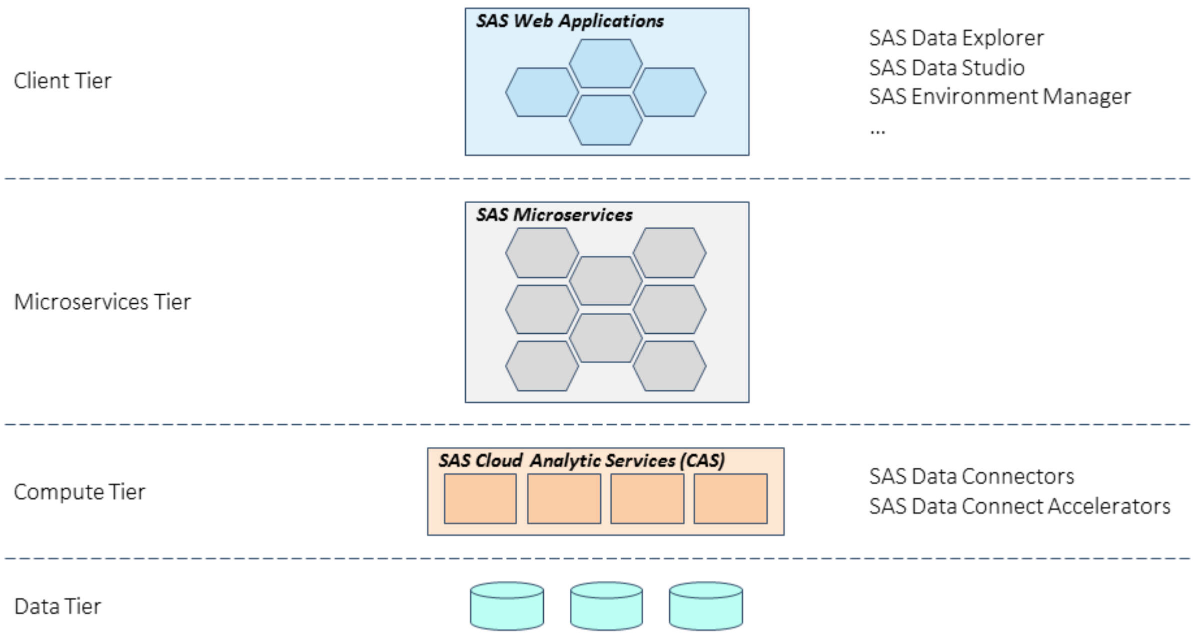
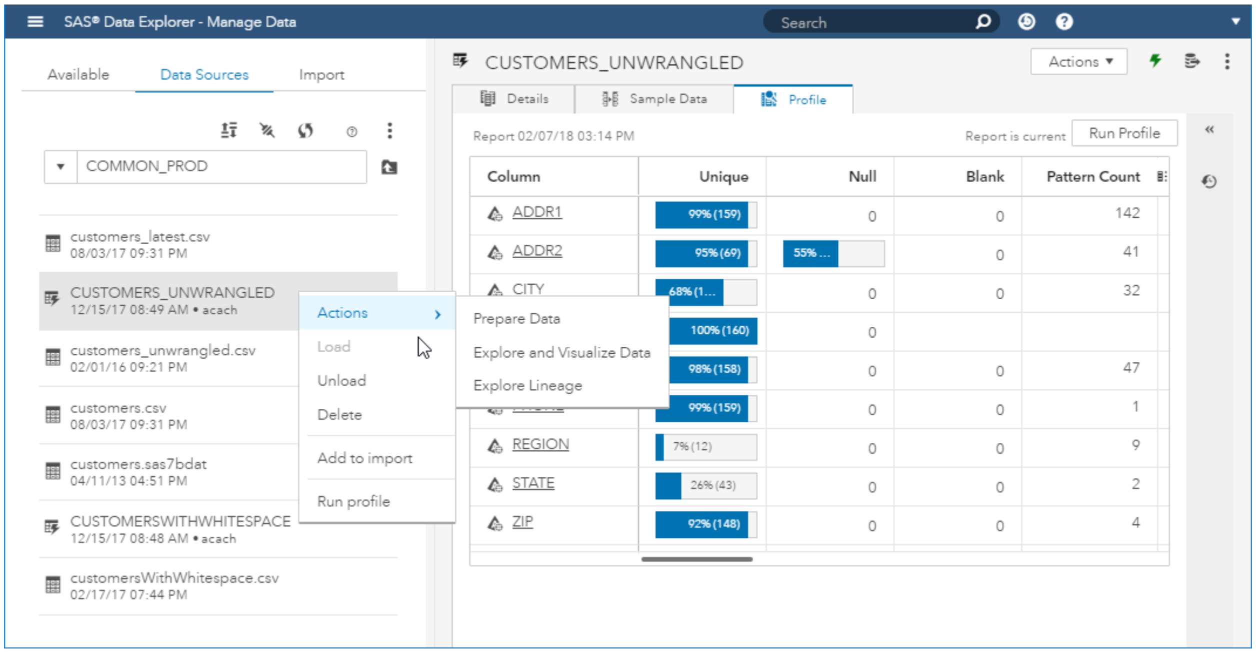


Figure 1. SAS Data Preparation: High-level Architecture

In this paper, we primarily focus on SAS Data Explorer and SAS Data Studio.

LOADING DATA IN CAS USING SAS DATA PREPARATION

SAS Data Preparation applications use CAS as the run-time engine. Using SAS Data Explorer, you can load data to CAS from a variety of sources, including files, databases, social media sources, and Esri. Profile metrics are available as well as can be seen in the following figure.



customers_latest.csv
08/03/17 09:31 PM

CUSTOMERS_UNWRANGLED
12/15/17 08:49 AM • acach

customers_unwrangled.csv
02/01/16 09:21 PM

customers.csv
08/03/17 09:31 PM

customers.sas7bdat
04/11/13 04:51 PM

CUSTOMERSWITHWHITESPACE
12/15/17 08:48 AM • acach

customersWithWhitespace.csv
02/17/17 07:44 PM

CUSTOMERS_UNWRANGLED

Report 02/07/18 03:14 PM

Report is current

Column	Unique	Null	Blank	Pattern Count
ADDR1	99% (159)	0	0	142
ADDR2	95% (69)	55% ...	0	41
CITY	68% (1...	0	0	32
	100% (160)	0		
	98% (158)	0	0	47
	99% (159)	0	0	1
REGION	7% (12)	0	0	9
STATE	26% (43)	0	0	2
ZIP	92% (148)	0	0	4

Figure 2. SAS Data Explorer: Profile Metrics

Loading Data from DBMS and Hadoop in CAS

Loading data from a DBMS or Hadoop in CAS is done using SAS Data Connectors and SAS Data Connect Accelerators.

SAS Data Connectors connect to the data source and can load data in a serial mode. In SAS Viya 3.3, these connectors have been enhanced to support a new data transfer mode called MultiNode. This mode allows multiple CAS worker nodes to connect to the data source at the same time. For more information about this data transfer mode, see *SAS® 9.4 and SAS® Viya® 3.3 Programming Documentation*.

SAS Data Connect Accelerators extend the functionality of SAS Data Connectors by enabling a parallel data load capability between the database clusters and CAS, while using SAS® Embedded Process framework to orchestrate such parallelization. Here is a diagram to illustrate this.

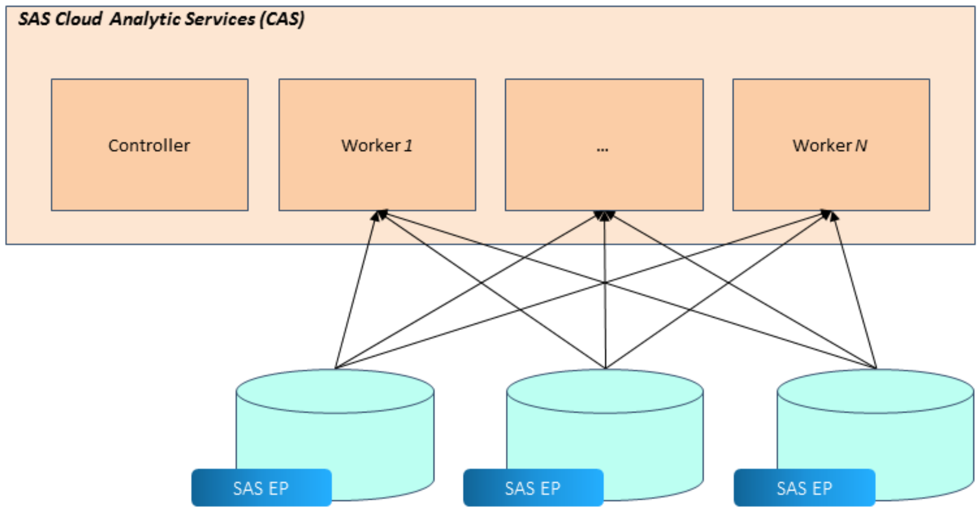


Figure 3. Parallel Loading from DBMS and Hadoop to CAS using SAS Embedded Process

SAS Embedded Process on each of the data source nodes has quick access to local data and can work directly with CAS worker nodes.

When using the SAS Data Connect Accelerator to Hadoop, the Hadoop ecosystem optimizes the request for data in such a way that each Hadoop node first tries to serve up its local data. This is to minimize data transfer between the Hadoop nodes. Therefore, the SAS Embedded Process transfers this local data to a CAS worker node. If CAS is co-located, then the SAS Embedded Process chooses the co-located worker, enabling optimal data transfer in terms of network usage.

While using the MultiNode approach, the data source is asked for a partition of the data, and that complete partition is then received by a single CAS node, the CAS node that requested the partition. Although MultiNode should be able to improve load times compared to using serial load, the SAS Data Connector Accelerator approach has more capabilities to optimize the parallel data load and if available is the recommend approach when working with large data sizes. Actually, when requesting data without the MultiNode option and SAS Data Connect Accelerator being available, CAS automatically uses parallel loading using SAS Embedded Process.

Starting with SAS Viya 3.3, these SAS data access components support both Read and Write access. The Write capabilities also apply to both of the parallel data transfer approaches discussed above.

Loading Files in CAS

SAS Data Preparation supports loading files into CAS. The file formats supported include Microsoft Excel, comma-separated value format (CSV format) files, and SAS data sets (sas7bdat and sashdat).

When the files reside on a file system that is directly accessible by CAS, we talk about server-side loading. This approach is used when you select a file from the Data Source panel in SAS Data Explorer. This data source requests a list of files from CAS and then lists server-side-available files.

When the files reside on the local file system of the machine where the browser is running, server-side processes, which includes CAS, typically cannot access that file system. However, using SAS Data Explorer, you can still load the data from the local file into a CAS table. We call this local file loading. This functionality is available using the Import tab of SAS Data Explorer. Under the covers, the data has to travel over the wire from web browser, to SAS Data Explorer mid-tier, and then (using microservices) to CAS as shown in the following diagram.

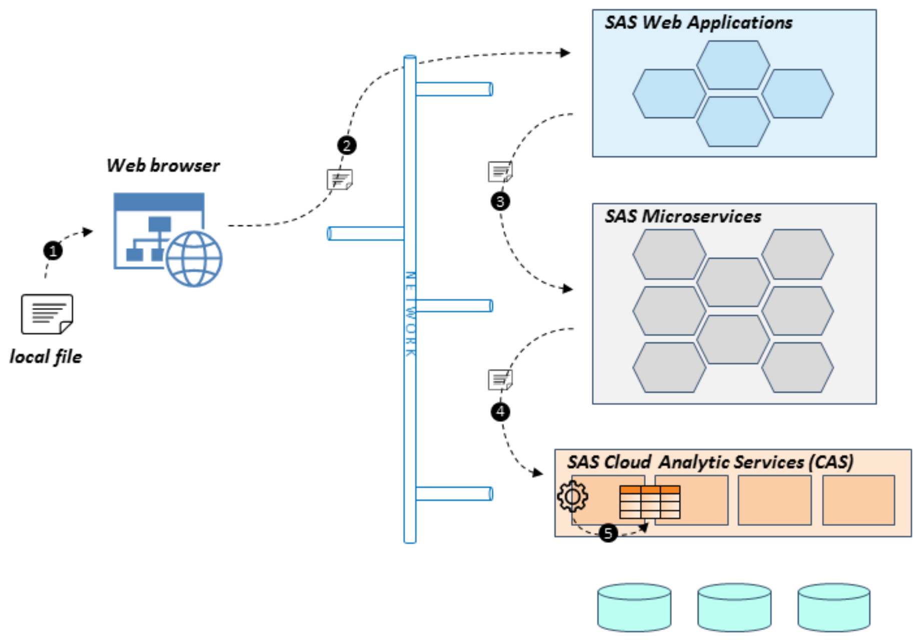


Figure 4. SAS Data Explorer: Local File Loading

Once the data arrives at the CAS server, functionality in the CAS server converts the file to a CAS table.

When loading large files, you should use server-side loading to minimize network usage.

Note that you can load data that is bigger than the total amount of memory in the CAS cluster.

TRANSFORMING DATA IN SAS DATA PREPARATION

SAS Data Studio is used to build and execute collections of transformations on data. It comes with a transformation library that includes common column transformations, table transformation to join and filter data, and data quality transformations. SAS Data Studio generates CASL code to execute the transformations and hands over the code directly to CAS. CASL is the CAS language and is used to invoke CAS actions. Actions are the equivalent of SAS procedures in the SAS language, and CAS supports a wide variety of actions. For more details about CASL, see *SAS® Cloud Analytic Services 3.3: CASL Reference*.

Because of the interactive nature of SAS Data Studio, it applies a transformation as soon as it has been added in the UI. Therefore, the user immediately sees the results of applying the transformation. These intermediate results are stored in CAS session tables, and these tables have system-generated names. If you want to keep these results for future use, save the Data plan, which is the collection of all transformations that have been applied. When saving the Data plan, you are asked to provide a name for the output table. That name replaces the system-generated name. That output table is then also written to disk, so the data is still available after the CAS server restarts.

When a Data plan is executed, it is optimized such that all consecutive column transformations are executed in one DATA step action to minimize the number of times each record in the table is touched.

CUSTOM CODE TRANSFORMATIONS IN SAS DATA STUDIO

SAS Data Studio also supports Custom Code transformations, which allow you to write custom code that perform CAS actions or transformations on a table. There are two code languages available: CASL and DATA step. These Custom Code transformations support special references to use in your code to point to the output table of the previous transformation, and to indicate the table that is (going to be) created by your custom code, so the next transformation knows which table to use. Here is a diagram that shows this in pictures.

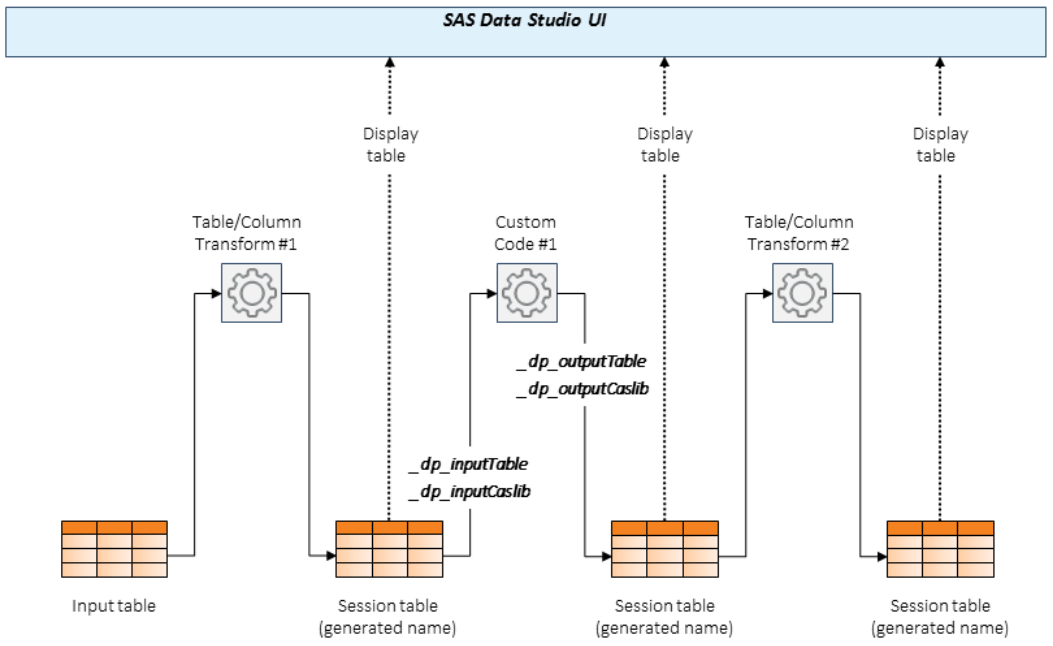


Figure 5. Referencing Input and Output Tables in Custom Code in SAS Data Studio

NOTE: These references are case-insensitive. We write them using special casing for readability purposes.

Later in this paper, we show code examples that use the _dp_inputTable, dp_inputCaslib, _dp_outputTable, and _dp_outputCaslib references.

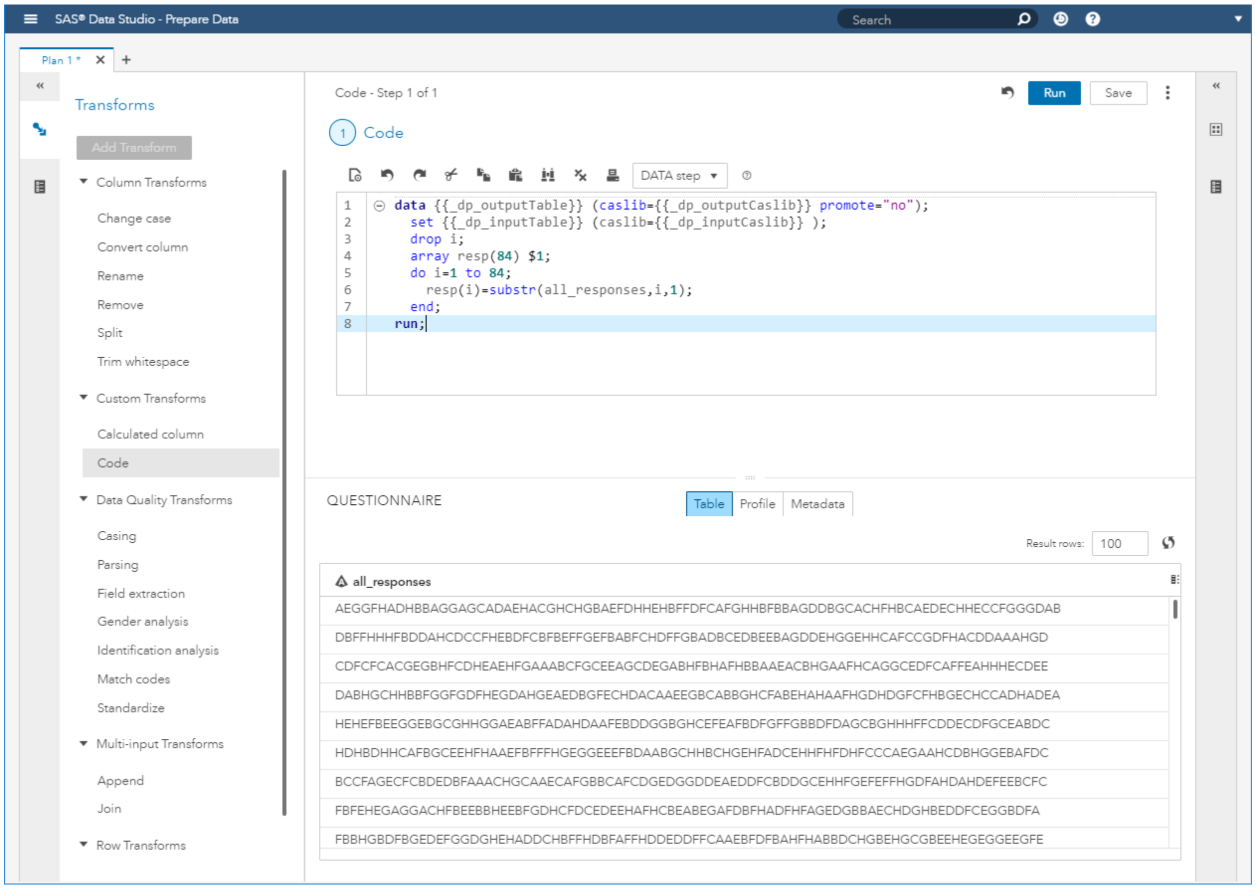
A caslib is an in-memory space to hold tables, access control lists, and data source information. All data is available to CAS through caslibs, and all operations in CAS that use data are performed with a caslib in place.

When you use the DATA step language, you can take advantage of your SAS DATA step skills to write transformations, and those transformations will run in multiple threads in parallel in CAS. Note that not all SAS language elements are supported in CAS. See the “DATA Step Programming for CAS” topic in the *SAS 9.4 and SAS Viya 3.3 Programming Documentation* for more details around which elements are supported.

Given the breadth of CAS actions available, the Custom Code transformation is typically used to take advantage of CAS actions that are not covered by the standard transformations. Another usage scenario is one where you need to perform similar transformations many times to the same data record. Often this is easier to accomplish writing a small piece of Custom Code of type DATA step, compared to adding the same transformation many times and filling in the transformation properties each time.

CUSTOM CODE TRANSFORM OF TYPE DATA STEP

Here is a simple example where the Custom Code transformation of type DATA step can be used. In this example, a table contains answers for each question in a multiple-choice questionnaire where those answers were “compressed” into a single field. The following screenshot shows the input data and the Custom Code transform before the transformation is run.



```
data {{_dp_outputTable}} (caslib={{_dp_outputCaslib}} promote="no");
  set {{_dp_inputTable}} (caslib={{_dp_inputCaslib}} );
  drop i;
  array resp(84) $1;
  do i=1 to 84;
    resp(i)=substr(all_responses,i,1);
  end;
run;
```

all_responses
AEGGFHADHBBAGGAGCADAEHACGHCHGBAEFDHHEHBFFDFCAFGHHBFBBAGDDBGCACHFHBCAEDECHHECCFGGGDAB
DBFFHHHFBDDAHCDCCFHEBDFCBFBEFFGEFBABFCHDFFGBADBCEDBEEBAGDDEHGGEHHCAFCCGDFHACDDAAAHGD
CDFCFCACGEGBHFCDHEAEHFGAAABCFGCEEAGCDEGABHFBHAFHBBAAEACBHGAAFHCAGGCEDFCAFFEAHHHECDEE
DABHGCHHBBFGGFGDFHEGDAHGEAEDBGFECHDACAAEEGBCABBGHCFABEHAHAAFHGDHDGFCFHBGECHCCADHADEA
HEHEFBEEGGEBGCGHHGGAEABFFADAHDAAFEBDDGGBGHCEFEAFBDFGFFGBBDFDAGCBGHHHFFCDDECDFGCEABDC
HDHBDHHCAFBGCEEHFHAAEFBFFFHGEGGEEEFBDAABGCHHBCHGEHFADCEHHFHFDHFCCCAEGAAHCDBHGGEBAFDC
BCCFAGECFCBDEDBFAAACHGCAAECAFGBBCAFCDGEDGGDDEAEDDFCBDDGCEHHFGEFEFFHGDFAHDAHDEFEEBCFC
FBFEHEGAGGACHFBEEBBHEEBFGDHCFDCEDEEHAFHCBEABEGAFDBFHADFHFAGEDGBBAECHDGHBEDDFCEGGBDFA
FBBHGBDFBGEDEFGGDGHEHADDCHBFFHDBFAFFHDDEDDFFCAAEBFDFBAHFHABBDCHGBEHGCGBEEHEGEGGEEGFE

Figure 6. SAS Data Studio: Data before Custom Code Transform

Here is the DATA step code that can be used to create dedicated fields for each question.

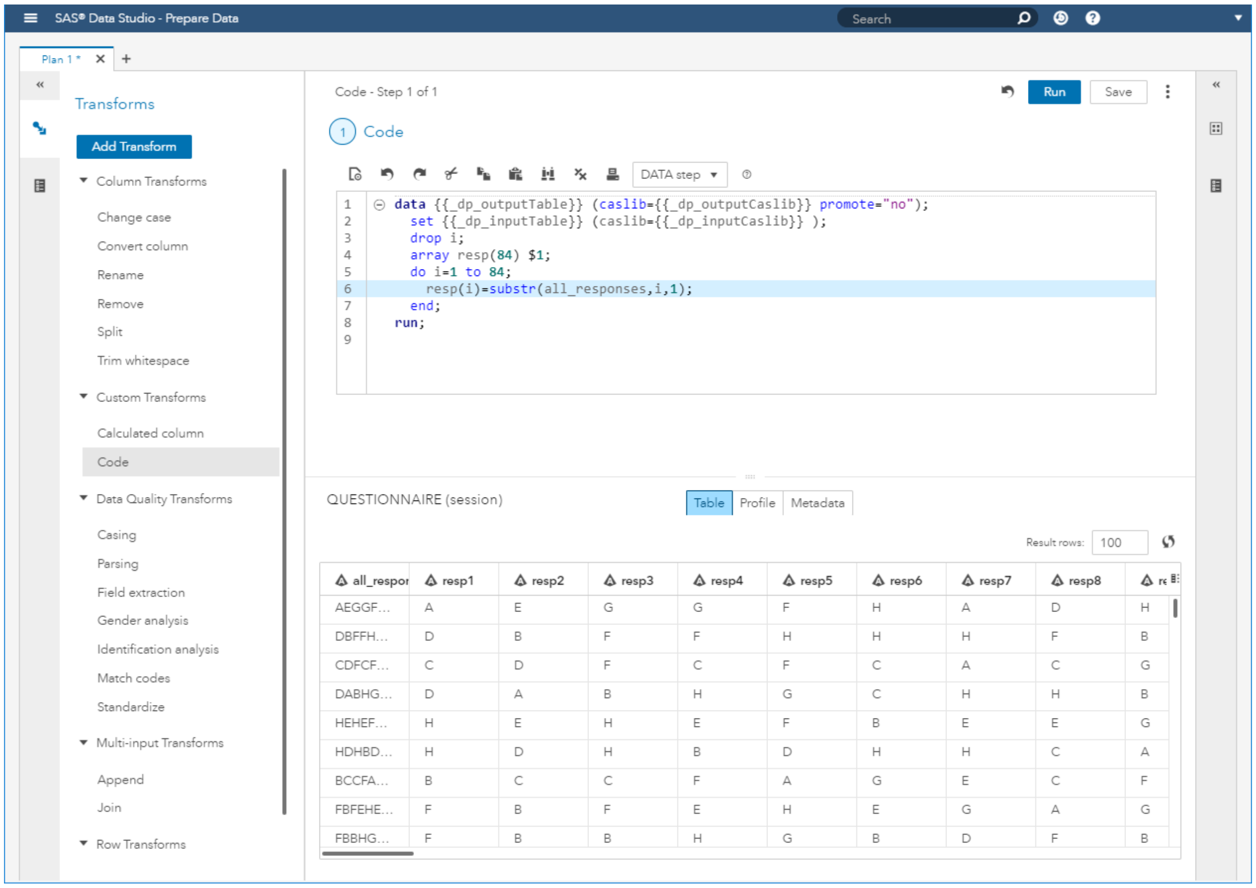
```
      data {{_dp_outputTable}} (caslib={{_dp_outputCaslib}} promote="no");
        set {{_dp_inputTable}} (caslib={{_dp_inputCaslib}} );
        drop i;
        array resp(84) $1;
        do i=1 to 84;
          resp(i)=substr(all_responses,i,1);
        end;
      run;
```

As mentioned earlier, Custom Code transformations support special references to point to input and output tables (and their corresponding caslib). Note that for the Custom Code transformation of type

DATA step, these references need to be enclosed in double curly braces as shown in the preceding code sample.

- {{_dp_inputTable}} points to the (system-generated) name of the table that was created by the previous transform.
- {{_dp_outputTable}} points the (system-generated) name of the table that the transformation is going to create. This table will be input for the next transform.

When the Data plan runs, the application replaces these references with actual table names. Here is screenshot showing the data after transformation.



```
data {{_dp_outputTable}} (caslib={{_dp_outputCaslib}} promote="no");
  set {{_dp_inputTable}} (caslib={{_dp_inputCaslib}} );
  drop i;
  array resp(84) $1;
  do i=1 to 84;
    resp(i)=substr(all_responses,i,1);
  end;
run;
```

all_respor	resp1	resp2	resp3	resp4	resp5	resp6	resp7	resp8	re
AEGGF...	A	E	G	G	F	H	A	D	H
DBFFH...	D	B	F	F	H	H	H	F	B
CDFCF...	C	D	F	C	F	C	A	C	G
DABHG...	D	A	B	H	G	C	H	H	B
HEHEF...	H	E	H	E	F	B	E	E	G
HDHBD...	H	D	H	B	D	H	H	C	A
BCCFA...	B	C	C	F	A	G	E	C	F
FBFEHE...	F	B	F	E	H	E	G	A	G
FBBHG...	F	B	B	H	G	B	D	F	B

Figure 7. SAS Data Studio: Data after Custom Code Transform

When the transformation is the first one in a Data plan, the {{_dp_inputTable}} reference will resolve to the input table name for the Data plan. This table was specified during creation of the plan. Similarly, when a transformation is the last one in a Data plan and the user specified to save the plan, {{_dp_outputTable}} will resolve to the output table name that was specified by the user in the UI as part of the save plan interaction.

CUSTOM CODE TRANSFORM OF TYPE CASL

When using Custom Code of type CASL, you should not use double curly braces around the special references. When the Data plan runs, the application will prefix your code with some CASL code that creates CASL variables with the names _dp_inputTable, _dp_inputCaslib, _dp_outputTable, and_dp_outputCaslib, and fills in the appropriate values. These names can then be directly referenced in your CASL code.

Here is an example that uses the partition action from the table action set.

```
        loadactionset('table');
        partition status=rc /
          table={caslib=_dp_inputCaslib, name=_dp_inputTable}
          casout={caslib=_dp_outputCaslib, name=_dp_outputTable, replace=true}
        ;
```

Using the Custom Code transformation of type CASL, you can implement complex logic and invoke multiple CAS actions.

CONCLUSION

We hope that the information provided in this paper gives you a better understanding of how the SAS Data Preparation applications interact with SAS Cloud Analytic Services and some of the important options available for optimizing usage, especially when large amounts of data need to be handled.

RECOMMENDED READING

- Maher, Salman. 2018. “What’s New in SAS® Viya Data Connectors.” *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings18/SAS1906-2018.pdf.
- Rausch, Nancy. 2018. “What’s New in SAS® Data Management.” *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings18/SAS1669-2018.pdf.
- SAS Institute Inc. 2017. *SAS® Cloud Analytic Services 3.3: CASL Reference*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.3&docsetId=proccas&docsetTarget=titlepage.htm&locale=en.
- SAS Institute Inc. 2017. *SAS® Cloud Analytic Services 3.3: DATA Step Programming for CAS.* Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.3&docsetId=casdspgm&docsetTarget=p1eyivn5kal7qwn1drrdt71v21ml.htm&locale=en.
- SAS Institute Inc. 2017. “Data Connectors.” *SAS® Cloud Analytic Services 3.3: User’s Guide.* Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.3&docsetId=casref&docsetTarget=n01iumvu56308zn1bud38udhg8w5.htm&locale=en.
- SAS Institute Inc. 2017. *SAS® 9.4 and SAS® Viya® 3.3 Programming Documentation: CAS User’s Guide*. Cary, NC: SAS Institute Inc. Available at http://go.documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.3&docsetId=casref&docsetTarget=titlepage.htm&locale=en.
- SAS Institute Inc. 2017. *SAS® Data Studio 2.1: User’s Guide*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/api/docsets/datastudioadv/2.1/content/datastudioadv.pdf?locale=en#nameddest=home.
- SAS Institute Inc. 2017. *SAS® Viya® 3.3: Data Preparation*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?cdcId=dprepcdc&cdcVersion=2.1.

CONTACT INFORMATION

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

Wilbram Hazejager
100 SAS Campus Dr

Cary, NC 27513
SAS Institute Inc.
Work Phone: (919) 677-8000
Fax: (919) 677-4444
Wilbram.Hazejager@sas.com
support.sas.com

Nancy Rausch
100 SAS Campus Dr
Cary, NC 27513
SAS Institute Inc.
Work Phone: (919) 677-8000
Fax: (919) 677-4444
Nancy.Rausch@sas.com
support.sas.com