Technical Paper

SAS® SPD Engine and Hadoop Working Together

Requirements and Best Practices, Second Edition

Release Information Content Version: 3.0 March 2017.

Contents

Introduction

Using the SAS® Scalable Performance Data (SPD) Engine with SAS® applications, you can access data that is stored in the Hadoop Distributed File System (HDFS). Once you connect to the Hadoop cluster, you can write data, retrieve data for analysis, perform administrative functions, and update data.

This paper explains the benefits of using the SPD Engine and what you need to do in order to connect to a Hadoop cluster. In addition, this paper describes how specific SPD Engine functionality works with Hadoop and provides requirements and best practices.

Note: This paper has been updated to include the enhancements for the third maintenance release for SAS 9.4.

Why Use the SPD Engine for Hadoop Data Storage?

Using the SPD Engine and its proprietary file format provides these benefits:

- The SPD Engine is a scalable engine delivered to SAS customers as part of Base SAS®. You do not have to license additional software (besides Base SAS) to connect to a Hadoop cluster using the SPD Engine.
- The SPD Engine is designed for high-performance data delivery, reading data sets that contain billions of observations. The engine uses threads to read data very rapidly and in parallel.
- The SPD Engine organizes data into a streamlined file format that has advantages for a distributed file system like HDFS. Data is separate from the metadata, and the file format partitions the data.
- Most existing SAS applications can run with the SPD Engine with little modification other than to the LIBNAME statement.
- Using the SPD Engine, many SAS file features are supported, such as encryption, file compression, indexes, SAS passwords, special missing values, user-defined formats and informats, and physical ordering of returned observations.
- The SPD Engine provides several performance benefits, such as creating a partitioned index, parallel processing, and optimized WHERE processing. In addition, you can control the SPD Engine I/O block size and the data partition size to improve performance in your environment.
- The SPD Engine supports SAS Update operations for data stored in HDFS. For performance best practices, see "Updating Data in HDFS."
- You can use the SAS® High-Performance Analytics procedures on an SPD Engine data set stored in HDFS, taking advantage of the distributed processing capabilities of Hadoop. The procedures use the SAS® Embedded Process to submit a MapReduce program to the Hadoop cluster.

What Is the SPD Engine?

The SPD Engine is a SAS library engine that is delivered to SAS customers as part of Base SAS. The SPD Engine's computing scalability provides high-performance data delivery, accessing data sets that contain billions of observations. The SPD Engine is extended to read, write, and update data in HDFS.

The SPD Engine works like other SAS library engines. That is, you execute a LIBNAME statement to assign a libref and specify the engine. You use that libref throughout the SAS session where a libref is valid. Here is an example of a LIBNAME statement that connects to a Hadoop cluster:

```
libname myspde spde '/user/abcdef' hdfshost=default;
```

In the LIBNAME statement, you specify the pathname to a directory in a Hadoop cluster. In addition, you must include the HDFSHOST=DEFAULT argument, which specifies to connect to the specific Hadoop cluster that is defined in Hadoop cluster configuration files.

Most existing SAS applications run with the SPD Engine with little modification other than to the LIBNAME statement. For example, SAS file features such as encryption, file compression, member-level locking, indexes, SAS passwords, special missing values, user-defined formats and informats, and physical ordering of returned observations are supported.

The following SAS file features, however, are not supported:

- Audit trails
- Cross-Environment Data Access (CEDA)
- Extended attributes
- Generation data sets
- Integrity constraints
- Record-level locking
- SAS catalogs, SAS views, and MDDB files

In addition, SAS/CONNECT® and SAS/SHARE® software do not support SPD Engine data sets stored in HDFS.

Requirements

- To access data in HDFS using the SPD Engine, you must have the second maintenance release or later for SAS 9.4.

- You must make the specific Hadoop cluster configuration files available to the SPD Engine. This is explained in the topic "Configuring the SPD Engine to Connect to a Hadoop Cluster."

- Within a SAS session, you can submit multiple LIBNAME statements to different directories in the Hadoop cluster, but you can connect to only one Hadoop cluster at a time per SAS session. You must clear the LIBNAME statement connection to a Hadoop cluster before connecting to a different Hadoop cluster. In addition, JAR files are specific to each Hadoop distribution, as well as to the version of the Hadoop distribution. So, within a SAS session, you can connect to only those Hadoop clusters that use the same set of Hadoop distribution JAR files.

More Information

See the LIBNAME statement syntax and examples of how to use Hadoop data storage in the document, *SAS® 9.4 SPD Engine: Storing Data in the Hadoop Distributed File System.*

For complete instructions, including the list of the required Hadoop JAR files and Hadoop cluster configuration files, see "Configuring SPD Engine" in the *SAS® 9.4 Hadoop Configuration Guide for Base SAS and SAS/ACCESS*.

Understanding the SPD Engine File Format

The SPD Engine organizes data into a streamlined file format that has advantages for a distributed file system like HDFS.

- Data is separate from the metadata. The file format consists of separate file types: one for data, one for metadata, and two for indexes. Each type of file has an identifying file extension. The extensions are `.dpf` for data, `.mdf` for metadata, and `.hbx` and `.idx` for indexes.

- The SPD Engine file format partitions the data by spreading it across multiple files based on a partition size. Each partition is stored as a separate physical file with the extension `.dpf`. Depending on the amount of data and the partition size, the data can consist of one or more physical files, but is referenced as one logical file. The default partition size is 128 megabytes.

Data Directory Names

When the SPD Engine writes data to a Hadoop cluster directory (which is specified in the LIBNAME statement), the SPD Engine automatically creates a subdirectory with the specified data set name and the suffix `_spde`. For example, if you write a data set named BigFile to the Hadoop cluster directory `/user/abcdef/`, the SPD Engine data partition files are located at `/user/abcdef/bigfile_spde/`. The SPD Engine metadata and index files for that data set are located at `/user/abcdef/`.

The suffix `_spde` helps you identify directories that include SPD Engine data sets and avoid directory name collisions.

If you delete a data set, the generated SPD Engine subdirectory is removed along with the metadata and index files that are located in the Hadoop cluster directory.

Data Distribution

When writing data to a Hadoop cluster, the SPD Engine ensures that the data is distributed appropriately. The SPD Engine uses its partition size and the HDFS block size to compute the maximum number of observations that can fit into both parameters. That is, the observations never span multiple partitions or multiple blocks.

After the data set is written to a Hadoop cluster, the actual block size of the loaded data might be less than the block size that was defined by the Hadoop administrator. The reason for the size difference can be because of the SPD Engine calculations for the partition size, block size, and observation length.

Best Practices

You should not defragment the Hadoop cluster. Changing the block size and re-creating the files could result in the data becoming inaccessible by SAS.

Replacing a File

The default behavior for the SPD Engine is that a permanently stored data set is replaced with another data set of the same name. The REPLACE system option, which specifies whether permanently stored data sets can be replaced, is specified by default.

You can specify the NOREPLACE system option to prevent the accidental replacement of an existing data set. In addition, you can use the REPLACE= data set option to override whether a new data set can overwrite an existing data set of the same name.

More Information

For more information about the NOREPLACE system option, see *SAS® 9.4 System Options: Reference*. For more information about the REPLACE= data set option, see *SAS® 9.4 Data Set Options: Reference*.

I/O Operation Performance

To improve I/O operation performance, consider adjusting the SPD Engine I/O block size and the data partition size.

The I/O block size determines the amount of data that is physically transferred in an I/O operation. The larger the block size, the less I/O. The default block size is 1,048,576 bytes (1 megabyte). You can specify a different block size with the IOBLOCKSIZE= LIBNAME statement option and the IOBLOCKSIZE= data set option.

I/O Block Size Best Practices

Depending on the amount of data and the partition size, your data will consist of one or more physical data partition files. The larger the partition size, the fewer the partitions. The default partition size is 128 megabytes. You can specify a partition size with the PARTSIZE= LIBNAME statement option and the PARTSIZE= data set option.

Partition Size Best Practices

- Processing data using multiple threads is related to having multiple data partition files to process. It is recommended that you set the partition size so that the resulting number of partitions is a small multiple of the number of CPUs on the SAS client machine. For example, if the number of CPUs is 4, then the resulting number of partitions should be 8, 12, 16, or so on, depending on the amount of data.

- For most WHERE processing, the SPD Engine creates one or more threads per data partition file. To allow sufficient threading, set a partition size so that you have several partitions per CPU, but not so many that there are too many partitions (for example, hundreds of partitions) per CPU.

- To update data, a smaller partition size provides the best performance. When you submit an update, the SPD Engine locates the appropriate data partition file, modifies the value, and rewrites all replications of the partition. Because each update requires that the partition be rewritten, it is recommended that you perform updates only occasionally or set a small partition size if you are planning to update the data frequently.

More Information

See the IOBLOCKSIZE= options and the PARTSIZE= options in *SAS® 9.4 SPD Engine: Storing Data in the Hadoop Distributed File System*.

Updating Data in HDFS

HDFS does not support updating data. However, because traditional SAS processing involves updating data, the SPD Engine supports SAS Update operations for data stored in HDFS.

To update data in HDFS, the SPD Engine uses an approach that replaces the data set's data partition file for each observation that is updated. When an update is requested, the SPD Engine re-creates the data partition file in its entirety (including all replications), inserting the updated data. Because the data partition file is replaced for each observation that is updated, the greater the number of observations to be updated, the longer the process.

For a general-purpose data storage engine like the SPD Engine, the ability to perform small, infrequent updates can be beneficial. However, updating data in HDFS is intended for situations when the time it takes to complete the update outweighs the alternatives.

Best Practices

- It is recommended that you set up a test in your environment to measure Update operation performance. For example, update a small number of observations to gauge how long updates take in your environment. Then, project the test results to a larger number of observations to determine whether updating is realistic.

- It is recommended that you do not use the SQL procedure to update data in HDFS because of how PROC SQL opens, updates, and closes a file. There are other SAS methods that provide better performance such as the DATA step UPDATE statement and MODIFY statement.

- The performance of appending a data set can be slower if the data set has a unique index. Test case results show that appending a data set to another data set without a unique index was significantly faster than appending the same data sets with a unique index.

SPD Engine File System Locking

The HDFS concurrent access model allows multiple readers and a single writer. If an application accesses a file to write to it, no other application can write to the file, but multiple applications can read the file. The SPD Engine supports a file system locking strategy that honors the HDFS concurrent access model and provides additional levels of concurrent access to ensure the integrity of the data stored in HDFS.

By default, the SPD Engine creates a Write access lock file when a data set stored in HDFS is opened for Write access. With the Write access lock file, no other SAS session can write to the file, but multiple SAS sessions can read the file if the readers accessed the data set before the Write access lock file was created. In addition, a Write lock file protects a data set during a replacement operation.

During concurrent access, the following describes the results of the default SPD Engine locking mechanism:

- Once a SAS session opens a data set for Write access, any previous readers can continue to access the data set. However, the readers could experience unexpected data results. For example, the writer could delete the data set while the readers are accessing the data set.

- Once a SAS session opens a data set for Write access, any subsequent reader is not allowed access to the data set.

To increase the level of concurrent access protection, you can request that a lock file be created for Read access. To request a Read lock file, define the SAS environment variable SPDEREADLOCK and set it to YES. Then, when a SAS session opens a data set for Read access, a Read lock file is created. A Read lock file prevents concurrent Read and Write access to the data set. Multiple readers still have concurrent access.

To store both the Read and Write lock files, the SPD Engine creates a lock directory in the `/tmp` directory. The lock directory name includes the name of the data set, an eight-character hexadecimal value, and the suffix _spdslock9, such as `BigFile_0000393a_spdslock9`. In most situations, you do not see the lock directory because lock files are deleted when the application process completes.

You can specify a pathname for the SPD Engine lock directory by defining the SAS environment variable SPDELOCKPATH to specify a directory in the Hadoop cluster.

Requirements

When you request a Read lock file, all data access requires Write permission to the Hadoop cluster.

Best Practices

- In some situations, a SAS process can be terminated in such a way that the SPD Engine does not follow normal shut down procedures, which can result in a lock file not being deleted. The orphan lock file could prevent a subsequent open of the data set. If this occurs, the orphan lock file must be manually deleted by submitting HDFS commands. To delete the orphan lock file, you can use the HADOOP procedure to submit HDFS commands.

- When you request Read lock files, submitting PROC DATASETS could cause other SAS sessions to be locked out of the data sets in the library. With the SPD Engine, PROC DATASETS must open each data set in the library to retrieve file attributes, such as whether the data set is indexed. When a data set is open, it is locked for Read access and is unavailable to writers.

- It is recommended that instead of using PROC DATASETS, you should use a SAS procedure that has one focus such as PROC DELETE or PROC COPY (instead of the PROC DATASETS DELETE or COPY statements) unless you intend to execute multiple steps.

More Information

For more information about the SPDEREADLOCK and the SPDELOCKPATH environment variables, see *SAS® 9.4 SPD Engine: Storing Data in the Hadoop Distributed File System.* For more information about PROC HADOOP, see *Base SAS® 9.4 Procedures Guide*.

SPD Engine Distributed Locking

The SPD Engine supports distributed locking for data stored in HDFS. Distributed locking provides synchronization and group coordination services to clients over a network connection. For the service provider, the SPD Engine uses the Apache ZooKeeper coordination service (specifically the implementation of the recipe for Shared Lock that is provided by Apache Curator).

Distributed locking does not result in orphan lock files. ZooKeeper automatically deletes locks when a client disconnects from the service.

Requirements

- ZooKeeper 3.4.0 or later must be downloaded, installed, and running on the Hadoop cluster. The zookeeper JAR file is required.
- Curator 2.7.0 or later must be downloaded on the Hadoop cluster. The JAR files curator-client, curator-framework, and curator-recipes are required.
- The Hadoop distribution JAR files guava, log4j, and slf4j are required on the client side.
- The JAR files must be available to the SAS client machine. The SAS environment variable SAS_HADOOP_JAR_PATH must be defined and set to the location of the Hadoop distribution JAR files.
- To request distributed locking, you must first create an XML configuration file that contains information so that the SPD Engine can communicate with ZooKeeper. In addition, you must define the SAS environment variable SPDE_CONFIG_FILE to specify the location of the user-defined XML configuration file. The location must be available to the SAS client machine.

More Information

For more information about distributed locking, how to create the XML configuration file, and the SAS environment variable SPDE_CONFIG_FILE, see *SAS® 9.4 SPD Engine: Storing Data in the Hadoop Distributed File System*.

Parallel Processing

Parallel processing uses multiple threads that run in parallel so that a large operation is divided into multiple smaller ones that are executed simultaneously. Both Hadoop and the SPD Engine support parallel processing, which is described below:

- Hadoop processes large amounts of data across a cluster of connected machines by running different processes in parallel. HDFS facilitates parallel processing by storing data in a distributed manner.
- On the SAS client machine, the SPD Engine supports parallel processing to improve the performance of reading data stored in HDFS by running multiple threads in parallel.
- Together, Hadoop and the SPD Engine have a direct connection between each thread running on the SAS client machine and the corresponding node on the Hadoop cluster where the data resides.

By default, the SPD Engine performs parallel processing only if a Read operation includes WHERE processing. If the Read operation does not include WHERE processing, the Read operation is performed by a single thread. To request parallel processing for all Read operations (available in SAS 9.4) and for Write operations (available in the third maintenance release for SAS 9.4), use these options:

- SPDEPARALLELREAD= system option to request parallel Read processing for the SAS session.
- PARALLELREAD= LIBNAME statement option to request parallel Read processing when using the assigned libref.
- PARALLELREAD= data set option to request parallel Read processing for the specific data set.
- PARALLELWRITE= LIBNAME statement option to request parallel Write processing when using the assigned libref.
- PARALLELWRITE= data set option to request parallel Write processing for the specific data set.

Requirements

- When you request parallel processing for all Read operations, the SPD Engine generates a WHERE expression that qualifies all observations. You might see messages in the SAS log that refer to the generated WHERE expression.
- The SPD Engine can determine that parallel processing is not appropriate. For example, if the SAS code includes a BY statement or if the data has been sorted, parallel processing is not appropriate because parallel processing returns data in a random order. If the SPD Engine determines that parallel processing is not appropriate, the SPD Engine disables parallel processing and displays a message in the SAS log, such as:

  ```
  NOTE: Parallel WHERE evaluation suppressed due to sort order on table.
  ```

- In some situations, submitting SAS code that depends on an observation number (such as the OBS= system and data set options or the POINT= statement option) might produce an error. Because an observation number is calculated as the data is retrieved and not stored as a value, the observation number is meaningless when retrieving the data in parallel.
- When parallel processing occurs, the order in which the observations are returned might not be in the physical order of the observations in the data set. In the SAS log, you might see the following message:

  ```
  NOTE: Parallel read processing is in effect. Physical order is not preserved.
  ```

  Some applications require that observations be returned in the physical order. For example, the COMPARE procedure expects that observations are read from the data set in the same order in which they were written to the data set. Also, legacy SAS code that uses the DATA step or the OBS= data set option might rely on physical order to produce the expected results.

Best Practices

- For some environments, parallel processing might not improve the performance of reading data stored in HDFS. The availability of network bandwidth and the number of CPUs on the SAS client machine determine the performance. It is recommended that you set up a test in your environment to measure performance with and without parallel processing.

- To display information in the SAS log about parallel processing, set the MSGLEVEL= system option to I, such as `options msglevel=i;`. The SAS log reports whether parallel processing is in effect.

More Information

For more information about the SPDEPARALLELREAD= and PARALLELREAD= options, see *SAS® 9.4 SPD Engine: Storing Data in the Hadoop Distributed File System*.

WHERE Processing Optimization with MapReduce

To optimize the performance of WHERE processing, you can request that data subsetting be performed in the Hadoop cluster. Then, when you submit SAS code that includes a WHERE expression, the SPD Engine submits a Java class to the Hadoop cluster as a component in a MapReduce program. By requesting that data subsetting be performed in the Hadoop cluster, performance might be improved by taking advantage of the filtering and ordering capabilities of the MapReduce framework. As a result, only the subset of the data is returned to the SAS client.

Requirements

- To perform data subsetting in the Hadoop cluster, the following option must be specified and the following data set and SAS code requirements must be met:
    - The ACCELWHERE= LIBNAME statement option or data set option must be specified.
    - The data set cannot be encrypted or compressed.
    - The data set must be larger than the HDFS block size.
    - The submitted SAS code cannot request BY-group processing or include the STARTOBS= option or ENDOBS= option.
    - The LIBNAME statement cannot include the HDFSUSER= option.
    - The following WHERE expression syntax is supported :
        - comparison operators, such as EQ, NE, GT, LT, GE, and LE
        - IN operator
        - full bounded range condition, such as `where 500 = empnum <= 1000;`
        - BETWEEN-AND operator, such as `where empnum between 500 and 1000;`
        - compound expressions using the logical operators AND, OR, and NOT, such as `where skill = 'java' or years = 4;`
        - parentheses to control the order of evaluation, such as `where (product='GRAPH' or product='STAT') and country='Canada';`
    - The submitted WHERE expression cannot include any of the following syntax :
        - a variable as an operand, such as `where lastname;`
        - variable-to-variable comparison

    - SAS functions, such as SUBSTR, TODAY, UPCASE, and PUT
    - arithmetic operators *, /, +, -, and **
    - IS NULL or IS MISSING and IS NOT NULL or IS NOT MISSING operators
    - concatenation operator, such as || or !!
    - negative prefix operator, such as `where z = -(x + y);`
    - pattern-matching operators LIKE and CONTAINS
    - sounds-like operator SOUNDEX (=*)
    - truncated comparison operator using the colon modifier, such as `where lastname=:'S';`
- The Hadoop configuration file must include the properties to run MapReduce (MR1) or MapReduce 2 (MR2) and YARN.
- The JRE version for the Hadoop cluster must be either 1.6 (the default) or 1.7. If the JRE version is 1.7, you must use the ACCELJAVAVERSION= LIBNAME statement option to specify the version.
- On a Windows SAS client machine, to submit an MR2 or YARN program to a Hadoop cluster, you must first apply a patch that is available for the Hadoop distribution. Contact Cloudera or Hortonworks for the patch.

Best Practices

- Performance is often improved with large data sets when the WHERE expression qualifies only a relatively small subset.

- To display information in the SAS log about WHERE processing optimization, set the MSGLEVEL= system option to I, such as `options msglevel=i;`. The SAS log reports whether data filtering occurred in the Hadoop cluster. If optimization occurred, the Hadoop Job ID is displayed in the SAS log. If optimization did not occur, additional messages explain why.

More Information

For more information about the ACCELWHERE= options, see *SAS® 9.4 SPD Engine: Storing Data in the Hadoop Distributed File System.*

Using SAS High-Performance Analytics Procedures

You can use the SPD Engine with SAS High-Performance Analytics procedures to read and write the SPD Engine file format in HDFS. In many cases, the SPD Engine data used by SAS High-Performance Analytics can be read and written in parallel through the SAS Embedded Process.

Requirements for SAS Embedded Process Parallel Read

- Access to the machines in the cluster where a SAS High-Performance Analytics deployment of Hadoop is installed and running.
- The data set cannot be encrypted or compressed.
- The STARTOBS= and ENDOBS= data set options cannot be specified.

Requirements for SAS Embedded Process Parallel Write

- The ALIGN=, COMPRESS=, ENCRYPT=, and PADCOMPRESS= data set options cannot be specified.
- The SAS client machine must have a data representation that is compatible with the data representation of the Hadoop cluster. The SAS client machine must be either Linux x64 or Solaris x64.

Best Practices

- With SAS® Enterprise Miner™, a SAS process can be terminated in such a way that the SPD Engine does not follow normal shut down procedures, which can result in a lock file not being deleted. The orphan lock file could prevent a subsequent open of the data set. If this occurs, the orphan lock file must be manually deleted by submitting HDFS commands. To delete the orphan lock file, you can use the HADOOP procedure to submit HDFS commands.

- For SAS High-Performance Analytics Work files, the SPD Engine uses the standard UNIX temporary directory `/tmp`. To override the default Work directory, you can define the SAS environment variable SPDE_HADOOP_WORK_PATH and set a different directory. To set the location, the directory must exist, and you must have Write access. For example, the following OPTIONS statement sets the Work directory:

  ```
  options set=SPDE_HADOOP_WORK_PATH="/sasdata/cluster1/hpawork";
  ```

More Information

For more information about the SPDE_HADOOP_WORK_PATH environment variable, see the *SAS® 9.4 SPD Engine: Storing Data in the Hadoop Distributed File System.*

Kerberos Authentication

The SPD Engine can be configured for Kerberos ticket cache-based logon authentication using MIT Kerberos 5 Version 1.9.

If the Hadoop cluster supports Kerberos, the SPD Engine honors Kerberos authentication and authorization as long as the Hadoop cluster configuration files are accessible.

Best Practices

If the Hadoop cluster supports Kerberos, do not use the HDFSUSER= option in the LIBNAME statement to specify a different user ID than your current logon ID. If you do so, Kerberos authentication is bypassed, which prevents access to the Hadoop cluster.

Accessing SPD Engine Data Using Hive

SAS provides a custom Hive SerDe for SPD Engine data that is stored in HDFS. The SerDe makes the data available for applications outside of SAS to query using HiveQL.

Requirements

- You must deploy SAS® Foundation using the SAS® Deployment Wizard. Select **SAS Hive SerDe for SPDE Data**.
- You must be running a supported Hadoop distribution that includes Hive 0.13 (for example, Cloudera CDH 5.2, Hortonworks HDP 2.1 or later, or MapR 4.0.2 or later).
- The data stored in HDFS must have been created using the SPD Engine.
- The SerDe is delivered as two JAR files that must be deployed to all nodes in the Hadoop cluster.
- The data set must be registered in the Hive metastore. SAS supplies a registration utility to the data set. You cannot use any other method to register data sets.

More Information

For more information about the Hive SerDe for the SPD Engine, see *SAS® 9.4 SPD Engine: Storing Data in the Hadoop Distributed File System*.

References

SAS® 9.4 SPD Engine: Storing Data in the Hadoop Distributed File System, Fourth Edition.
http://support.sas.com/documentation/cdl/en/engspdehdfsug/69725/PDF/default/engspdehdfsug.pdf

SAS® 9.4 Hadoop Configuration Guide for Base SAS and SAS/ACCESS, Fourth Edition.
http://support.sas.com/resources/thirdpartysupport/v94/hadoop/hadoop-configuration-guide-base-access-4th.pdf

SAS® 9.4 System Options: Reference, Fifth Edition.
http://support.sas.com/documentation/cdl/en/lesysoptsref/69799/PDF/default/lesysoptsref.pdf

SAS® 9.4 Data Set Options: Reference, Fourth Edition.
http://support.sas.com/documentation/cdl/en/ledsoptsref/69751/PDF/default/ledsoptsref.pdf

Base SAS® 9.4 Procedures Guide, Sixth Edition.
http://support.sas.com/documentation/cdl/en/proc/69850/PDF/default/proc.pdf

Contact Information

Contact the developers at:

Lisa Brown
SAS Institute Inc.
600 Research Drive
Cary, NC 27513
Work Phone: (919) 531-7749
Lisa.Brown@sas.com

Scott Vance
SAS Institute Inc.
11920 Wilson Parke Avenue
Austin, TX 78726
Work Phone: (512) 840-6303
Scott.Vance@sas.com

Jim Craig
SAS Institute Inc.
600 Research Drive
Cary, NC 27513
Work Phone: (919) 531-5330
Jim.Craig@sas.com

To contact your local SAS office, please visit: sas.com/offices