Paper 145-2023

With a View to Make Your Metadata Function(al): Exploring the FMTINFO() Function

Louise S. Hadden, Abt Associates Inc.

ABSTRACT

Many SAS® programmers are accustomed to using SAS metadata on SAS data stores and processes to enhance their coding and promote data driven processing. SAS metadata is most frequently accessed through SAS "V" functions, SAS View tables, and dictionary tables. Information can be gained regarding SAS data stores and drilling down, attributes of columns in data files. However, few programmers are aware of SAS's similar resources and capabilities with respect to SAS formats. This paper and presentation will briefly discuss how SAS metadata may be exploited in general, and will demonstrate how to use the FMTINFO() function specifically.

INTRODUCTION

SAS practitioners frequently use PROC FORMAT to transform variables into an enhanced or aggregated version of the original schemas, via assigned formats used in a SAS procedure, through the use of put statements, and other intriguing methods, but there is so much more to this venerable SAS procedure.

- SAS formats may be created via text entry or from a data table via PROC FORMAT CNTLIN.
- SAS formats may be stored in specialized catalogs (work or permanent), exported to Excel workbooks, or SAS datasets via PROC FORMAT CNTLOUT.
- SAS formats may be used for data cleaning, reporting, subsetting observations, and to match data without merge/hash/SQL join.
- SAS format catalogs can be modified via PROC CATALOG.
- SAS format catalogs converted to SAS data sets may be compared using PROC COMPARE when sorted by format name and start variables.
- Use PROC FORMAT FMTLIB, PROC CATALOG, SAS Dictionary files, SAS views, and the FMTINFO() function to obtain information about SAS formats.

This paper and presentation will focus on reporting information available through SAS metadata and SAS procedures regarding SAS formats, including the use of the relatively unknown FMTINFO() function. The content is useful for all SAS users.

ANATOMY OF A FORMAT

According to the SAS documentation, a SAS format is a type of SAS language element that applies a pattern to or executes instructions for a data value to be displayed or written as output. Types of formats correspond to the type of data element: for example, numeric, character, date, time, etc. Users can also define their own formats. The bare minimum of variables that define a format are: format name (up to 32 characters), start (start of a range or value), and varlabel (value label). Other variables include hlo (high, low, other), sexcl (exclude the start of a range), eexcl (exclude the end of a range), decimal information, length, etc.

HOW DO FORMATS WORK?

The best way to figure out how formats work is to create, and then analyze them. A small data set with formats, including the special dates format, is created, and explored to see how a more complex format looks in various forms. The same technique can be used to look at ranges. What we are trying to achieve is an input data set which looks like what SAS expects under all conditions. In the example below custom date formats are being created.

```
data temp;
    d1='01jan1900'd; d2='01jan1960'd; d3=today(); d4='01jan1940'd;
run;

proc print data=temp;
run;

proc print data=temp;
format d1 d2 d3 d4 mmddyy10.;
run;

proc format fmtlib;
     value foo '01jan1900'd='Invalid'
               '01jan1940'd='Still in'
               '01jan1960'd='SAS zero'
               other=[mmddyy10.];
run;

proc print data=temp;
format d1 d2 d3 d4 foo.;
run;

proc format cntlout=foo2;
run;

proc print data=foo2;
run;
```

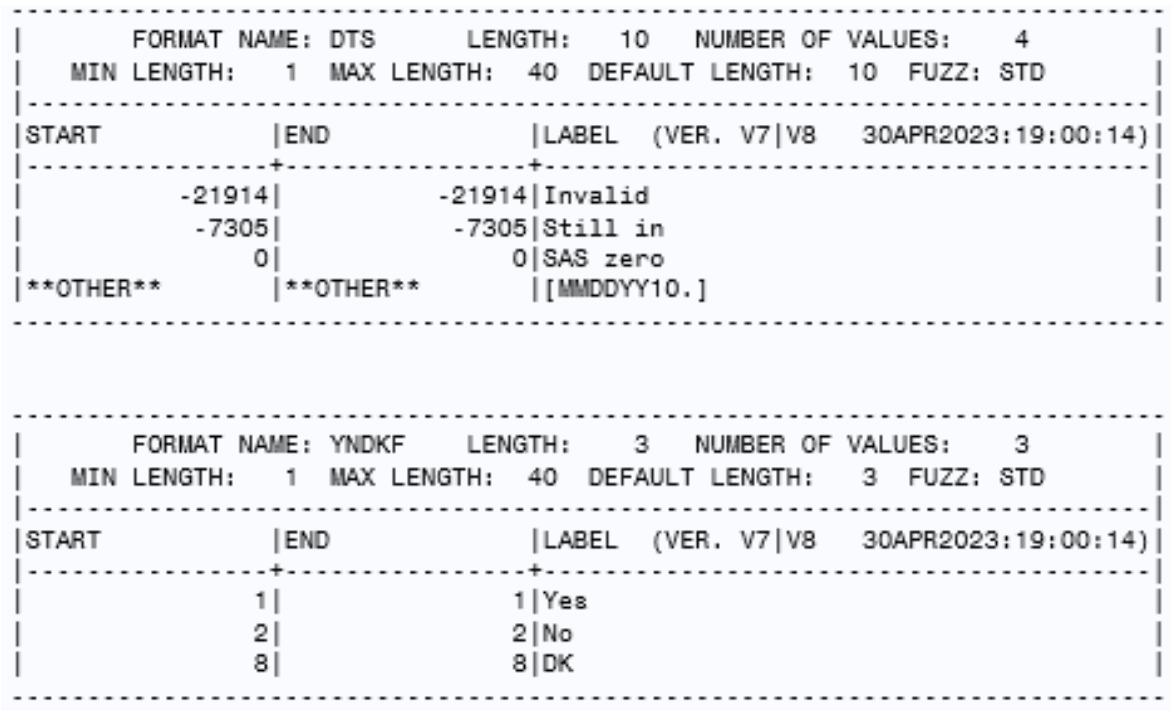
Below we see the number representation of the special dates, followed by their formatted version (SAS data format), followed by our user-defined format.

```
Obs      d1      d2      d3       d4
  1     -21914     0    22475    -7305

Obs      d1            d2            d3            d4
  1     01/01/1900    01/01/1960    07/14/2021    01/01/1940

Obs      d1          d2          d3           d4
  1     Invalid    SAS zero    07/14/2021    Still in
```

Below follows the result of using the FMTLIB in the PROC FORMAT creation step, showing how SAS represents the special date format in printed form. Note the other – this is a nested format indicating that any “other” dates should appear in MMDDYY10. format. You can use any SAS-supplied or user-created format as long as the program has access to where the format is stored.



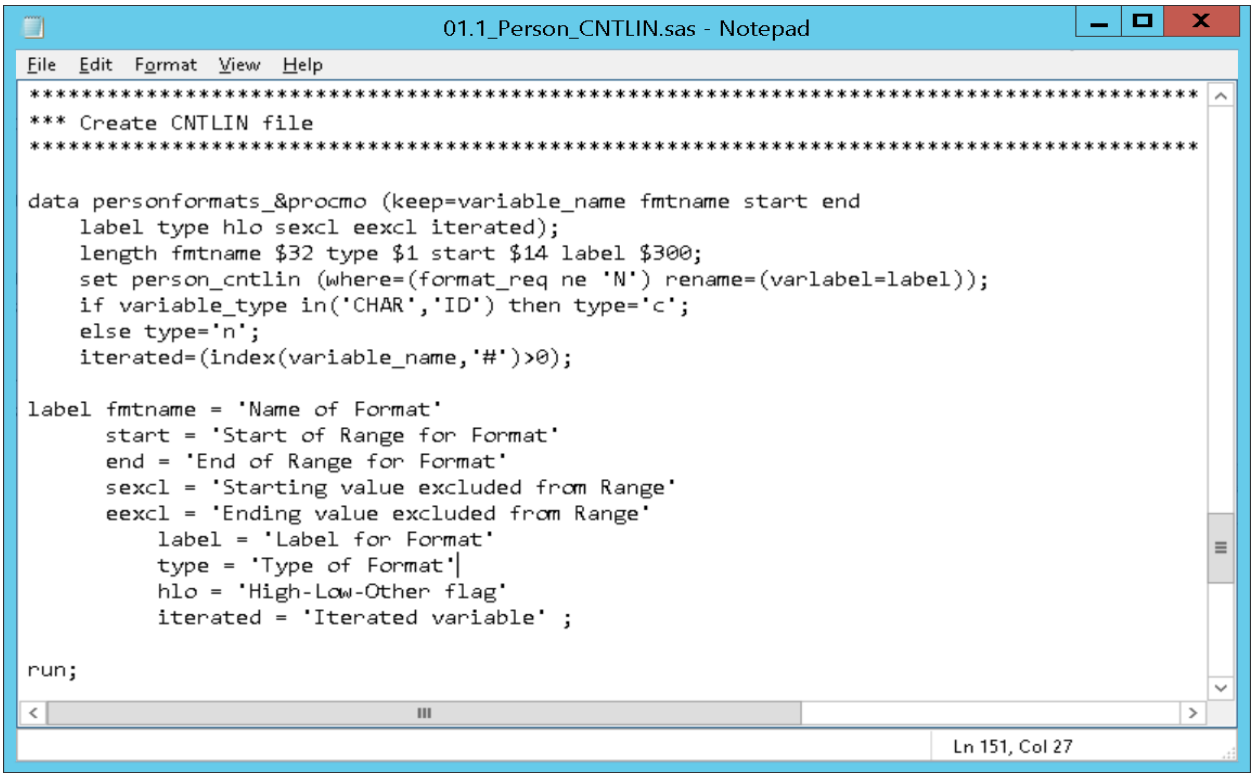
```
----------------------------------------------------------------------------
|       FORMAT NAME: DTS      LENGTH:    10   NUMBER OF VALUES:     4      |
|   MIN LENGTH:    1  MAX LENGTH:   40  DEFAULT LENGTH:   10  FUZZ: STD     |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. V7|V8   30APR2023:19:00:14)|
|----------------+----------------+-----------------------------------------|
|          -21914|          -21914|Invalid                                  |
|           -7305|           -7305|Still in                                 |
|               0|               0|SAS zero                                 |
|**OTHER**       |**OTHER**       |[MMDDYY10.]                              |
----------------------------------------------------------------------------


----------------------------------------------------------------------------
|       FORMAT NAME: YNDKF    LENGTH:     3   NUMBER OF VALUES:     3      |
|   MIN LENGTH:    1  MAX LENGTH:   40  DEFAULT LENGTH:    3  FUZZ: STD     |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. V7|V8   30APR2023:19:00:14)|
|----------------+----------------+-----------------------------------------|
|               1|               1|Yes                                      |
|               2|               2|No                                       |
|               8|               8|DK                                       |
----------------------------------------------------------------------------
```

PROC FORMAT CNTLOUT produces a SAS data set from a format catalog file, which produces yet another vision of the same format. It is this version that we need to reproduce in order to use metadata to create format catalogs.

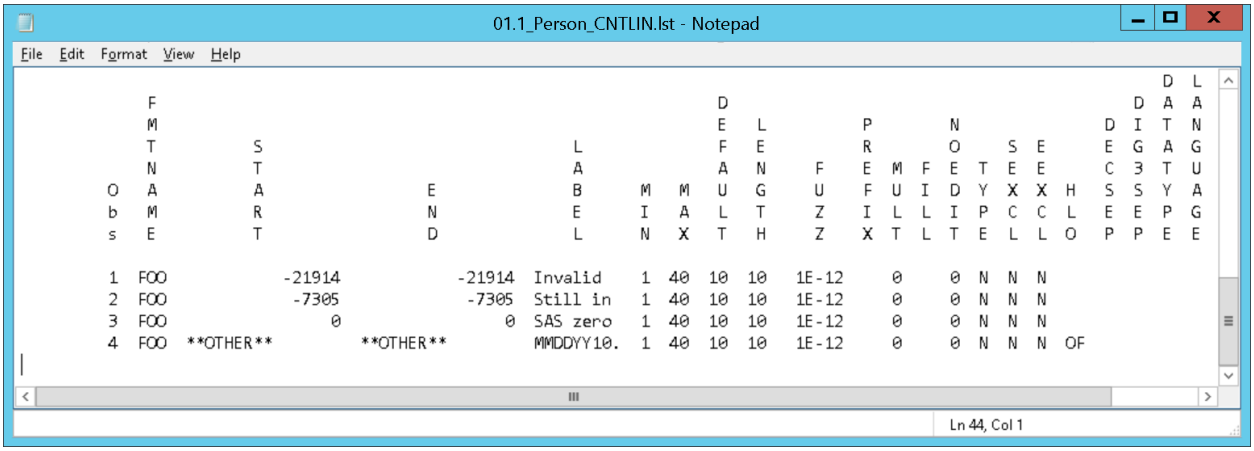
USING PROC FORMAT CNTLIN

A SAS data set is created from an Excel data dictionary import file, adjusting conform to SAS' requirements for CNTLIN data sets.

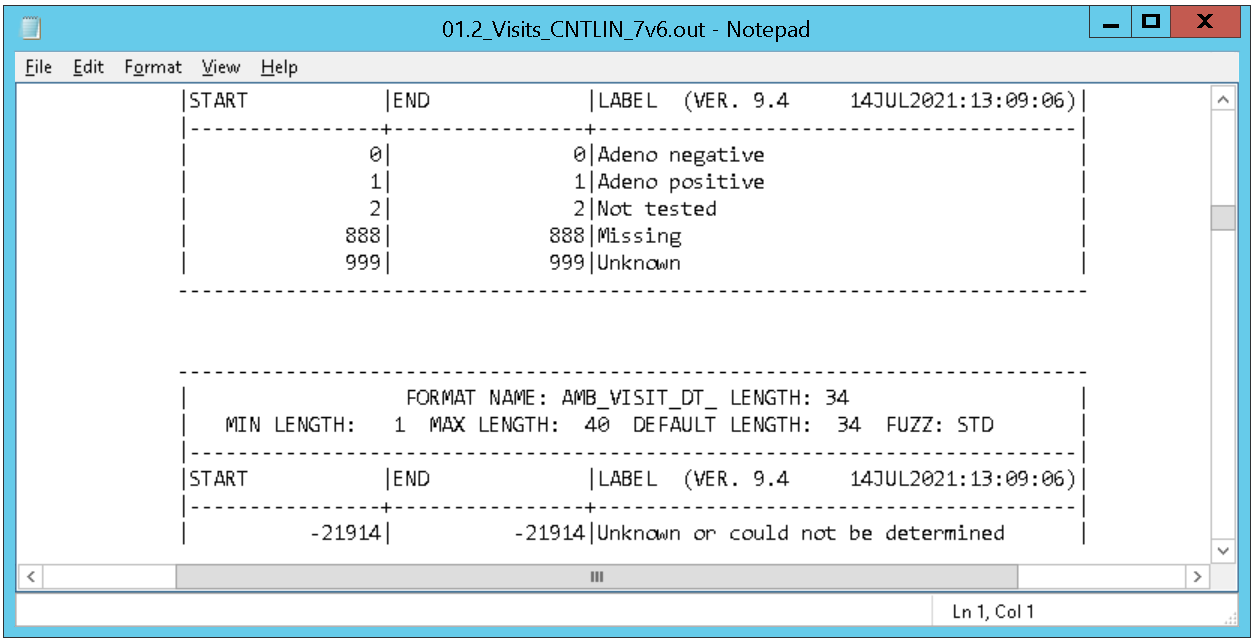


PROC FORMAT CNTLIN is then used to create a SAS format catalog from a SAS data set.

```
proc format library=library.personformats cntlin=personformats_&procmo
fmtlib ;
run;
```



The FMTLIB option prints out the resulting SAS data set. Note that the start values are SAS date values, except for the **OTHER** row, which nests an existing SAS date format. The HLO variable flags this with OF – Other Format.



MODIFYING FORMATS

The code snippet show below demonstrates how to modify a format by adding a description, and then lists the contents of the format catalog. Note that PROC CATALOG is the only way to add a description to a format, as PROC FORMAT (one of the earliest SAS procedures written) does not have a LABEL or DESCRIPTION option. This functionality is to the PROC CATALOG procedures.

```
ODS OUTPUT CATALOG_RANDOM=CR;
PROC CATALOG CATALOG = WORK.FORMATS ET=FORMAT ;
       MODIFY YNDKF
       (DESC = "1=Yes, 2=No, 8=DK, Other missing") ;
       CONTENTS;
RUN ;
QUIT;
ODS OUTPUT CLOSE;
```

The PROC CATALOG listing shows the entry name, type, dates, and description of the format.

num	objname	type	crdate	moddate	desc
1	YNDKF	FORMAT	04/29/2023 14:27:18	04/29/2023 17:24:42	1=Yes, 2=No, 8=DK, Other missing

SAS DICTIONARY FILES, VIEWS, AND "V" FUNCTIONS

SAS has several sets of internal metadata files which are available to SAS practitioners behind the scenes when a SAS session is initiated. These files are present in the form of dictionary tables, SAS views, and through selected SAS functions and call routines which build off of the dictionary tables.

SAS DICTIONARY FILES

Dictionary tables are frequently accessed through PROC SQL, while SAS views and functions are often utilized through the data step. Many programmers are familiar with the ubiquitous DICTIONARY.TABLES,

which provides information on what SAS data sets are present in what SAS libraries. However, there are a number of additional dictionary tables that have great utility for the SAS programmer. The code snippet and screen shot below show how to determine which dictionary tables are available to you during a given session.

```
PROC SQL ;
 SELECT UNIQUE MEMNAME
 FROM DICTIONARY.DICTIONARIES
QUIT ;
```

Member Name
CATALOGS
CHECK_CONSTRAINTS
COLUMNS
CONSTRAINT_COLUMN_USAGE
CONSTRAINT_TABLE_USAGE
DATAITEMS
DESTINATIONS
DICTIONARIES
ENGINES
EXTFILES
FILTERS
FORMATS
FUNCTIONS
GOPTIONS

Running the same procedure specifying MEMNAME and LABEL and using a WHERE statement to select the FORMATS table, we get a listing of what is contained in the DICTIONARY.FORMATS table.

```
PROC SQL ;
 SELECT MEMNAME LABEL
 FROM DICTIONARY.DICTIONARIES
 WHERE MEMNAME IN("FORMATS") ;
QUIT ;
```

Member Name	Column Label
FORMATS	Library Name
FORMATS	Member Name
FORMATS	Pathname
FORMATS	Object Name
FORMATS	Format Name
FORMATS	Format Type
FORMATS	Format Source
FORMATS	Minimum Width
FORMATS	Minimum Decimal Width
FORMATS	Maximum Width
FORMATS	Maximum Decimal Width
FORMATS	Default Width
FORMATS	Default Decimal Width

Library Name	Member Name	Pathname	Object Name	Format Name	Format Type	Format Source	Minimum Width	Minimum Decimal Width	Maximum Width	Maximum Decimal Width	Default Width	Default Decimal Width
				MMDDYY	F	B	2	0	10	9	8	0
				MMDDYY	I	B	6	0	32	31	6	0
				MMDDYYB	F	B	2	0	10	9	8	0
				MMDDYYC	F	B	2	0	10	9	8	0
				MMDDYYD	F	B	2	0	10	9	8	0
				MMDDYYN	F	B	2	0	8	7	8	0
				MMDDYYP	F	B	2	0	10	9	8	0
				MMDDYYS	F	B	2	0	10	9	8	0

The DICTIONARY.FORMATS table can be used to retrieve information about a specific format, or to create a report that lists all of the formats in your SAS environment. Here is an example of using DICTIONARY.FORMATS to retrieve information about a specific format:

```
PROC FORMAT LIBRARY=WORK;
    VALUE YNDKF      1 = 'Yes'
                     2 = 'No'
                     8 = 'DK'
                     ;
RUN;
```

```
28          PROC FORMAT LIBRARY=DD;
29              VALUE YNDKF 1 = 'Yes'
30                          2 = 'No'
31                                  8 = 'DK'
32                                  ;
NOTE: Format YNDKF has been written to WORK.FORMATS.
33          RUN;
```

```
PROC SQL;
  SELECT *
  FROM DICTIONARY.FORMATS
  WHERE libname='WORK' and memname='FORMATS' and fmtname='YNDKF';
QUIT;
```

Library Name	Member Name	Pathname	Object Name	Format Name	Format Type	Format Source	Minimum Width	Minimum Decimal Width	Maximum Width	Maximum Decimal Width	Default Width	Default Decimal Width
WORK	FORMATS		YNDKF	YNDKF	F	C	0	0	0	0	0	0

This example retrieves format information for the format named YNDKF in the FORMATS catalog in the WORK library.

SAS VIEWS

Views of a dictionary table perform in an analogous manner. This example retrieves format information for the format named YNDKF in the VFORMAT view in the SASHELP library.

```
data subset_vformat;
    set sashelp.vformat;
       where libname='WORK' and memname='FORMATS' and fmtname='YNDKF';
run;
proc print data=subset_vformat (obs=5) noobs;
run;
```

libname	memname	path	objname	fmtname	fmttype	source	minw	mind	maxw	maxd	defw	defd
WORK	FORMATS		YNDKF	YNDKF	F	C	0	0	0	0	0	0

Note that while the data is identical to the dictionary table information retrieved via PROC SQL, the default for SQL is to use variable labels as column headers, while the default for the view read into a data set and printed is to use variable names as column headers.

"VIEW" FUNCTIONS

SAS has variable information functions whose name starts with "V." These handy functions provide programmers with direct access to information within their code which would otherwise be retrieved from PROC CONTENTS, PROC DATASET, DICTIONARY tables, SAS views, or review of a data dictionary. The VFORMAT function is one of a number of similar functions which return information about a format that has been applied to a variable in a data set (VFORMATN, VFORMATC, VFORMATX, etc.) Note that behind the scenes, SAS is doing some calculations as to the maximum length of the returned value from the function – we all know that formats cannot end in a number, and SAS has added that information to the returned value of the VFORMAT function.

```
data vfunc;
    length q1format $8;
    format q1 yndkf.;
       q1=1; output;     put q1;
       q1=2; output;     put q1;
       q1=8; output;     put q1;
       q1format=vformat(q1);
       put q1format;
run;
```

```
28         data vfunc;
29             length q1format $8;
30             format q1 yndkf.;
31           q1=1; output;     put q1;
32           q1=2; output;     put q1;
33           q1=8; output;     put q1;
34           q1format=vformat(q1);
35           put q1format;
36         run;

Yes
No
DK
YNDKF3.
NOTE: The data set WORK.VFUNC has 3 observations and 2 variables.
```

THE FMTINFO() FUNCTION

As we have seen above, the DICTIONARY.FORMATS table contains information about the formats defined in your SAS environment. This table contains one observation for each format that has been defined. The DICTIONARY.FORMATS table includes the following columns:

- LIBRARY: The name of the SAS library that contains the format – only for formats in catalogs.
- MEMNAME: The name of the format catalog – only for formats in catalogs.
- PATH: If the format is not in a catalog, SAS-provided formats are stored in an executable DLL. The path is the path to the DLL.
- OBJNAME: The name of the format, exclusive of preceding $, etc – can be identical to FMTNAME.
- FMTNAME: The name of the format.
- FMTTYPE: The type of format. Note this is NOT character or numeric, but FORMAT and INFORMAT.
- SOURCE: C = Catalog.
- MINW: The minimum width of the format.

- MIND: The minimum decimal width.
- MAXW: The maximum width of the format.
- MAXD: The maximum decimal width.
- DEFW: The default width of the format.
- DEFD: The default decimal width of the format.

Along with PROC FORMAT and PROC CATALOG procedural output, SAS Views, and SAS Dictionary tables, SAS also offers us the FMTINFO() function, as of SAS 9.4 M3. There's no such thing as too much information, but where does the FMTINFO() function get format related information, and how does the function fit into our toolbox?

WHERE DOES FMTINFO() FIT IN THE SAS METADATA TOOLBOX?

The syntax for the FMTINFO() function is:

```
FMTINFO(fmtname,info);
```

Note that both arguments are required. The fmtname argument specifies the format or informat name, while the info argument specifies the category (cat), type, OR description of the format and informat. Only one "info" argument may be specified in a single function call.

- CAT specifies the categories of the function: BIDI Text Handling, Character, Currency Conversion, Date and Time, DBCS, Hebrew Text Handling, ISO 8601, Numeric.
- TYPE specifies the type of language element: informat, format, both.
- DESC specifies a brief description of the format or informat.
- MIND specifies the minimum decimal value of the format or informat.

We can see that a few pieces of "info"rmation are available from the dictionary.formats table and SAS view, but others are available from SAS catalog(s) and/or SAS's format DLL naming conventions.

EXAMPLES OF FMTINFO() USE

A small data set which contains several SAS provided formats and calculates various pieces of information regarding those formats is created below.

```
data fmtinfo411;
length Name $9. Type $8. Category $4. Desc $40.
       DefW $5. MinW $5. MaxW $5. DefD $2. MinD $2. MaxD $2.;
input Name @@;
Category = fmtinfo(Name, "Cat");  /* numeric, character, date, ... */
Type = fmtinfo(Name, "Type");     /* format, informat, or both */
Desc = fmtinfo(Name, "Desc");     /* brief description of the format */
DefW = fmtinfo(Name, "DefW");     /* default width if you omit w. Example: BEST., Z */
MinW = fmtinfo(Name, "MinW");     /* minimum width */
MaxW = fmtinfo(Name, "MaxW");     /* maximum width */
DefD = fmtinfo(Name, "DefD");     /* default decimal digits */
MinD = fmtinfo(Name, "MinD");     /* minimum decimal digits */
MaxD = fmtinfo(Name, "MaxD");     /* maximum decimal digits */
datalines;
ANYDTDTE BEST
DATETIME DOLLAR
Z   PERCENT
$   $UPCASE;
```

Printing the FMTINFO411 data set provides us with a reference sheet on the functions we analyzed using the FMTINFO() function.

```
proc print data=FmtInfo411 noobs;
   var Name Type Category Desc DefW MinW MaxW DefD MinD MaxD;
run;
```

Name	Type	Category	Desc	DefW	MinW	MaxW	DefD	MinD	MaxD
ANYDTDTE	INFORMAT	date	arbitrary input converted to a SAS date	12	1	32	0	0	31
BEST	BOTH	num	SAS System chooses best notation	12	1	32	0	0	31
DATETIME	BOTH	date	datetime value	16	7	40	0	0	39
DOLLAR	BOTH	curr	dollar sign, commas and decimal point	6	2	32	0	0	31
Z	BOTH	num	displays leading zeros	1	1	32	0	0	31
PERCENT	BOTH	num	writes numbers as percentages	6	4	32	0	0	31
$	BOTH	char	standard character	1	1	32767	0	0	0
$UPCASE	BOTH	char	writes all characters in uppercase	8	1	32767	0	0	0

A description was added to the YNDKF format in the work catalog. Adding a description via PROC CATALOG to our test format allows us to accomplish two goals: first, the description is added, and second, we see the contents of the format catalog. Note that you can modify descriptions for other catalog types such as function stores and macro stores using the same syntax, changing the entry type.

```
PROC CATALOG CATALOG = WORK.FORMATS ET=FORMAT ;
      MODIFY YNDKF
      (DESC = "1=Yes, 2=No, 8=DK, Other missing") ;
      CONTENTS;
RUN ;
QUIT;
```

Contents of Catalog WORK.FORMATS					
#	Name	Type	Create Date	Modified Date	Description
1	YNDKF	FORMAT	04/29/2023 14:27:18	04/29/2023 16:18:55	1=Yes, 2=No, 8=DK, Other missing

Unfortunately, FMTINFO() currently only works for SAS provided formats, so an attempt to get information on this format yielded incomplete information.

Name	Type	Category	Desc	DefW	MinW	MaxW	DefD	MinD	MaxD
YNDKF	BOTH	UNKN	NO DESC FOUND	3	1	40	0	0	39

Not all is lost, however. Obtaining the ODS output object from the PROC CATALOG step above allows us to find the missing description, which can be merged by OBJNAME onto information collected from the FMTINFO() function, dictionary.formats / dictionary.catalog tables or SAS views, and procedural output from PROC FORMAT.

```
ODS TRACE ON;
ODS OUTPUT CATALOG_RANDOM=CR;
PROC CATALOG CATALOG = WORK.FORMATS ET=FORMAT ;
      MODIFY YNDKF
```

```
       (DESC = "1=Yes, 2=No, 8=DK, Other missing") ;
      CONTENTS;
RUN ;
QUIT;
ODS OUTPUT CLOSE;
ODS TRACE OFF;

PROC PRINT DATA=CR noobs;
RUN;
```

num	objname	type	crdate	moddate	desc
1	YNDKF	FORMAT	04/29/2023 14:27:18	04/29/2023 17:24:42	1=Yes, 2=No, 8=DK, Other missing

CREATING A FORMATS RESOURCE TABLE AND CUSTOM FUNCTION (VIA PROC FCMP)

So, where does that leave us? Obtaining ODS output objects and SAS data sets using from the PROC CATALOG step above allows us to find the missing description, which can be merged by OBJNAME onto information collected from the FMTINFO() function, dictionary.formats / dictionary.catalog tables or SAS views, and procedural output from PROC FORMAT. to create a comprehensive Format Dictionary. For SAS supplied formats, there are many tools to analyze and document formats, using dictionary tables, sas views, and the fmtinfo function. For user-defined formats, using some other tools included PROC CATALOG and PROC FORMAT and specialized outputs can be rewarding.

Library Name	Member Name	Pathname	Object Name	Format Name	Format Type	Format Source	Minimum Width	Minimum Decimal Width	Maximum Width	Maximum Decimal Width	Default Width	Default Decimal Width
WORK	FORMATS		YNDKF	YNDKF	F	C	0	0	0	0	0	0

libname	memname	path	objname	fmtname	fmttype	source	minw	mind	maxw	maxd	defw	defd
WORK	FORMATS		YNDKF	YNDKF	F	C	0	0	0	0	0	0

num	objname	type	crdate	moddate	desc
1	YNDKF	FORMAT	04/29/2023 14:27:18	04/29/2023 17:24:42	1=Yes, 2=No, 8=DK, Other missing

FMTNAME	START	END	LABEL	MIN	MAX	DEFAULT	LENGTH	FUZZ	PREFIX	MULT	FILL	NOEDIT	TYPE	SEXCL	EEXCL	HLO	DECSEP	DIG3SEP	DATATYPE	LANGUAGE
DTS	-21914	-21914	Invalid	1	40	10	10	1E-12		0		0	N	N	N					
DTS	-7305	-7305	Still in	1	40	10	10	1E-12		0		0	N	N	N					
DTS	0	0	SAS zero	1	40	10	10	1E-12		0		0	N	N	N					
DTS	**OTHER**	**OTHER**	MMDDYY10.	1	40	10	10	1E-12		0		0	N	N	N	OF				
YNDKF	1	1	Yes	1	40	3	3	1E-12		0		0	N	N	N					
YNDKF	2	2	No	1	40	3	3	1E-12		0		0	N	N	N					
YNDKF	8	8	DK	1	40	3	3	1E-12		0		0	N	N	N					

Using selected variables from the data sets we have created, we can build a designer format dictionary in SAS. The example shown has limited variables, but you can choose as many elements as desired, and can build your dictionary or codebook at the format level, and/or at format value level.

```
DATA CustomFormats;
    merge work.dictformats (keep=libname memname objname objtype)
          cr (keep=objname crdate moddate desc);
      by objname;
run;

DATA CustomFormatDictValues;
    merge work.dictformats (keep=libname memname objname objtype)
         cr  (keep=objname crdate moddate desc)
         yndks (keep=fmtname start end label type
                    rename=(fmtname=objname));
    by objname;
run;
```

FMTINFO

Custom Format Dictionary - Format Level

libname	memname	objname	objtype	crdate	moddate	desc
WORK	FORMATS	DTS	FORMAT	10/19/2023 01:28:20	10/19/2023 01:28:20	
WORK	FORMATS	YNDKF	FORMAT	10/19/2023 01:28:20	10/19/2023 01:28:20	1=Yes, 2=No, 8=DK, Other missing

Custom Format Dictionary - Value Level

libname	memname	objname	objtype	crdate	moddate	desc	START	END	LABEL	TYPE
WORK	FORMATS	DTS	FORMAT	10/19/2023 01:28:20	10/19/2023 01:28:20		-21914	-21914	Invalid	N
WORK	FORMATS	DTS	FORMAT	10/19/2023 01:28:20	10/19/2023 01:28:20		-7305	-7305	Still in	N
WORK	FORMATS	DTS	FORMAT	10/19/2023 01:28:20	10/19/2023 01:28:20		0	0	SAS zero	N
WORK	FORMATS	DTS	FORMAT	10/19/2023 01:28:20	10/19/2023 01:28:20		**OTHER**	**OTHER**	MMDDYY10.	N
WORK	FORMATS	YNDKF	FORMAT	10/19/2023 01:28:20	10/19/2023 01:28:20	1=Yes, 2=No, 8=DK, Other missing	1	1	Yes	N

Using the format level data set, we can illustrate how a custom format information function could work:

```
proc fcmp outlib=dd.funcs.formatdesc listall;
 function fmt411(desc $) $ 200;
 length objname $ 32;
 if desc ne '' then description = catx(':',objname,desc);
 else description='Not Available';
 return (description);
 endfunc;
 listfunc fmt411;
quit;
options cmplib=(dd.funcs);
data fmtinfo411;
 length description $ 200;
 set customformats (keep=objname desc);
 description = fmt411(desc);
run;
proc print data=fmtinfo411 noobs;
title2 'Example of custom format for format description';
run;
```

FMTINFO
Example of custom format for format description

description	objname	desc
Not Available	DTS	
1=Yes, 2=No, 8=DK, Other missing	YNDKF	1=Yes, 2=No, 8=DK, Other missing

CONCLUSION

SAS® provides us with many tools to uncover information on the formats and informats we use. There is no single source that provides all the useful information contained in dictionary.formats / dictionary.catalog tables, SAS views, by using the FMTINFO() function, and via procedural output from PROC FORMAT and PROC CATALOG, but these sources can be combined to produce a curated report with all the desired information required, and even to write a custom formatinfo function. The FMTINFO() function itself is a valuable addition to the programmer's toolbox, and brings to light a number of different, useful resources for learning more about formats and informats.

REFERENCES

Borowiak, Kenneth W. "Additional Metadata for Common Catalog Entry Types." Proceedings of PharmaSUG 2014. April 2014. https://www.lexjansen.com/pharmasug/2014/CC/PharmaSUG-2014-CC08.pdf

Hadden, Louise S. "Putting the Meta into the Data: Managing Data Processing for a Large Scale CDC Surveillance Project with SAS®." Proceedings of SAS Global Forum 2020. https://www.lexjansen.com/wuss/2022/WUSS-2022-Paper-20.pdf

Hughes, Troy Martin. "Make You Holla' Tikka Masala: Creating User-Defined Informats Using the PROC FORMAT OTHER Option to Call User-Defined FCMP Functions That Facilitate Data Ingestion Data Quality." Proceedings of SESUG 2023. https://sesug.org/proceedings/sesug_2023_final_papers/Learning_SAS_I/SESUG2023_Paper_167_Final_PDF.pdf

Lafler, Kirk Paul. "A Hands-On Introduction to SAS® Metadata DICTIONARY Tables and SASHELP Views." Proceedings of MWSUG 2018. September 2017. https://www.lexjansen.com/mwsug/2018/HW/MWSUG-2018-HW-9.pdf

Langston, Rick. "Finding Out About Formats and Their Attributes." Proceedings of SAS Global Forum 2017. April 2017. https://support.sas.com/resources/papers/proceedings17/SAS0209-2017.pdf

Langston, Rick. "Using the New Features in PROC FORMAT." Proceedings of SAS Global Forum 2012. April 2012. https://support.sas.com/resources/papers/proceedings12/245-2012.pdf

Watson, Richann. "Have a Date with ISO®? Using PROC FCMP to Convert Dates to ISO 8601." Proceedings of SESUG 2022. November 2022. https://www.lexjansen.com/sesug/2022/SESUG2022_Paper_153_Final_PDF.pdf

Watson, Richann and Hadden, Louise S. “Functions (and More) on CALL!” Proceedings of SESUG 2022. November 2022. https://www.lexjansen.com/sesug/2022/SESUG2022_Paper_107_Final_PDF.pdf

Watson, Richann and Hadden, Louise S. “Quick, Call the “FUZZ”: Using Fuzzy Logic!” Proceedings of SAS Global Forum 2021. March 2021. https://communities.sas.com/t5/SAS-Global-Forum-Proceedings/Quick-Call-the-quot-FUZZ-quot-Using-Fuzzy-Logic/ta-p/726371

ACKNOWLEDGMENTS

SUPPORT.SAS.COM – the samples, FAQs and human beings behind the scenes are the greatest!

Rick Langston, who wrote the FMTINFO() function, and many other SAS tools we take for granted every day.

Rick Wicklin, whose “The DO Loop” SAS blog is a never-ending source of useful information and inspiration.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Louise Hadden
Abt Associates Inc.
10 Fawcett St
Cambridge, MA 02138
Email: louise_hadden@abtassoc.com

Sample code is available from the author upon request.