SCSUG 2018

Accredited, Bona Fide, Certified, Diploma'ed, and Edumacated: The ABCDEs of Automating the Validation and Monitoring of Professional Requirements for Employees and Job Candidates Through Dynamic, Data-Driven SAS® Reporting

Troy Martin Hughes

ABSTRACT

Job postings typically have stated requirements such as education, training, certifications, and other criteria. Some requirements must be met before a candidate is hired while others can be fulfilled after employment during a "grace period" of a specified duration. Because many certifications require periodic refresh or renewal, validation of professional requirements often continues after candidates are hired and throughout their careers. While validation and monitoring professional requirements for a small team might be accomplished in minutes, the effort increases with the number of employees, the diversity of employee roles, and the diversity of requirements that must be tracked. This text demonstrates a flexible, scalable, reusable macro (COMPLIANCE) that validates and monitors education, training, and professional certifications for job candidates or employees. The data-driven SAS® solution relies on an external data model (operationalized in an Excel spreadsheet) that specifies the requirements, desired achievements, and applicable grace periods. This software modularity—the separation of the data model from the underlying SAS software—ensures that the solution can be adapted to any industry, environment, or organization by modifying only the spreadsheet. COMPLIANCE creates dynamic, color-coded HTML reports that delineate personnel who meet requirements, who are within a grace period, who do not currently meet requirements but have met requirements in the past, and who have never met requirements, enabling immediate visual identification of top performers or non-compliant employees.

INTRODUCTION

Unless you're self-employed, a friend or relative of the owner, or an inside hire, most of us have secured our jobs through an application process that typically begins by responding to a job posting. And nearly all job postings include a "Requirements" section that specifies what boxes the prospective candidate must check to be considered for the position. Professional requirements can include education, degrees, certifications, accreditations, or trainings that must have been completed prior to employment, or which must be completed after hire. Many job postings additionally include a "Desirements" section that lists achievements that are preferred albeit not required. These "nice to haves"—in addition to the "must haves"—can help distinguish candidates during the application process and can be tracked after employment to continue to identify top performers.

True "requirements" are those that absolutely must be met for a candidate to be hired. For example, a company might only hire biostatisticians who have attained at least a bachelor's degree, or it might require all analysts to have the SAS Base Programmer certification before being considered for employment. In other cases, a "requirement" might be able to be met after hire during a grace period. For example, a company might require the SAS Base Programmer certification before employment but might afford analysts six months to achieve the SAS Clinical Trials Programmer certification and one year to achieve the SAS Advanced Programmer certification. While SAS certifications never expire, many certifications, licenses, or trainings do require renewals or refreshers to remain active. Thus, professional requirements typically must be assessed not only during the application process but also throughout an employee's career. Failure to maintain professional requirements can result in fines, firing, loss of a license, or even criminal or civil liability for the individual or the company.

Whether professional requirements are tracked by human resources, by functional managers, by employees themselves, or by all of these, the task can be difficult. An employee might have the greatest awareness of his own requirements but might fail to realize when a six-month grace period is approaching (or has passed) for a specific requirement. A functional manager might be focused on domain-specific requirements (e.g., SAS certifications or other role-specific certifications) for his employees, but not on company-specific requirements (e.g., annual sexual harassment training). And although human resources might be best equipped to monitor all professional requirements

for all employees, this can become overwhelming as the number and complexity of employees increase, especially when monitoring is not automated and given the normal ebb and flow of employees into and out of an organization.

This text introduces the COMPLIANCE macro (see Appendix A), a data-driven solution that validates and monitors professional requirements that is generalizable across diverse industries, organizations, teams, and positions. The macro uses an employee personnel roster (simulated within an Excel workbook) to obtain job titles. A control table (also an Excel workbook) contains a matrix of applicable requirements by position and any applicable grace periods. Achievements are maintained in a document library and, when compared against the requirements for each position, the macro builds reports that demonstrate compliance while also highlighting non-compliance within an organization.

THE PERSONNEL ROSTER

Validation and monitoring of professional requirements begins with a list of employees (or job candidates). For simplicity, the remaining text and examples refer to "employees" only and include a hire date, although the COMPLIANCE macro is equally applicable to job candidates and can be used to track successful candidates from the application phase through onboarding and throughout their careers. Table 1 demonstrates a sample employee roster led fearlessly by the project manager, Ron Burgundy. This roster could be maintained in a database, Excel spreadsheet, XML file, SAS data set, or other file format or application. The COMPLIANCE macro requires only three fields (i.e., employee ID, hire date, and job classification) to enforce business rules and validate staff; however, the employee's name and job title are also included to improve report readability. In actual implementation, the inclusion of other fields such as employee division or team would further improve the customization and specification of reports.

Emp ID	First Name	Last Name	Hire Date	Job Title	Job Class
1	Ron	Burgundy	3/4/2009	Project Manager	project manager
2	Sky	Corrigan	1/4/2018	Data Analyst	data analyst (SAS)
3	Chazz	Michaels	8/1/2017	Data Analyst	data analyst (SAS)
4	Franz	Liebkind	1/4/2015	Software Developer	software developer (SAS)
5	Harold	Crick	4/12/2017	Software Developer	software developer (SAS)
6	Ricky	Bobby	12/4/2013	Software Developer	software developer (general)
7	Jacobim	Mugatu	11/4/2017	Senior Software Developer	senior software developer (SAS)

Table 1. Application-Agnostic Employee Roster

Figure 1 demonstrates Table 1 as an Excel workbook (Personnel.xlsx) and saved as D:\sas\compliance\personnel.xlsx. Note that the spreadsheet must be named "personnel" (which is case-sensitive within the IMPORT procedure).

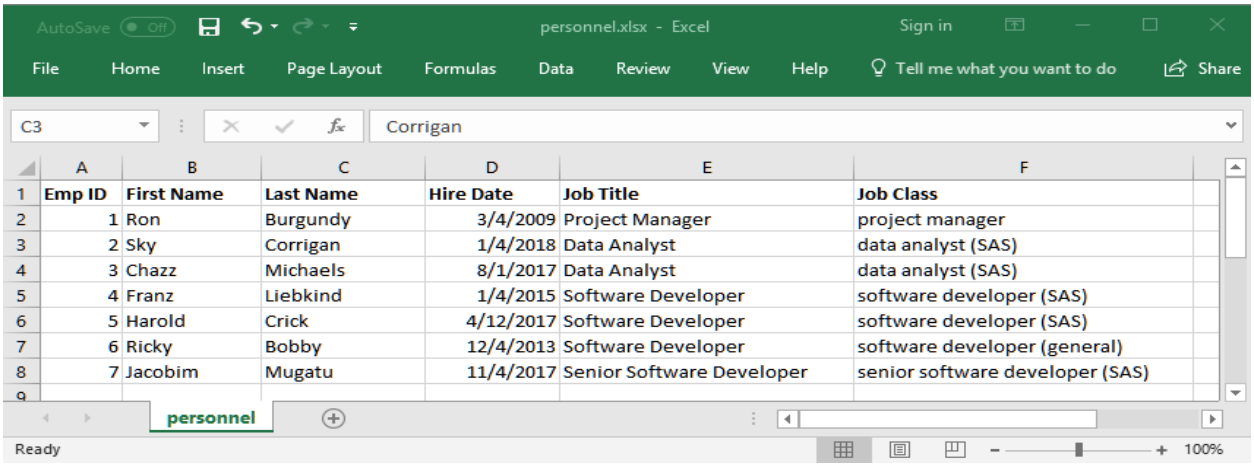

Figure 1. Excel Workbook (Personnel.xlsx) Containing Employee Roster

In this example, the Personnel workbook should be created in the same folder in which the Compliance.sas macro is saved, after which the COMPLIANCE macro imports the workbook to create the Personnel SAS data set. However, to simulate this data import without Excel, the equivalent data set can be created in SAS using the DATALINES option within the DATA step:

```
data personnel;
   infile datalines delimiter=',';
   length Emp_ID 8 First_Name $50 Last_Name $50 Hire_Date 8 Job_Title $50
Job_Class $50;
   input Emp_ID First_Name $ Last_Name $ Hire_Date :mmddyy10. Job_Title $
Job_Class $;
   format hire_date mmddyy10.;
   datalines;
1, Ron, Burgundy, 3/4/2009, Project Manager, project manager
2, Sky, Corrigan, 1/4/2018, Data Analyst, data analyst (SAS)
3, Chazz, Michaels, 8/1/2017, Data Analyst, data analyst (SAS)
4, Franz, Liebkind, 1/4/2015, Software Developer, software developer (SAS)
5, Harold, Crick, 4/12/2017, Software Developer, software developer (SAS)
6, Ricky, Bobby, 12/4/2013, Software Developer, software developer (general)
7, Jacobim, Mugatu, 11/4/2017, Senior Software Developer, senior software
developer (SAS)
;
```

The sample Personnel data set contains the following variables, with “required” fields necessary for the functionality of COMPLIANCE business rules, and “optional” fields necessary only for the functionality of the sample report within COMPLIANCE:

- **Emp_ID** (required) – This unique key represents the numeric value used to identify employees.
- **First_Name** (optional) – The first name of the employee supplements the compliance report.
- **Last_Name** (optional) – The last name of the employee supplements the compliance report.
- **Hire_Date** (required) – The date of hire facilitates the use of grace periods—effectively the number of months after hire by which a requirement must have been met. Because employees are often promoted from one position to another, a transfer date (rather than hire date) is often more aptly tied to grace periods. For example, a data analyst might be hired in 2015 and meet all professional requirements, after which she is promoted to a biostatistician role in 2017. In this later position, she might be required to have attained the SAS Business Analyst certification within a six-month grace period—but this grace period would typically be measured not from her hire date but from her promotion date. Thus, while hire date is referenced throughout this text and encoded within the COMPLIANT macro, this variable should usually be mapped to the date the current position was begun.
- **Job_Title** (optional) – This variable represents the job title of the employee and is used to illustrate that job title alone may not sufficiently reflect the respective professional requirements, which instead is tied to the Job_Class variable. For example, within Table 1, both Harold Crick and Ricky Bobby are software engineers in title; however, Harold is a SAS specialist while Ricky is a general developer, thus their professional requirements are tied to their respective job classifications, not job titles. This type of distinction is also common in larger organizations in which positions bearing the same job title but appearing in separate teams or divisions might have vastly different roles, responsibilities, and requirements.
- **Job_Class** (required) – This variable shows the job classification, essentially differentiating positions based on requirements alone rather than job title. The values in the Job_Class variable must correspond to the values listed in the Professional Requirements matrix, demonstrated in the next section. For example, Ricky Bobby’s and Harold Crick’s respective job classifications (“software developer (SAS)” and “software developer (general)”) correspond to entries in the Job_Class variable in Table 2.

Note that employees must be uniquely identified (by Emp_ID) within the Personnel workbook (or other source for personnel data). Thus, if a transactional employee database is being queried that contains historical records (e.g., past positions that predate current roles) or records for inactive employees (e.g., employees who have died, resigned, or been terminated), only the current employee record should be selected.

THE PROFESSIONAL REQUIREMENTS MATRIX

The requirements matrix provides the business rules that facilitate tracking required and relevant degrees, certifications, training, and other achievements. Requirements are met through achievements, which can be temporary (in which a requirement is met for a finite period, after which the achievement must be renewed or refreshed) or permanent (in which the achievement never expires). Requirements can also be required at the time of hire (or promotion, transfer, or other job reclassification) or within some defined grace period (e.g., six months from the date of hire). Table 2 demonstrates a sample requirements matrix that specifies the business rules for the positions listed in Table 1.

Requirement	Abbreviation	Job Class	Grace Period
Bachelor's Degree	BABS	all	0
SAS Certified Base Programmer	SAS-Base	data analyst (SAS)	6
		software developer (SAS)	3
		senior software developer (SAS)	0
		project manager	
SAS Certified Advanced Programmer	SAS-Adv	senior software developer (SAS)	6
Project Management Professional (PMP)	PMP	project manager	0
Certified Secure Software Lifecycle Professional (CSSLP)	CSSLP	project manager	

Table 2. Application-Agnostic Professional Requirements Matrix

Figure 2 demonstrates Table 2 operationalized within an Excel workbook (Requirements.xlsx) and saved as D:\sas\compliance\requirements.xlsx. Note that the spreadsheet has been named “requirements” (which is case-sensitive within the IMPORT procedure).

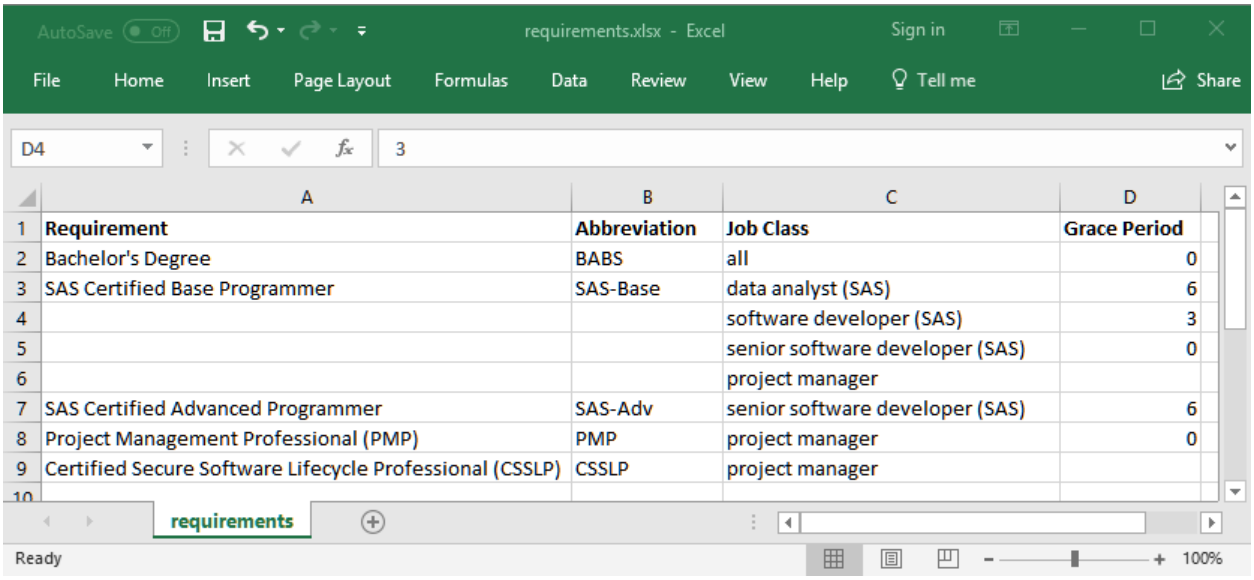

Figure 2. Excel Workbook (Requirements.xlsx) Containing Professional Requirements Matrix

In this example, the Requirements workbook should be created in the same folder in which the Compliance.sas macro is saved, after which the COMPLIANCE macro imports the workbook to create the Requirements SAS data set. However, to simulate this data import without Excel, the equivalent data set can be created in SAS using the DATALINES option within the DATA step:

```
data requirements;
   infile datalines delimiter=',';
   length Requirement $100 Abbreviation $16 Job_Class $50 Grace_Period 8;
   input Requirement $ Abbreviation $ Job_Class $ Grace_Period;
   datalines;
Bachelor's Degree, BABS, all, 0
SAS Certified Base Programmer, SAS-Base, data analyst (SAS), 6
 , ,software developer (SAS), 3
 , ,senior software developer (SAS), 0
 , ,project manager,
SAS Certified Advanced Programmer,SAS-Adv, senior software developer (SAS), 6
Project Management Professional (PMP), PMP, project manager, 0
Certified Secure Software Lifecycle Professional (CSSLP), CSSLP, project
manager,
;
```

The sample Requirements data set contains the following variables, with “required” fields necessary for the functionality of COMPLIANCE business rules, and “optional” fields necessary only for the functionality of the sample report within COMPLIANCE:

- **Requirement** (optional) – This variable describes the requirement; it is not used to assess requirements but is used in compliance reporting. Note that if multiple observations represent the same requirement (as applied to different job classifications), then only the first observation *must* contain the requirement, although all observations *can* contain the requirement. This option is demonstrated in the three lines within Figure 2 in which the Requirement value is blank.

- **Abbreviation** (required) – This variable is used to identify the requirement within the file name of the achievement documentation. For example, if a non-specific bachelor’s degree is required for a position, a scanned or digital copy of the bachelor’s certificate can be saved in a file in which the file name includes BABS (i.e., BA or BS degree). Each abbreviation must uniquely identify one requirement. Note that if multiple observations represent the same requirement (as applied to different job classifications), then only the first observation *must* contain the abbreviation, although all observations *can* contain the abbreviation. This option is demonstrated in the three lines within Figure 2 in which the Requirement value is blank. Note that the Abbreviation variable is case-sensitive only insofar as the case included in the Requirements workbook is demonstrated in compliance reports; however, case is not sensitive when matching achievements documentation so file names with “BABS” or “babs” will each map to the BABS job classification.

- **Job_Class** (required) –The job classification must match the Job_Class values within the personnel roster; however, specifying “All” indicates that the requirement is applicable to all personnel regardless of job classification. Each observation must contain a job classification. The Requirements data set essentially contains a composite primary key that includes both Abbreviation and Job_Class, so for each Abbreviation, no Job_Class should ever be duplicated. Note that job classification is case sensitive only insofar as the case included in the Requirements workbook is demonstrated in compliance reports; however, case is not sensitive when joining the Personnel and Requirements workbooks.

- **Grace_Period** (required) – The grace period is a required variable for the COMPLIANCE macro, but no values must be included. The grace period references the number of months after the hire date by which the associated requirement must have been met. A grace period of 0 reflects the requirement must be met before the hire date while an empty grace period reflects that the “requirement” is relevant (or preferred) but not truly

required. Preferred requirements are tracked in compliance reporting and can be used to distinguish employees. For example, Figure 2 reflects that while project managers are not required to attain the SAS Base Programmer certification, this preferred requirement will still be tracked for all project managers, perhaps because the technical knowledge would benefit anyone managing a team of SAS practitioners.

Table 3 demonstrates the Requirements data set after the missing (i.e., implied) Requirement and Abbreviation values have been imputed.

Requirement	Abbreviation	Job Class	Grace Period
Bachelor's Degree	BABS	all	0
SAS Certified Base Programmer	SAS-Base	data analyst (SAS)	6
SAS Certified Base Programmer	SAS-Base	software developer (SAS)	3
SAS Certified Base Programmer	SAS-Base	senior software developer (SAS)	0
SAS Certified Base Programmer	SAS-Base	project manager	
SAS Certified Advanced Programmer	SAS-Adv	senior software developer (SAS)	6
Project Management Professional (PMP)	PMP	project manager	0
Certified Secure Software Lifecycle Professional (CSSLP)	CSSLP	project manager	

Table 3. Application-Agnostic Professional Requirements Matrix

Note that Table 3 indicates that a bachelor's degree is required by all personnel at the time of hire, specified by the "All" in the job classification field and the corresponding "0" in the grace period.

ACHIEVEMENTS DOCUMENTATION

One of the simplest methods to validate that employee requirements are met is to maintain a digital archive of all diplomas, certifications, and other documentation that demonstrates the date on which each achievement occurred. In the case of temporary achievements, such as trainings that must be retaken annually or certifications that expire, the document should also indicate the expiration date. By standardizing the file naming convention for all achievement-related documents within an organization, requirements compliance can be automated more readily because the employee, requirement, achievement attainment date, and achievement expiration date (if applicable) each can be assessed by parsing only the file name without the necessity to open or parse the document itself.

COMPLIANCE requires the following file-naming convention for all achievements that have an expiration date:

```
EmployeeID_Abbreviation_AchievementDate_ExpirationDate
```

COMPLIANCE requires the following file-naming convention for all achievements that do not have an expiration date:

```
EmployeeID_Abbreviation_AchievementDate
```

Because only file names are parsed, file extensions of any format can be optionally added. Thus, the file name "1_PMP_20141010_20171009.pdf" is as valid as "1_PMP_20141010_20171009" that contains no extension, yet for clarity, only TXT file extensions are demonstrated in this text. The values that are tokenized within file names are separated by underscore characters (_) and include:

- **EmployeeID** (required) – This value corresponds to the numeric employee ID within the personnel roster, thus the Emp_ID variable within the Personnel data set.
- **Abbreviation** (required) – This value corresponds to the achievement abbreviation within the professional requirements matrix, thus the Abbreviation variable within the Requirements data set. The abbreviation is not case sensitive within file names.

- **AchievementDate** (required) – The date of the achievement must be specified in YYYYMMDD format.

- **ExpirationDate** (optional) – The date that the achievement expires (if applicable) should be specified in YYYYMMDD format. Some achievements, such as a degree or SAS certification, do not expire, so the AchievementDate should be omitted.

The COMPLIANCE macro relies on the PARSEDIR macro (see Appendix B) to identify all achievement-related documents located in the Docs folder, so in this example, all documents should be stored in this folder. However, because PARSEDIR can optionally parse not only folders but also subdirectories thereof, documents could be stored separately by type (e.g., degree vs certification), expiration status (e.g., achievements that expire vs those that don't), team, division, or any other distinction that can be made within the folder structure to support readability and maintainability of the archive. PARSEDIR is not discussed further in this text although its SUBDIR parameter specifies whether subdirectories should be examined.

Table 4 demonstrates the text files that should be created for this example within the Docs folder. Additionally, the relationship between each achievement and its associated requirement is listed. The text files should be created as empty files, as their contents are not interrogated; however, in practice, these files would represent the documentation that could be inspected to manually confirm that associated requirements had been met. Thus, in this example, the first file in Table 4 would be saved as D:\sas\compliance\docs\1_CSSLP_20160812_20190811.txt.

File Name	Relationship to Requirement
1_CSSLP_20160812_20190811.txt	optional certificate for all project managers
1_PMP_20081021_20111020.txt	required certificate (at hire) for all project managers (1st expired)
1_PMP_20111021_20141020.txt	required certificate (at hire) for all project managers (2nd expired)
1_PMP_20141021_20171020.txt	required certificate (at hire) for all project managers (3rd expired)
1_BABS_19820623.txt	required degree (at hire) for all staff
4_SAS-Adv_20161031.txt	irrelevant certificate for SAS software developers
4_SAS-Base_20150601.txt	required certificate (after 3 months) for SAS software developers
7_SAS-Base_20120606.txt	required certificate (at hire) for SAS senior software developers

Table 4. Sample Achievement-Related Text Files Created in the Docs Folder

Note that in Table 4, Franz Liebkind (Employee #4) optionally attained the SAS Advanced Programmer certification; however, because the requirements matrix (Tables 2 and 3) doesn't associate this certification with more junior-level SAS developers, the achievement is considered irrelevant. This oversight (within construction of the business rules of requirements matrix) is overcome later in this text.

COMPLIANCE MACRO SETUP AND INVOCATION

Minimal setup is required to run COMPLIANCE in SAS 9.4 for Windows. These steps include:

1. Download the COMPLIANCE macro (see Appendix A) and save as Compliance.sas. In these examples, the program is saved in the folder D:\sas\compliance, although the location is flexible.

2. Download the PARSEDIR macro (see Appendix B) and save as Parsedir.sas. The location of the PARSEDIR macro is flexible but must be indicated in the header of the COMPLIANCE macro.

3. Within the COMPLIANCE macro, modify the &LOCATION global macro variable to point toward the location of Parsedir.sas. Note that the final character should be a slash (forward or backward, depending on the operating system).

4. Thereafter, COMPLIANCE can be referenced with an %INCLUDE statement or through use of the SAS Autocall Macro Facility.

5. COMPLIANCE requires a personnel roster and requirements control table to run; samples are demonstrated as Tables 1 and 2, and Figures 1 and 2.

6. COMPLIANCE requires a folder from which achievements documentation (e.g., diplomas, certifications, training documenting, etc.) can be pulled. COMPLIANCE also requires a folder in which HTML compliance reports will be generated.

Additional setup is required to run the example in this text:

1. Create the Personnel spreadsheet by copying and pasting Table 1 and saving as Personnel.xlsx. For this text, the spreadsheet is saved in D:\sas\compliance, although this location is flexible. The spreadsheet tab must be named “personnel” as this is referenced in the SHEET statement of the IMPORT procedure.

2. Create the Requirements spreadsheet by copying and pasting Table 2 and saving as Requirements.xlsx. For this text, the spreadsheet is saved in D:\sas\compliance. The spreadsheet tab must be named “requirements” as this is referenced in the SHEET statement of the IMPORT procedure.

3. Create the subfolder Docs, which in this example corresponds to D:\sas\compliance\docs.

4. Create the subfolder Reports, which in this example corresponds to D:\sas\compliance\reports.

5. Individually create the text files listed in TABLE 3 and save in the Docs folder. Thus, in this example, the first file in the table would be saved as D:\sas\compliance\docs\1_BABS_19820623.txt.

The COMPLIANCE definition follows, which includes four required parameters followed by nine optional parameters (as well as their default values) that specify the report color schema:

```
%macro compliance(docsloc= /* folder with docs (or subfolders with docs) */,
   rptloc= /* folder in which compliance report is produced */,
   personfile= /* folder and file name for Personnel workbook (XLSX format) */,
   reqfile= /* folder and file name for Requirements matrix (XLSX format) */,
   req_met=light green /* color for current requirements that are met */,
   req_exp=very light red /* color for current reqs that are expired */,
   req_nev=light red /* color for current requirements never met */,
   reqfut_met=light green /* color for future requirements that are met */,
   reqfut_no=very light yellow /* color for future reqs (grace period) */,
   pref_met=very light green /* color for preferences that are met */,
   pref_exp=very light gray /* color for preferences that are expired */,
   pref_nev=very light gray /* color for preferences never met */,
   req_irr=light gray /* color for irrelevant achievements */);
```

Once the preceding steps have been followed, the COMPLIANCE macro can be invoked with a few lines of code. The following sample invocation specifies the Docs folder location (DOCSLOC), report folder location (RPTLOC), personnel roster folder and file name (PERSONFILE), and requirements matrix folder and file name (REQFILE):

```
* location must be changed to actual folder and include trailing slash;
%let location=d:\sas\compliance\;
%include "&location.compliance.sas";

%compliance(docsloc=&location.docs,
   rptloc=&location.reports,
   personfile=&location.personnel.xlsx,
   reqfile=&location.requirements.xlsx);
```

This sample invocation produces the color-coded HTML report Professional_requirements (utilizing all default colors), demonstrated in Figure 3.

#	Emp_ID	First_Name	Last_Name	Hire_Date	Job_Title	Job_Class	BABS	CSSLP	PMP	SAS-Adv	SAS-Base
1	1	Ron	Burgundy	03/04/2009	Project Manager	project manager	06/23/1982 (no exp)	08/12/2016 (exp 08/11/2019)	10/21/2014 (exp 10/20/2017)		
2	2	Sky	Corrigan	01/04/2018	Data Analyst	data analyst (SAS)					
3	3	Chazz	Michaels	08/01/2017	Data Analyst	data analyst (SAS)					
4	4	Franz	Liebkind	01/04/2015	Software Developer	software developer (SAS)				10/31/2016 (no exp)	06/01/2015 (no exp)
5	5	Harold	Crick	04/12/2017	Software Developer	software developer (SAS)					
6	6	Ricky	Bobby	12/04/2013	Software Developer	software developer (general)					
7	7	Jacobim	Mugatu	11/04/2017	Senior Software Developer	senior software developer (SAS)					06/06/2012 (no exp)

Figure 3. Sample Report (Professional_requirements.html) with Compliance Focus

If an achievement has been attained, the achievement date (and expiration date, if applicable) is displayed in the appropriate cell. For example, Ron Burgundy completed his undergraduate studies in 1982 and this bachelor's degree has no expiration. Ron's CSSLP is a slightly lighter green, representing that this is an optional (preferred) certification rather than a required one. His PMP is shown in light red, indicating that he has had the certification in the past, but it has since expired. This contrasts with brighter red cells that indicate requirements that have never been met. The two yellow cells highlight grace periods that will be expiring. Ron's light gray cell indicates that the SAS Base Programmer certification is relevant yet optional, whereas the remaining dark gray cells indicate irrelevant requirements to those respective job classifications. However, all business rules can be modified through the requirements matrix while the optional color parameters and their role in shaping report focus and intent is discussed in the following section.

COMPLIANCE REPORT CUSTOMIZATION

Nine optional parameters control the color-coding schema for the COMPLIANCE macro, including:

- **REQ_MET** – A current requirement has been met. This doesn't speak to whether the requirement was met on time, rather only that it is currently met.
- **REQ_EXP** – A current requirement is not currently met because the corresponding achievement has expired. This demonstrates an employee who is no longer compliant but who may still possess the knowledge or skills associated with the requirement.
- **REQ_NEV** – A current requirement exists for a position and that requirement has never been met.
- **REQFUT_MET** – A future requirement has been met.
- **REQFUT_NO** – A future requirement either has never been met or it has been met previously but has since expired. Regardless, the employee is within the designated grace period and is compliant.
- **PREF_MET** – A preferred requirement has been met.
- **PREF_EXP** – A preferred requirement has been met in the past but has since expired.
- **PREF_NEV** – A preferred requirement has never been met.
- **REQ_IRR** – A specific achievement is irrelevant (i.e., neither required nor preferred for a position).

The default focus of the report (using default parameter color-coding values) is compliance—i.e., the clear delineation between either meeting or failing to meet specific professional requirements. Thus, less emphasis is paid to preferred requirements, because presumably those should not affect compliance during an audit. Thus, in Figure 3, because Ron's PMP has expired, this is counted against him (and the cell is turned red) because this is an actual requirement for the project manager job classification.

In other cases, however, the intent of auditing professional requirements is less about validating compliance and more about differentiating top employees by their skills, knowledge, or achievements. Thus, in this sense, an expired certificate may demonstrate a higher skill level than someone who has never attained that certification. Conversely, with this different focus, an employee who has attained some achievement before it is required might be viewed more highly than an employee who has not, despite the requirement not yet being due. The following COMPLIANCE invocation focuses more on attainment of achievements rather than strict compliance.

```
%compliance(docsloc=&location.docs,
   rptloc=&location.reports,
   personfile=&location.personnel.xlsx,
   reqfile=&location.requirements.xlsx,
   req_met=light green,
   req_exp=very light green,
   req_nev=light red,
   reqfut_met=light green,
   reqfut_no=very light red,
   pref_met=very light green,
   pref_exp=very light gray,
   pref_nev=very light gray,
   req_irr=light gray);
```

The output from COMPLIANCE is demonstrated in Figure 4 and shows this slight change in report focus.

#	Emp _ID	First_Name	Last_Name	Hire_Date	Job_Title	Job_Class	BABS	CSSLP	PMP	SAS-Adv	SAS-Base
1	1	Ron	Burgundy	03/04/2009	Project Manager	project manager	06/23/1982 (no exp)	08/12/2016 (exp 08/11/2019)	10/21/2014 (exp 10/20/2017)		
2	2	Sky	Corrigan	01/04/2018	Data Analyst	data analyst (SAS)					
3	3	Chazz	Michaels	08/01/2017	Data Analyst	data analyst (SAS)					
4	4	Franz	Liebkind	01/04/2015	Software Developer	software developer (SAS)				10/31/2016 (no exp)	06/01/2015 (no exp)
5	5	Harold	Crick	04/12/2017	Software Developer	software developer (SAS)					
6	6	Ricky	Bobby	12/04/2013	Software Developer	software developer (general)					
7	7	Jacobim	Mugatu	11/04/2017	Senior Software Developer	senior software developer (SAS)					06/06/2012 (no exp)

Figure 4. Sample Report (Professional_requirements.html) with Knowledge/Achievement Focus

COMPLIANCE BUSINESS RULES CUSTOMIZATION

At the heart of COMPLIANCE lie the business rules codified within the requirements matrix and one of the principle benefits of data-driven design is the extraction of these rules from software, thus supporting modularity and maintainability, not only of the software but also of these business rules. It was previously mentioned that Franz Liebkind went out of his way to attain the SAS Advanced Programmer certification, despite this not being a requirement of his position. Yet, if Franz wants to advance to a senior SAS programmer position, the requirements matrix shows that the SAS Advanced Programmer certification is required within six months of hire (i.e., advancement to that position). Given this succession, it would be beneficial to list this certification as preferred (but not required) under the job classification for junior SAS programmers (i.e., software developer (SAS)). And with external data models, this is extremely easy to accomplish.

Table 5 demonstrates this improvement, aligning the data model more closely with the reality that the SAS Advanced Programmer certification would be relevant and beneficial to more junior-level SAS practitioners. Note that only one additional record needed to be added to the workbook (or whatever artifact maintains the requirements matrix).

Requirement	Abbreviation	Job Class	Grace Period
Bachelor's Degree	BABS	all	0
SAS Certified Base Programmer	SAS-Base	data analyst (SAS)	6
		software developer (SAS)	3
		senior software developer (SAS)	0
		project manager	
SAS Certified Advanced Programmer	SAS-Adv	senior software developer (SAS)	6
		software developer (SAS)	
Project Management Professional (PMP)	PMP	project manager	0
Certified Secure Software Lifecycle Professional (CSSLP)	CSSLP	project manager	

Table 5. Application-Agnostic Professional Requirements Matrix (Updated Business Rules)

Once the Requirements workbook has been saved and updated, the COMPLIANCE macro can be immediately rerun to apply the revised business rules:

```
%compliance(docsloc=&location.docs,
   rptloc=&location.reports,
   personfile=&location.personnel.xlsx,
   reqfile=&location.requirements.xlsx);
```

The macro invocation produces a revised report, demonstrated in Figure 5.

#	Emp_ID	First_Name	Last_Name	Hire_Date	Job_Title	Job_Class	BABS	CSSLP	PMP	SAS-Adv	SAS-Base
1	1	Ron	Burgundy	03/04/2009	Project Manager	project manager	06/23/1982 (no exp)	08/12/2016 (exp 08/11/2019)	10/21/2014 (exp 10/20/2017)		
2	2	Sky	Corrigan	01/04/2018	Data Analyst	data analyst (SAS)					
3	3	Chazz	Michaels	08/01/2017	Data Analyst	data analyst (SAS)					
4	4	Franz	Liebkind	01/04/2015	Software Developer	software developer (SAS)				10/31/2016 (no exp)	06/01/2015 (no exp)

#	Emp _ID	First_Name	Last_Name	Hire_Date	Job_Title	Job_Class	BABS	CSSLP	PMP	SAS-Adv	SAS-Base
5	5	Harold	Crick	04/12/2017	Software Developer	software developer (SAS)					
6	6	Ricky	Bobby	12/04/2013	Software Developer	software developer (general)					
7	7	Jacobim	Mugatu	11/04/2017	Senior Software Developer	senior software developer (SAS)					06/06/2012 (no exp)

Figure 5. Sample Report (Professional_requirements.html) with Compliance Focus (Updated Business Rules)

The revised business rules now more accurately depict the reality of the requirements. Not only is Franz now highlighted for his accomplishment, but Harold Crick's SAS Advanced Programmer certification cell has turned light gray, indicating that it is not required, but is relevant and should be tracked.

EXCEPTION MANAGEMENT

Wherever requirements exist, there will also exist exceptions to those requirements, either those specified explicitly or tacitly accepted. Documenting requirement exceptions can be a straightforward process that involves composing a document that memorializes the exception (and which might seek the signature or authorization of the employee or some key stakeholder). For example, Figure 5 demonstrates that Sky Corrigan may not have (or has not provided) proof of his bachelor's degree. But as a relatively new employee, Sky may have been hired with the express understanding that he would have his degree completed by December 2018. This temporary exception, even if it's only an email memorializing this intent between Sky and his manager, Ron, can be captured in a document where it will be interpreted by COMPLIANCE as a temporary achievement.

To simulate adding this exception document, the following empty text file can be created in the Docs folder:

```
2_BABS_20180103_20181231.txt
```

As soon as the exception file is added to the folder, COMPLIANCE can be rerun with the familiar invocation:

```
%compliance(docsloc=&location.docs,
   rptloc=&location.reports,
   personfile=&location.personnel.xlsx,
   reqfile=&location.requirements.xlsx);
```

Figure 6 now partially shows the updated report, demonstrating that the exception was processed, and that Sky's BABS cell updated to green. Moreover, whether Sky meets or fails to meet the December deadline to receive his degree, the exception will be recorded for posterity. Most importantly, the ability to update records swiftly and accurately (without the necessity to update the underlying code or, in this case, even the business rules) contributes directly to a more timely and complete view of requirements compliance.

#	Emp_ID	First_Name	Last_Name	Hire_Date	Job_Title	Job_Class	BABS	CSSLP	PMP	SAS-Adv	SAS-Base
1	1	Ron	Burgundy	03/04/2009	Project Manager	project manager	06/23/1982 (no exp)	08/12/2016 (exp 08/11/2019)	10/21/2014 (exp 10/20/2017)		
2	2	Sky	Corrigan	01/04/2018	Data Analyst	data analyst (SAS)	01/03/2018 (exp 12/31/2018)				

Figure 6. Sample Report (Professional_requirements.html) with Compliance Focus (Updated with Exception)

CONCLUSION

Professional requirements monitoring is a critical human resources function, not only during the job application and onboarding phases, but also often throughout an employee's career. Yet compliance monitoring can be thwarted by shifting personnel, changing requirements, a large or complex organization, and the general difficulties associated with performing audits manually. The COMPLIANCE macro represents a flexible SAS solution, generalizable to diverse industries, teams, and personnel, and scalable to organizations both big and small. Moreover, the modular nature of the macro preserves the data model and all business rules outside of the software, ensuring that both the personnel roster and requirements matrix can be maintained independent of the underlying code. Thus, this data-driven design benefits not only those stakeholders responsible for monitoring but also those responsible for maintaining the underlying code that drives these processes.

CONTACT INFORMATION

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

Name: Troy Martin Hughes
E-mail: troymartinhughes@gmail.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

APPENDIX A. COMPLIANCE MACRO

```
* change this location based on local configuration;
%let location=d:\sas\compliance\;

* this must point to the location of Parsedir.sas;
%include "&location.parsedir.sas";


%macro compliance(docsloc= /* folder with docs (or subfolders with docs) */,
   rptloc= /* folder in which compliance report is produced */,
   personfile= /* folder and file name for Personnel workbook (XLSX format) */,
   reqfile= /* folder and file name for Requirements matrix (XLSX format) */,
   req_met=light green /* color for current requirements that are met */,
   req_exp=very light red /* color for current reqs that are expired */,
   req_nev=light red /* color for current requirements never met */,
   reqfut_met=light green /* color for future requirements that are met */,
   reqfut_no=very light yellow /* color for future reqs (grace period) */,
   pref_met=very light green /* color for preferences that are met */,
   pref_exp=very light gray /* color for preferences that are expired */,
   pref_nev=very light gray /* color for preferences never met */,
   req_irr=light gray /* color for irrelevant achievements */);
%local reqlist classlist reqcount i;

*** REQUIREMENTS ***;

proc import datafile="&reqfile"
   out=requirements
   dbms=XLSX
   replace;
   sheet='requirements';
run;
proc sql noprint;
   select distinct job_class into :classlist separated by '*'
   from requirements
   where strip(lowcase(job_class))^='all';
   select distinct abbreviation into :reqlist separated by '*'
   from requirements;
   select count(distinct requirement) into :reqcount trimmed
   from requirements;
   quit;
data reqs1 (drop=oldReq oldAbbrev i);
   set requirements end=eof;
   length oldReq $100 oldAbbrev $100;
   if missing(grace_period) then grace_period=-1;
   if _n_^=1 then do;
      if missing(requirement) then requirement=oldReq;
      if missing(abbreviation) then abbreviation=oldAbbrev;
      end;
   if strip(lowcase(job_class))='all' then do;
      i=1;
      do while(lengthn(scan("&classlist",i,'*'))>0);
         job_class=scan("&classlist",i,'*');
         output;
         i=i+1;
         end;
```

```
      end;
   output;
   oldReq=requirement;
   oldAbbrev=abbreviation;
   retain oldReq oldAbbrev;
run;
proc sort data=reqs1;
   by job_class;
run;
data reqs2 (drop=requirement abbreviation grace_period);
   set reqs1;
   by job_class;
   length reqdate1-reqdate&reqcount 8;
   if first.job_class then do;
      %do i=1 %to &reqcount;
         reqdate&i=.;
         %end;
      end;
   %do i=1 %to &reqcount;
      if abbreviation="%scan(&reqlist,&i,*)" then reqdate&i=grace_period;
      %end;
   if last.job_class then output;
   retain reqdate1-reqdate&reqcount;
run;

*** PERSONNEL ***;

proc import datafile="&personfile"
   out=personnel
   dbms=XLSX
   replace;
   sheet='personnel';
run;
proc sort data=personnel;
   by job_class;
run;
data validate1;
   merge personnel (in=a) reqs2 (in=b);
   by job_class;
run;
data validate2;
   set validate1;
   format reqdate1-reqdate&reqcount mmddyy10.;
   %do i=1 %to &reqcount;
      if missing(reqdate&i) then reqdate&i=.;
      else if reqdate&i=0 then reqdate&i=hire_date;
      else if reqdate&i>0 then reqdate&i=intnx('month',hire_date,reqdate&i);
      else if reqdate&i<0 then reqdate&i='31dec2099'd;
      %end;
run;

*** ACHIEVEMENTS ***;

%parsedir(dir=&docs, dsn=docs, subdir=YES, filetype=);
data docs1 (drop=file_name);
```

```
   set docs (keep=file_name);
   length Emp_ID 8 Abbreviation $20 date1 8 date2 8;
   * remove file extension (any text after first period);
   if find(file_name,'.')>1 then
      file_name=substr(file_name,1,length(scan(file_name,1,.)));
   if lengthn(scan(file_name,1,'_'))>0 then
      emp_id=input(scan(file_name,1,'_'),8.);
   if lengthn(scan(file_name,2,'_'))>0 then abbreviation=scan(file_name,2,'_');
   if lengthn(scan(file_name,3,'_'))>0 then
      date1=input(scan(file_name,3,'_'),yymmdd8.);
   if lengthn(scan(file_name,4,'_'))>0 then
      date2=input(scan(file_name,4,'_'),yymmdd8.);
   else date2=.; * set non-expiring docs to missing;
run;
proc sort data=docs1;
   by emp_id abbreviation date1;
run;
data docs2;
   set docs1;
   by emp_id abbreviation date1;
   if last.abbreviation;
run;
data docs3 (drop=abbreviation date1 date2);
   set docs2;
   by emp_id;
   if first.emp_id then do;
      %do i=1 %to &reqcount;
         dt1doc&i=.;
         dt2doc&i=.;
         %end;
      end;
   length dt1doc1-dt1doc&reqcount 8 dt2doc1-dt2doc&reqcount 8;
   format dt1doc1-dt1doc&reqcount mmddyy10. dt2doc1-dt2doc&reqcount mmddyy10.;
   %do i=1 %to &reqcount;
      if lowcase(abbreviation)="%lowcase(%scan(&reqlist,&i,*))" then do;
         dt1doc&i=date1;
         dt2doc&i=date2;
         end;
      %end;
   if last.emp_id then output;
   retain dt1doc1-dt1doc&reqcount dt2doc1-dt2doc&reqcount;
run;
proc sort data=validate2;
   by emp_id;
run;
data compliance;
   merge validate2 (in=a) docs3 (in=b);
   by emp_id;
run;

*** REPORTING ***;

ods html path="&rptloc" file="professional_requirements.html";
title;
proc report data=compliance nocenter nowindows nocompletecols
```

```
      style(report)=[foreground=black backgroundcolor=white background=black]
      style(header)=[font_size=2 background=black backgroundcolor=black
      foreground=white]
      style(column)=[backgroundcolor=very light grey];
   columns obs Emp_ID First_Name Last_Name Hire_Date Job_Title Job_Class
      %do i=1 %to &reqcount;
         reqdate&i dt1doc&i dt2doc&i dtprint&i
         %end;;
   define obs / computed '#';
   define Emp_ID / display;
   define First_Name / display;
   define Last_Name / display;
   define Hire_Date / display;
   define Job_Title / display;
   %do i=1 %to &reqcount;
      define reqdate&i / display noprint;
      define dt1doc&i / display noprint;
      define dt2doc&i / display noprint;
      define dtprint&i / computed "%scan(&reqlist,&i,*)" format=$50.;
      %end;
   compute obs;
      obs_pvt+1;
      obs=obs_pvt;
      call define('_c1_','style','style=[backgroundcolor=black
foreground=white]');
      endcomp;
   %do i=1 %to &reqcount;
      compute dtprint&i / char length=50;
         if ^missing(_c%sysevalf(((&i-1)*4)+9)_) then
            dtprint&i=strip(put(_c%sysevalf(((&i-1)*4)+9)_,mmddyy10.)) ||
            ifc(missing(_c%sysevalf(((&i-1)*4)+10)_),' (no exp)',' (exp ' ||
            strip(put(_c%sysevalf(((&i-1)*4)+10)_,mmddyy10.)) || ')');
         * for requirements;
         if ^missing(_c%sysevalf(((&i-1)*4)+8)_) and _c%sysevalf(((&i-
                  1)*4)+8)_^='31dec2099'd then do;
            * for current requirements;
            if _c%sysevalf(((&i-1)*4)+8)_ <= date() then do;
                  * current achievement;
                  if ^missing(_c%sysevalf(((&i-1)*4)+9)_) and
                        (missing(_c%sysevalf(((&i-1)*4)+10)_)
                        or _c%sysevalf(((&i-1)*4)+10)_ >= date()) then
                        call define("_c%sysevalf(((&i-1)*4)+11)_",
                        'style','style=[backgroundcolor=&REQ_MET]');
                  * expired achievement;
                  else if ^missing(_c%sysevalf(((&i-1)*4)+9)_) and
                        ^missing(_c%sysevalf(((&i-1)*4)+10)_)
                        and _c%sysevalf(((&i-1)*4)+10)_ < date() then
                        call define("_c%sysevalf(((&i-1)*4)+11)_",
                        'style','style=[backgroundcolor=&REQ_EXP]');
                  * never achievement;
                  else if missing(_c%sysevalf(((&i-1)*4)+9)_) then
                        call define("_c%sysevalf(((&i-1)*4)+11)_",
                        'style','style=[backgroundcolor=&REQ_NEV]');
                  end;
            * for future requirements;
```

```
            else do;
                  * current achievement (early);
                  if ^missing(_c%sysevalf(((&i-1)*4)+9)_) and
                        (missing(_c%sysevalf(((&i-1)*4)+10)_)
                        or _c%sysevalf(((&i-1)*4)+10)_ < date()) then
                        call define("_c%sysevalf(((&i-1)*4)+11)_",
                        'style','style=[backgroundcolor=&REQFUT_MET]');
                  * grace period (never achievement or expired achievement);
                  else call define("_c%sysevalf(((&i-1)*4)+11)_",
                        'style','style=[backgroundcolor=&REQFUT_NO]');
                  end;
            end;
         * for preferred (desirements);
         else if _c%sysevalf(((&i-1)*4)+8)_='31dec2099'd then do;
            * current achievement;
            if ^missing(_c%sysevalf(((&i-1)*4)+9)_) and
                  (missing(_c%sysevalf(((&i-1)*4)+10)_)

                  or _c%sysevalf(((&i-1)*4)+10)_ >= date()) then
                  call define("_c%sysevalf(((&i-1)*4)+11)_",
                  'style','style=[backgroundcolor=&PREF_MET]');
            * expired achievement;
            else if ^missing(_c%sysevalf(((&i-1)*4)+9)_) and
                  ^missing(_c%sysevalf(((&i-1)*4)+10)_)
                  and _c%sysevalf(((&i-1)*4)+10)_ < date() then
                  call define("_c%sysevalf(((&i-1)*4)+11)_",
                  'style','style=[backgroundcolor=&PREF_EXP]');
            * never achievement;
            else if missing(_c%sysevalf(((&i-1)*4)+9)_) then
                  call define("_c%sysevalf(((&i-1)*4)+11)_",
                  'style','style=[backgroundcolor=&PREF_NEV]');
            end;
         * for irrelevant (neither required nor preferred);
         else if missing(_c%sysevalf(((&i-1)*4)+8)_) then
            call define("_c%sysevalf(((&i-1)*4)+11)_",
            'style','style=[backgroundcolor=&REQ_IRR]');
         endcomp;
      %end;
run;
ods html close;
%mend;
```

APPENDIX B. PARSEDIR MACRO

```
%macro parsedir(dir= /* logical location of folder to parse */,
   dsn= /* data set in LIB.DSN format into which to put directory info */,
   subdir=YES /* YES to parse subdirectories, NO to only parse the current DIR
*/,
   filetype=SAS EGP /* space-delimited list of file extensions to find or
<blank> for all file types */);
%let syscc=0;
%global parsedirRC;
%let parsedirRC=GLOBAL FAILURE;
%local subdirs i ext;
%if %upcase(&subdir)=YES %then %let subdirs=/s;
%else %let subdirs=;
filename filelist pipe "dir &subdirs ""&dir"""; * double quotes required in
case path has space;
data &dsn (keep=dir_name file_name crdate);
   length dir_name $200 file_name $200 crdate 8 inp $400;
   format crdate datetime17.;
   infile filelist truncover;
   input inp $400.;
   if length(inp)>9 and substr(strip(inp),1,9)='Directory' then
dir_name=strip(substr(strip(inp),13));
   else if lengthc(strip(inp))>0 and find(inp,'<DIR>')=0 and
not(find(inp,'Volume in drive')>0 and find(inp,'has no label')>0)
      and not(find(inp,'File(s)')>0 and find(inp,'bytes')>0) and
not(find(inp,'Dir(s)')>0 and find(inp,'bytes')>0) then do;
      if
      %if %length(&filetype)>=1 %then %do;
         %let i=1;
         %do %while(%length(%scan(&filetype,&i,,S))>1);
            %let ext=%scan(&filetype,&i,,S);
            %if &i>1 %then or;
            lowcase(scan(inp,-1,.))="%lowcase(&ext)"
            %let i=%eval(&i+1);
            %end;
         %end;
      %else %do;
         1=1
         %end;
      then do;
         crdate=input(substr(inp,1,20),mdyampm20.);
         file_name=strip(lowcase(substr(inp,39)));
         if lengthn(file_name)>0 then output;
         end;
      end;
   retain dir_name;
run;
%if &syscc=0 %then %let parsedirRC=;
%mend;
```