SAS® Infrastructure for Risk Management 3.4: Programmer's Guide

SAS® Documentation
October 2, 2017

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2017. *SAS® Infrastructure for Risk Management 3.4: Programmer's Guide*. Cary, NC: SAS Institute Inc.

SAS® Infrastructure for Risk Management 3.4: Programmer's Guide

Copyright © 2017, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

October 2017

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.

3.4-P1:irmscug

Contents

Chapter 1 • About Parallel Programming for SAS Infrastructure for Risk Management **1**
About Creating Parallel Program's Using SAS Infrastructure for Risk Management . . . 1
About This Book 1
Recommended Reading 2

Chapter 2 • About SAS Infrastructure for Risk Management **3**
SAS Infrastructure for Risk Management Architecture 3
SAS Infrastructure for Risk Management Data Flow 5
SAS Infrastructure for Risk Management Distributed Development 5
What is a SAS Infrastructure for Risk Management Federated Area? 6
Federated Content Development 12

Chapter 3 • About the Scripting Client **17**
What is the Scripting Client? 17
SetUp Environment for Using the Scripting Client 18

Chapter 4 • Creating a Basic Parallel Program **21**
A Basic Parallel Program 21
Identify the Tasks That You Need in Your Parallel Program 22
Write the Script That Creates the Hello World Parallel Program 23
View the Results 25

Chapter 5 • Using Additional Parallel Programming Features **27**
About Scripting Client Parallel Programming Features 27
Create Tasks 27
Generate Doxygen Documentation 30
Use Parameter Substitution 30
Create a Job Flow 30
Create and Add SubFlows 30
Use Partitions 33
Use Generic and Temporary Libraries 41
Create and Add a New Federated Area 42

Appendix 1 • Scripting Client: A Tutorial **45**
SAS Scripting Client: A Tutorial 45
About This Tutorial 45
Getting Started – Hello World 46
Set Up the Environment for Using SAS Scripting Client 46
View the Job Tasks 47
Write the SAS Scripting Client Program 48
View and Consume Results 50
Wrap-Up 52

Chapter 1

About Parallel Programming for SAS Infrastructure for Risk Management

About Creating Parallel Program's Using SAS Infrastructure for Risk Management 1

About This Book 1

Recommended Reading 2

About Creating Parallel Program's Using SAS Infrastructure for Risk Management

The SAS Infrastructure for Risk Management 3.4 platform is a high-performance job execution engine that leverages the *many-task computing* model. This is a method of parallel computing that emphasizes the use of many computing resources to accomplish many computational tasks. SAS programmers can use the SAS Infrastructure for Risk Management platform to implement parallel computing.

To simplify the process of creating parallel programs, SAS Infrastructure for Risk Management 3.4 provides a *scripting client*. The scripting client is a set of macros that simplify the complexity of creating parallel programs. Therefore, even if you are not a SAS programmer who is experienced in traditional parallel programming methods, such as multithreading and Message Passing Interface (MPI), you can use the scripting client to create and implement parallel programs. Using the SAS Infrastructure for Risk Management platform, you can scale your parallel programs from multicore modern CPU to grid computing.

About This Book

SAS Infrastructure for Risk Management: Programmer's Guide is intended to help SAS programmers understand how to easily create parallel programs that run on the SAS Infrastructure for Risk Management platform using the SAS Infrastructure for Risk Management 3.4 scripting client.

Recommended Reading

- *SAS Infrastructure for Risk Management 3.4: Administrator's Guide*
- *SAS Infrastructure for Risk Management 3.4: User's Guide*
- *SAS Studio 3.7: User's Guide*
- The Little SAS Book: A Primer (Buy)
- Learning SAS by Example: A Programmer's Guide (Buy)

Chapter 2

About SAS Infrastructure for Risk Management

SAS Infrastructure for Risk Management Architecture 3

SAS Infrastructure for Risk Management Data Flow 5

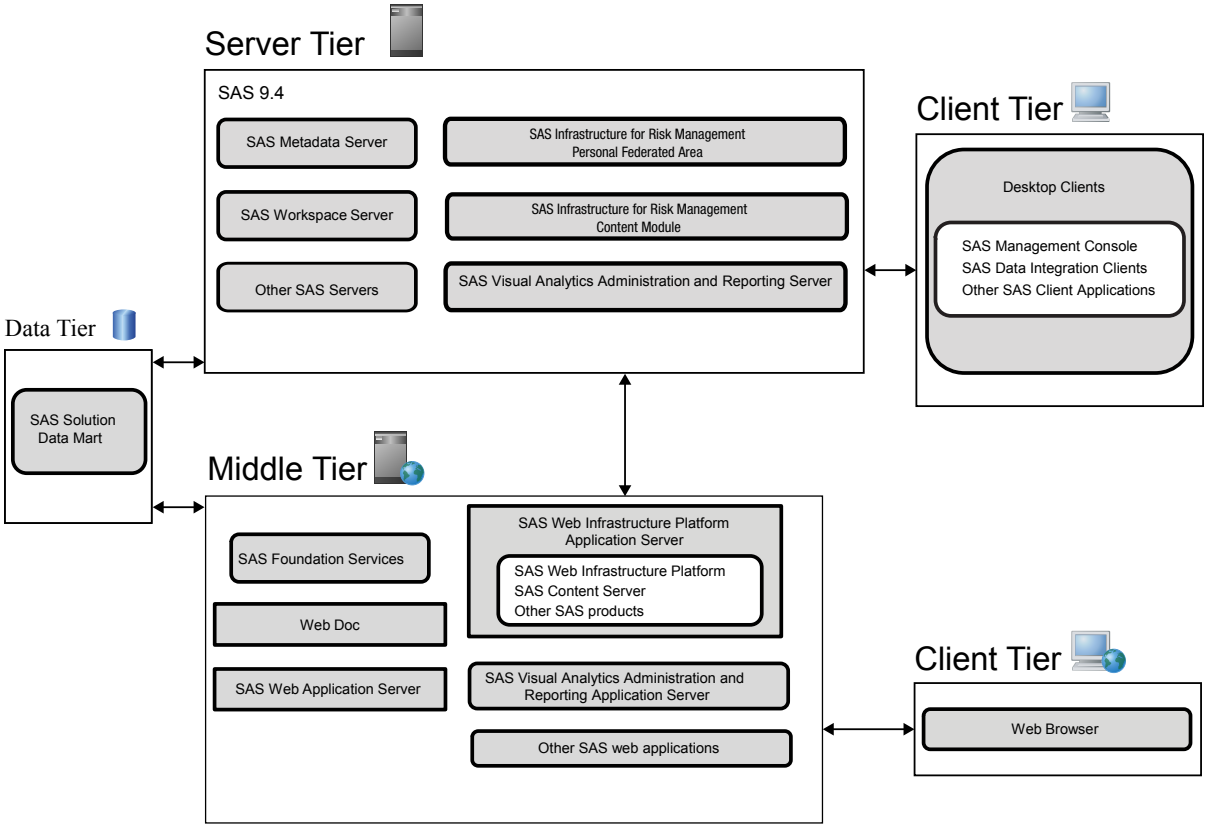
SAS Infrastructure for Risk Management Distributed Development 5
Overview 5
Contributors 6

What is a SAS Infrastructure for Risk Management Federated Area? 6
About SAS Infrastructure for Risk Management Federated Areas 6
Your Personal Federated Area 8
Folders in a Federated Area 9

Federated Content Development 12
What is Federated Content? 12
How Federated Content Is Processed 12
Job Flows 12
Tasks 13
Federated Input Tables 14
How Federated Input Tables Are Processed 15
SAS Infrastructure for Risk Management Scripting Client 16
Job Flow Script 16

SAS Infrastructure for Risk Management Architecture

SAS Infrastructure for Risk Management operates in a three-tiered environment, as shown in the following figure:



Server Tier

- handles requests from the client tier and the middle tier
- serves as an abstract layer between the data tier and the middle tier or between the data tier and the client tier
- consists of SAS applications, such as the SAS Metadata Server and SAS Application Server

Middle Tier

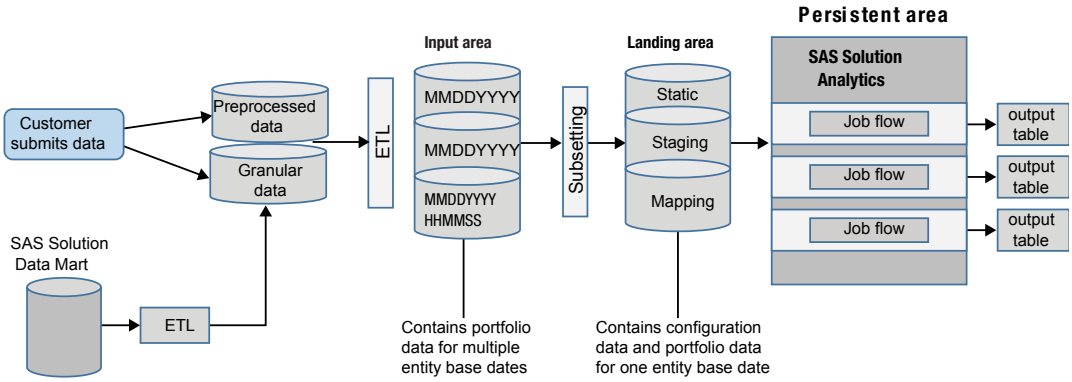
- receives and processes web requests from the client tier and passes these requests to the server tier and the data tier
- contains a web application server in addition to web applications such as the SAS Infrastructure for Risk Management web application

Client Tier

- initiates requests (via desktop client applications or web browsers) to perform the necessary work and to view formatted output
- contains theSAS Infrastructure for Risk Management web application from systems that are part of the client tier
- contains the GUI, which is developed in HTML5

SAS Infrastructure for Risk Management Data Flow

The following figure shows the flow of data in SAS Infrastructure for Risk Management solutions.



Here is the basic SAS Infrastructure for Risk Management data flow process:

1. Data is supplied to a solution in one of three ways:
   - The customer submits data directly to the input area of a federated area.
   - Data can be loaded into the input area when the system is running.
   - If the SAS Detail Data Store is in place, data can be drawn from the SAS Detail Data Store into the input area of a federated area.
2. Subsets of the input data are created in separate folders, for each reporting period. These subsets of input data are created in the Read-Only *staging* or Read-Only *landing area* of the SAS Infrastructure for Risk Management solution. The tables are versioned by date (8-character string – **`mmddyyyy`**) or date and time (14-character string – **`mmddyyyyhhmmss`**).
3. The output of the job flow is placed in the *persistent area*. The persistent area is a Read/Write area for the solution.

SAS Infrastructure for Risk Management Distributed Development

Overview

SAS Infrastructure for Risk Management solutions are designed to support *distributed development*. Distributed development means that developers in different locations can independently develop code that runs on the SAS Infrastructure for Risk Management platform.

Distributed development has the following implications:

- Code that is developed in one location must not break code that is developed in another location.
- Subsequent releases of a SAS Infrastructure for Risk Management solution must support all changes or fixes that are deployed since the prior release, including additions to flows, code, and data.
- Developers are responsible for the integrity of their code.
- If you modify a subflow that is used by other flows, you can break those flows. For example, you might break the flows if you changed the number or nature of the outputs of a subflow. Therefore, coordination of development groups is also necessary to ensure the integrity of the code that is being developed.
- With the exception of loading data, every installed federated area is read-only. In particular, once installed, a federated area must never be removed.
- You can modify your personal federated area as described in this programmer's guide.

Contributors

Contributors to the distributed development of SAS Infrastructure for Risk Management solutions include the following:

- SAS Research & Development

  SAS Research & Development provides the content that is included with your SAS Infrastructure for Risk Management solution.
- SAS Consultants

  SAS Consultants provide custom content that can be included in a future release of all SAS Infrastructure for Risk Management solutions.
- Consulting firms

  Consulting firms develop a custom product on top of SAS Infrastructure for Risk Management solutions.

What is a SAS Infrastructure for Risk Management Federated Area?

About SAS Infrastructure for Risk Management Federated Areas

The main goal of federated areas is to deploy content independently of the SAS Infrastructure for Risk Management platform. A secondary goal of a federated area is reproducibility; meaning, the ability to run existing content forever. In other words, deploying new content or a new platform must not break existing content.

The content that runs on the SAS Infrastructure for Risk Management platform shares the same architecture and layout. The difference between old and new content is the calculation content that is stored in the *federated area* of the federated content.

Content is any file that can be delivered in SAS Infrastructure for Risk Management, but that is not necessarily part of the SAS Infrastructure for Risk Management installation. The only way for software developers to deploy content for SAS Infrastructure for Risk

Management is to deploy a federated area. Therefore, federated areas and content are synonymous in SAS Infrastructure for Risk Management.

The federated area consists of a set of folders that has a specific structure. As a SAS Infrastructure for Risk Management developer, you must organize your content in a federated area by conforming to this specific structure.

A federated area contains the following elements:

- flows – files that describe the job flow
- code – string message data, tasks (including Java tasks), or macros
- inputs – SAS data sets, CSV files, Microsoft Excel templates, or XBRL templates
- documentation and tooltips files – information that is presented to the end user through the user interface

CAUTION:
Federated areas must not be altered once they are deployed.

Note: Federated areas must be designed to work with other federated areas.

It is important to understand the distinction between federated and non-federated content in order to maintain future releases or to develop content.

Only the flows, tasks, and inputs are *federated content*, which means that this content is shared among multiple federated areas. All the other content is local to a federated area and cannot directly be shared between federated areas. However, SAS Infrastructure for Risk Management must know about the other content areas in order to deliver all functionality.

When you install SAS Infrastructure for Risk Management, the following federated areas are also installed:

- fa.0.3.4 — contains only those elements that are required to make the platform run. There is no content in the platform federated area.
- fa.sample.3.4 — contains sample content to use for testing the SAS Infrastructure for Risk Management installation and to use as a reference.
- fa.*user_name* — is an optional *personal federated area*. The personal federated area is created on demand. In a personal federated area, developers can write content using parallel programs called job flows.

CAUTION:
Do not modify or delete a federated area Installed federated areas should not be changed. You cannot delete or modify the content of a federated area on disk, and you cannot delete or modify the federated area identifier that is metadata. If you modify or delete a federated area, database corruption and loss of data might occur.

However, there are two exceptions:

- You can upload data to the input area or the landing area of a federated area other than the platform federated area.
- If you are a content developer, you can modify your personal federated area (typically using the scripting client). The system manages the integrity of the job flow instances that reference your personal federated area. You cannot delete your personal federated area. However, you can delete the contents of your personal federated area.

CAUTION:
Tasks, flows, and data is shared across all federated areas. Therefore, do not change the definition of tasks, flows, and data. Doing so can cause

unpredictable results. In particular, do not change the inputs and outputs of tasks or flows. You can change the implementation of a task in a higher federated area, provided that the task's inputs and outputs remain the same. In addition, you can replace one task by two other tasks in a flow definition, provided that the inputs and outputs of the flow are preserved.

Note: A federated area can contain more folders or fewer folders than appear in the following examples.

Your Personal Federated Area

Your personal federated area is where you create tasks to use in your parallel programs and where you write job flow scripts that when executed create your parallel programs.

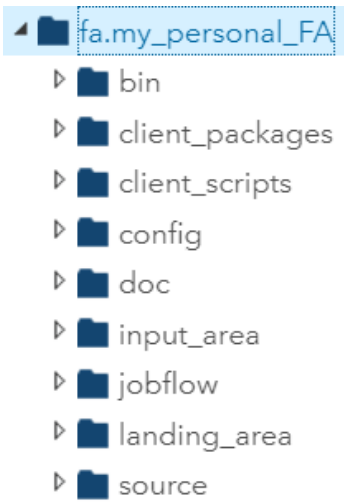
Your personal federated area is not seen or available to any other programmer's personal federated area. You cannot see or access other programmers' federated areas. In addition, your personal federated area is at the top of the precedence change. It is always the first federated area searched by SAS Infrastructure for Risk Management for federated content.

To access your personal federated area, complete the following steps:

1. Generate your personal federated area by logging on to SAS Infrastructure for Risk Management or SAS Studio with your account and executing the %irm_sc_init() macro.
2. In UNIX, create a symbolic link to your personal federated area on the SAS Infrastructure for Risk Management server. In Windows, create a folder shortcut to your personal federated area. This enables you to access your personal federated area from SAS Studio.

   To create a connection, you must have Read and Write permissions to your personal federated area.
3. Using a valid user account, log on to SAS Studio.
4. In the navigation pane, select **Server Files and Folders** ⇨ **Folder Shortcuts** ⇨ **SASIRM** ⇨ **pa** ⇨ **fas** ⇨ **fa.*user_name***, where *user_name* is the name of your personal federated area.

Here is the basic folder structure of your personal federated area:



Before you begin creating parallel programs, ensure that you understand the contents of the following folders located in your personal federated area:

- **client_packages** — contain metadata about all the tasks that are available on the SAS Infrastructure for Risk Management server for you to use with the scripting client.
- **client_scripts** — contain the job flow scripts that you have created.
- **input_area** — is the area in which you load data. For information about loading data, see *SAS Infrastructure for Risk Management 3.4: Administrator's Guide*.
- **source** — contains the individual tasks (programs) that you create to use in your parallel programs.

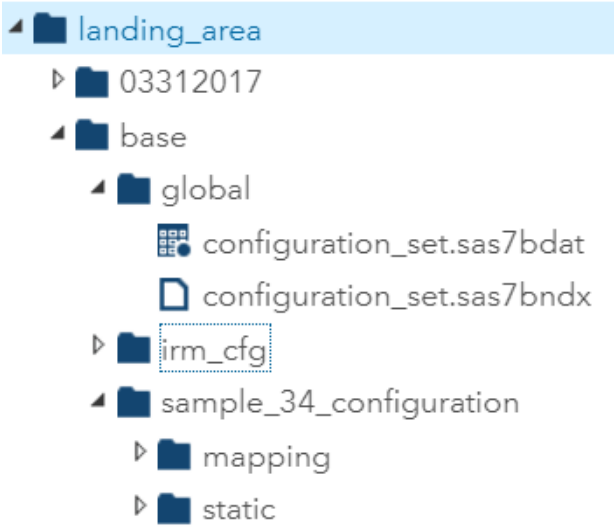
Folders in a Federated Area

The following sections describe the folders that are located in a SAS Infrastructure for Risk Management federated area.

Landing_Area Folder

Each federated area has its own *landing area*. The landing area is the read-only data mart of a SAS Infrastructure for Risk Management federated area. It contains the data objects (for example, SAS data sets) that are required for the flows that are defined in that federated area.

Here is the basic folder structure of landing_area in a federated area:



The landing area contains the following:

- The base date folders (named **`mmddyyyy`**) or date time folders (named **`mmddyyyyhhmmss`**) for which calculations are performed.

  Theses folders contain the data sets that pertain to the specific base date.
- The base folder, which contains configuration sets. Each configuration set contains the following folders:
  - mapping

Contains *mapping tables* that are designed as stand-alone tables. They are not joined to other tables. Mapping tables map one or more variables to each other. The mappings are used for transforming some raw data into the forms that are expected by the application. This transformation is part of the *data enrichment process*. The location of these tables is mapped by using the libnames.txt file in the config folder.

- static

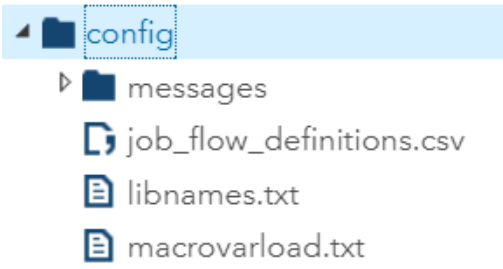
  Contains *input tables* that make up the configuration data model. These tables contain a historical repository of risk configurations for SAS Infrastructure for Risk Management. The location of these tables is mapped by using the libnames.txt file in the config folder.

Note: The landing area of a federated area might contain additional or fewer folders, depending on the context of the flows.

Config Folder

Developers use the files in the config folder to configure the behavior of job flows.

Here is the structure of the config folder:



The config folder contains the following files:

- job_flow_definitions.csv

  Lists the available job flow definitions. The first column is the category in which the job flow definition resides. The second column is the identifier of the job flow definition. The third column indicates whether the job flow can be run as solo, group, or both. The fourth column is a pipe (|) delimited list of the configuration sets for which the job flow is visible in the Create Instance window of the user interface.

- libnames.txt

  Maps the static input tables that are used by SAS Infrastructure for Risk Management. This file maps a logical name (using the LIBNAME statement) to the location of the directory that contains the static input tables.

  Note: With SAS Infrastructure for Risk Management 3.4, you can also define generic libraries in the libnames.txt file that can be used as input.

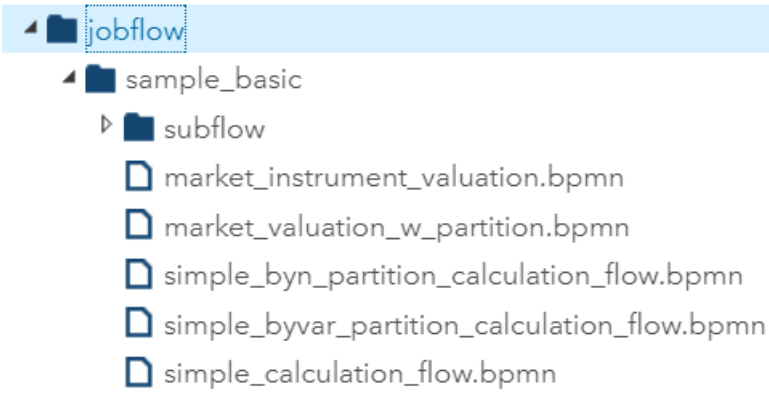
- macrovarload.txt

  Lists SAS data sets that define global macro variables that must be loaded before a task executes.

  Note: In order for the macro variables to be available, tasks must include the macro variables data sets as inputs.

Job Flow Folder

Here is the structure of the job flow folder:



The job flow folder contains *job flow definitions* or subdirectories that contain job flow definitions. A job flow definition is the program file that connects one or more tasks that need to be executed to complete a job.

Job flows are categorized as *definitions* and *instances*. A job flow has only one definition, but it can have many instances. One user can have multiple instances of the same job flow. In addition, many users can have multiple instances of the same job flow. An example of two instances of the same job flow is the same calculation that is performed using data from different base dates.

Subdirectories within the job flow folder are displayed as **Categories** in the SAS Infrastructure for Risk Management web application user interface. The categories are defined in the job_flow_definitions.csv file.

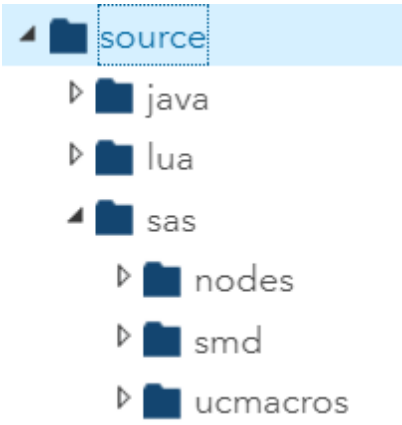
Job flows consist of one of the following elements:

- the tasks required to complete a job
- subflows
- input and output data files that are associated with the tasks

Source Folder

The source folder contains code that delivers the content functionality.

Here is the structure of the source folder:



The following folders are included in the source folder:

- doc — contains the solution-level federated content documentation files.
- java
  - bin — binary files (Java code can be delivered as Java files or compiled class files)
  - lib — JAR files
  - nodes — Java code that is directly invoked by flows
- lua
  - luarisk — Lua risk libraries
  - luastl — Lua collection, utility, and graph libraries
  - plugin — module that encodes and decodes JSON data
  - sas — Lua code for various functions and operations
- sas
  - nodes — SAS code that is directly invoked by flows
  - smd — string message data
  - umacros (compiled or uncompiled)

  Note: Depending on the content, additional folders might appear in the sas folder.

Note: The umacros folder might contain subfolders.

Federated Content Development

What is Federated Content?

- *Federated content* is contained in *federated areas*.
- Federated content is the mechanism by which developers add custom content to SAS Infrastructure for Risk Management.

How Federated Content Is Processed

- Only job flow files (.bpmn files), tasks (.sas files), and input tables are federated content that is shared across multiple federated areas. All other content is specific to the federated area in which it is located and not shared. For example, a task in federated area 1 cannot call a macro in federated area 2.
- SAS Infrastructure for Risk Management searches for federated content in federated areas from the highest to lowest precedence (by the federated ID assigned in metadata and in lexical order), until it finds the content.

Job Flows

- Job flow definitions are shared across federated areas.

- When searching for a job flow definition, SAS Infrastructure for Risk Management searches from the highest precedence federated area to the lowest precedence federated area. For example, if federated area 2 contains a file named flow1 and federated area 1 also contains a file named flow1, the file in federated area 2 is used to create a new instance of a job flow.
- After an instance is created, the instance does not change its definition. For example, if a flow1 file is added later to a higher precedence federated area the existing instances of previously created flows using this definition are not affected. However, new instances will use the new definition.

Tasks

Tasks are SAS programs. They are the basic building blocks of SAS Infrastructure for Risk Management.

- Tasks are analytical or reporting-oriented elements of processing that have a defined input or output, or both. All tasks should declare their inputs and outputs in their comment header using Doxygen syntax. For more information, see "Doxygen Syntax ".
- You can combine individual tasks as needed in a program with the objective of carrying out a job.
- When you use multiple tasks in your parallel program, SAS Infrastructure for Risk Management automatically determines the task dependencies and the order and parallelism in which the tasks are executed based on the availability of the input data sets and the number of cores that are available for processing.
- SAS tasks are located in the **`/source/sas/nodes`** folder of a federated area.
- The filenames of SAS tasks should be in lowercase and must not exceed 32 characters in length (excluding the suffix).
- Like job flows, tasks that are identified within a flow are searched for in federated areas from highest to lowest precedence.
- Tasks with the same name are assumed to be the same content. Therefore, a task named task1.sas accepts the same input tables and produces the same output tables as other tasks with the same name, regardless of their federated location.
- Like job flow definition files, changing or adding a new version of a task does not affect existing job flow instances. However, new executions of an instance will use definition in the highest precedence federated area.
- During execution of a task, the context of that execution environment is isolated to the federated area in which it resides. Therefore, any macros or Lua code that are called by the task must exist in the same federated area of the task.
- Tasks can have inputs and outputs that are partitioned. Partitioned tasks enable large amounts of data to be partitioned into smaller units of data and calculated across multiple cores. The task recombines the results of the partitioned data.

  For detailed information about partitioned tasks, see the documentation that is included in the generic sample federated area (fa.sample.3.4). The sample federated area contains sample flows that demonstrate the capabilities and functionality of SAS Infrastructure for Risk Management.

CAUTION:

All tasks must declare all of their inputs and outputs using the libnames.txt file. All inputs are immutable. All LIBNAME assignments must come from SAS

Infrastructure for Risk Management. Failure to follow the rules listed above will result in catastrophic failures. SAS Infrastructure for Risk Management is dependent on these programming rules. Not following them will render SAS Infrastructure for Risk Management inoperable.

Federated Input Tables

- Input tables are shared across multiple federated areas.
- All static input tables that are used by SAS Infrastructure for Risk Management tasks are mapped using the libnames.txt file. The libnames.txt file is located in the config folder of the federated area.

  All static input tables reside in the landing_area folder. Mappings are relative to the landing area. The file maps a logical name (libref) to a folder.

  For example, **`GLOBAL=%la/base/global`** specifies the folder **`base/global`** within the federated area in the landing_area folder. The libref *GLOBAL* should refer to that path.

  CAUTION:
  Directly accessing SAS data sets in librefs that are not mapped via libnames.txt is not permissible. Doing so could result in unpredictable results since SAS Infrastructure for Risk Management cannot guarantee that the task was executed at the right time. Basically, having undeclared LIBNAME statements in your SAS code violates the basic rule that all tasks must define all of their inputs and outputs.

  CAUTION:
  Do not define mappings to WebDAV folders that are the output of job flow instances. Doing so could result in invalid flows with cycles, erroneous results, and run-time execution errors. The correct way to use the output of one flow in another flow is to create a super-flow that contains both flows. However, you can define a generic library mapping to a WebDAV folder that is the output of a flow once the flow is published because a published flow cannot be modified or executed.

- Tasks can reference tables using one-, two-, or three-level names. Here are examples of table names:
  - GLOBAL
  - GLOBAL.myglobal
  - GLOBAL.myglobal.sas7bdat

  Note: The latter two examples are processed identically. In the second example, the *sas7bdat* suffix is assumed, by default. One-level names are processed somewhat differently than two- and three-level names.
- SAS Infrastructure for Risk Management 3.4 supports generic library definitions in the libnames.txt file of a federated area. Use of generic library definitions simplifies access to third-party data. For information about defining generic libraries, see *SAS Infrastructure for Risk Management 3.4: Administrator's Guide*.
- Input and output tables are defined by input and output Doxygen parameters in Doxygen. In each task source file, you must document the input and output table, table substitution, or collection. For more information, see "Doxygen Syntax "..

How Federated Input Tables Are Processed

This section explains how federated input tables are processed by SAS Infrastructure for Risk Management.

Assume that the following three federated areas exist:

- com.sas.solutions.risk.irm.fa.0.3.4 — ***/sas-configuration-directory/*****Levn/AppData/SASIRM/fa.0.3.4**
- com.sas.solutions.risk.irm.fa.2 — ***/sas-configuration-directory*****/Levn/AppData/SASIRM/fa2**
- com.sas.solutions.risk.irm.fa.2.5 — ***/sas-configuration-directory*****/Levn/AppData/SASIRM/fa2.5**

If a one-level name is specified, then SAS Infrastructure for Risk Management searches each libnames.txt file for the mapping in question in the federated area from the highest to lowest precedence.

For example, if the table references GLOBAL, then SAS Infrastructure for Risk Management searches the libnames.txt file in federated area 2.5. (Federated area 2.5 has the highest precedence because 2.5 is greater than 2.)

SAS Infrastructure for Risk Management is looking for a mapping for GLOBAL. If it finds a mapping, it adds the path to the concatenated LIBNAME statement that is used to define GLOBAL. This path is the first path in the libref path. If the mapping is not found, the search continues through the federated areas for a libnames.txt file that contains a mapping for GLOBAL. If no mapping is found, the task fails with an error.

Processing two- or three-level names is similar, except that SAS Infrastructure for Risk Management has the information that is required to verify that the actual table exists. As before, SAS Infrastructure for Risk Management searches for a mapping in the libnames.txt file. If it does not find a mapping, it searches the next federated area (by precedence). If SAS Infrastructure for Risk Management finds a mapping, it verifies that the file actually exists in the folder that is specified in the mapping.

Mapping enables content developers to overwrite a single table without having to override all tables using the same mapping (LIBNAME).

If SAS Infrastructure for Risk Management cannot locate the table, the task is not created and the SAS Infrastructure for Risk Management New Instance wizard reports an error that the instance cannot be created.

Consider the case of a pair of two-level names, GLOBAL.table1 and GLOBAL.table2, that use the same mapping that was previously described. Both tables reside in federated area 1, but only GLOBAL.table1 resides in federated area 2. The following LIBNAME statement is generated:

```
LIBNAME GLOBAL ('/sas-configuration-directory/Levn/AppData/SASIRM/fa2
/landing_area/base/global' ''/sas-configuration-directory/Levn/AppData
/SASIRM/fa1/landing_area/base/global);
```

According to the LIBNAME statement, the tables are located as follows:

- table1.sas7bdat is found in federated area 2 (***sas-configuration-directory*****/Levn/AppData/SASIRM/fa2/landing_area/base/global**)
- table2.sas7bdat is found in federated area 1 (***sas-configuration-directory*****/Levn/AppData/SASIRM/fa1/landing_area/base/global**)

The search for mappings uses the following case order:

1. as specified in the flow definition (for example, “GloBal”, if so specified in the flow definition)
2. all uppercase (for example, “GLOBAL”)
3. all lowercase (for example, “global”)
4. initial capitalization (for example, “Global”)

Note: SAS recommends that you use three-level names in your job flow definitions and uppercase mapping in your libnames.txt files.

SAS Infrastructure for Risk Management Scripting Client

To simplify the process of content development, SAS Infrastructure for Risk Management 3.4 provides a *scripting client*. The scripting client is a set of macros that simplifies the creation of parallel programs. Therefore, even if you are not an experienced SAS programmer, you can use the scripting client to create and implement parallel programs that you execute on the SAS Infrastructure for Risk Management platform. This guide explains how to create content using the SAS Infrastructure for Risk Management scripting client.

Job Flow Script

A *job flow script* creates your parallel program when you execute it. In a job flow script, you combine the scripting client macros and tasks in your parallel program that are needed to complete the job. You create and execute a job flow script in SAS Studio.

When you execute the job flow script, a job flow definition file is created and saved in the **job flow** folder of your personal federated area. In addition, an entry for the job flow is added to the job_flow_definitions.csv file in the config folder.

Chapter 3

About the Scripting Client

What is the Scripting Client? 17
Scripting Client Overview 17
Primary Scripting Client Macros 17

SetUp Environment for Using the Scripting Client 18

What is the Scripting Client?

Scripting Client Overview

The scripting client consists of a set of SAS macros. These macros simplify the process of creating parallel programs in the required format so that you to execute them on the SAS Infrastructure for Risk Management platform.

Using the scripting client, you can perform the following tasks:

1. Create a *job flow script*, that when executed creates a parallel program. In the job flow script, you specify tasks and their associated input and output data sets that are required in your parallel program to complete a job.
2. Generate and save a parallel program as a *job flow definition*. The job flow definition for your program contains tasks that are needed in your program. After you have created and saved a job flow definition, you can create multiple *instances* of that definition.
3. Create new tasks or modify existing tasks that you can use in new job flow definitions.

Primary Scripting Client Macros

The SAS Infrastructure for Risk Management provides a set of macros. These macros have the same names as the available tasks and are defined with a set of common parameters. All scripting client macros are located in an SAS Infrastructure for Risk Management autocall library and begin with the prefix **`%irm_sc`**.

Here are the primary scripting client macros that you use to create a job flow script:

`%irm_sc_init`
starts the scripting client.

%irm_sc_build_jobflow
names the parallel program and builds the program container.

%irm_sc_save_jobflow
saves the program file to the job flow folder in your personal federated area.

%irm_sc_execute_jobflow
executes the parallel program.

%irm_sc_show_jobflow_results
shows the results library and the link to the PFD SAS Infrastructure for Risk Management server.

SetUp Environment for Using the Scripting Client

The out-of-the-box scripting client runs in SAS Studio. However, to use the scripting client to create parallel programs, the system must be properly configured.

Before you begin to use the scripting client, complete the following tasks:

1. Ensure that the SAS Infrastructure for Risk Management server is in development mode.

   In the SAS Management Console **Plug-ins** tab, select **SAS Management Console** ⇨ **Application Management** ⇨ **SAS Application Infrastructure** ⇨ **IRM Mid-Tier Server** ⇨ **Advanced** and add the following system property and value:

Property Name	Property Value
com.sas.solutions.risk.irm.server.devmode	true

2. Ensure that you have access to the SAS Infrastructure for Risk Management server file system.
3. Ensure that you can log on to SAS Studio (that is connected to a SAS Infrastructure for Risk Management server) using a valid user account.
4. Generate your personal federated area.

   Log on to the SAS Infrastructure for Risk Management server or to SAS Studio with your account and run the %irm_sc_init() macro.
5. In UNIX, create a symbolic link to your personal federated area on the SAS Infrastructure for Risk Management server. In Windows, create a folder shortcut to your personal federated area. This enables you to access your personal federated area from SAS Studio.

   To create a connection, you must have Read and Write permissions to your personal federated area.

   In addition, you can create symbolic links to federated areas other than your own federated area (that is, non-personal federated areas). This enables you to access tasks from these federated areas that you can use in your programs.

   Note: Access to federated areas that are not your personal federated area must be Read-Only. Ensure that you do not modify any federated area other than your personal federated area. The only exceptions are when you are loading data via

the input area at run time or via the landing area when the SAS Infrastructure for Risk Management server is down.

6. Ensure that Doxygen is installed and configured on your system.

   For information about installing Doxygen on your system, refer to the Doxygen documentation:

   http://www.stack.nl/~dimitri/doxygen/index.html

7. After installing Doxygen, ensure that you update the following SAS Management Console properties located in **Plug-ins** ⇨ **Application Management** ⇨ **Configuration Manager** ⇨ **SAS Application Infrastructure** ⇨ **IRM Mid-Tier Server** ⇨ **Advanced**:

Property Name	Property Value
com.sas.solutions.risk.irm.sc.doxygen.path	*path-to-the-Doxygen-binary-file*

   The path varies depending on the location of where you installed Doxygen. For example:

   - Windows:

     `C:\Program Files\doxygen\bin\doxygen.exe`

   - Linux:

     `/usr/bin/doxygen`

Chapter 4

Creating a Basic Parallel Program

A Basic Parallel Program 21

Identify the Tasks That You Need in Your Parallel Program 22

Write the Script That Creates the Hello World Parallel Program 23

View the Results 25

View the Job Flow Diagram in the SAS Infrastructure for Risk Management Web Application 25

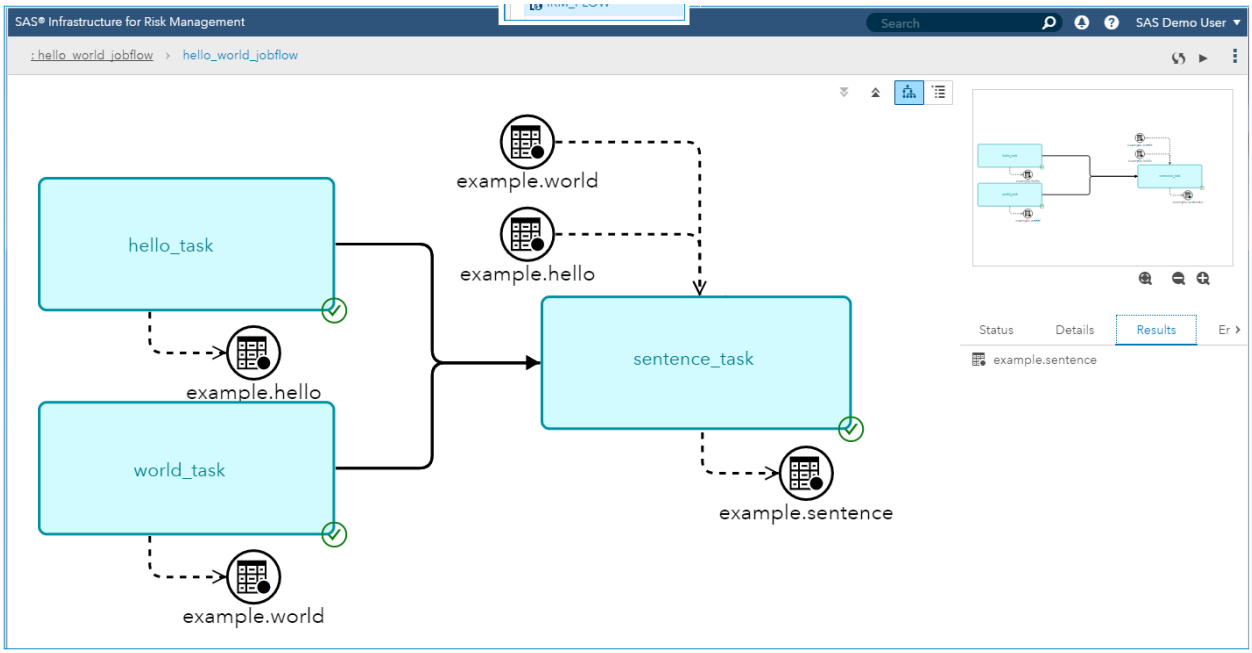
View the Results Library 25

A Basic Parallel Program

This chapter documents the steps that you must complete to create a basic parallel program for SAS Infrastructure for Risk Management that is named Hello World.

Here is how the Hello World parallel program appears in the SAS Infrastructure for Risk Management web application interface:

Figure 4.1 *Hello World Parallel Program Displayed in the SAS Infrastructure for Risk Management Web Application*



Note: A tutorial about creating this basic program is also included in this release. For more information about the tutorial, see "SAS Scripting Client: A Tutorial"..

Identify the Tasks That You Need in Your Parallel Program

A SAS Infrastructure for Risk Management task is an atomic unit of work that is organized into a program. A task must contain defined inputs, outputs, or both inputs and outputs. You define this data when you create a task by using the Doxygen tag**`\param dir`**, where ***`dir`*** specifies whether the data file is an input file or output file.

You combine the individual tasks that are needed to complete a job in a job flow. The SAS Infrastructure for Risk Management platform determines the tasks' dependencies and runs the tasks in parallel, based on the availability of required input data and the number of cores available for processing. Therefore, you can add the tasks to your parallel program in any order.

Tasks are stored in the **`/source/sas/nodes/`** folder of a federated area. You can create tasks to use in your parallel programs, or you can use existing tasks.

As seen in the previous figure, the Hello World program uses the following three tasks:

1. hello_task

```
/**
  \param[out] example.hello.sas7bdat
*/

data example.hello;
   greeting =  'Hello" ;
run;
```

2. world_task

```
/**
  \param[out] example.world.sas7bdat
*/

data example.world;
   to whom =  'Hello" ;
run;
```

3. sentence_task

```
/**
  \param[in] example.hello.sas7bdat
  \param[in] example.world.sas7bdat
  \param[out] example.sentence.sas7bdat
*/

data example.sentence;
   merge example.hello
         example.world;
   say_what = greeting || ',' ||
              to_whome || '!" ;
run;
```

Note: You can create the tasks and the script that are used to create the Hello World parallel program. Alternatively, you can copy the elements of the Hello World parallel program from the sample federated area (fa.sample.3.4).

Write the Script That Creates the Hello World Parallel Program

After you have identified or created the tasks that you want to include in your parallel program, you are ready to write the script that generates the parallel program.

To write the sample script, complete the following steps:

1. In SAS Studio, click ✱ and select **SAS Program** to open a new program tab in the work area. In the CODE window of the work area, create your script.
2. Initialize the scripting client.

```
%irm_sc_init();
```

3. Build the container for the job flow.

```
%irm_sc_build_jobflow(
        i_jf_name      =hello_world_jobflow,
        o_jf_ref_name  =example_jf_ref);
```

where :

- The value of **`i_jf_name`** is the name of the job flow (parallel program) that is created when you run the script. In this example, the name of the script is **`hello_world_jobflow`**. A valid value is a name that consists of up to 32 alphanumeric characters with no spaces.

- The value of **o_jf_ref_name** is the reference name to the job flow definition. This is the name that appears in the SAS Infrastructure for Risk Management web application.

  Note: Throughout the script, when you use the reference name, you precede the name with an ampersand (&). For example, in the script, the value for **o_jf_ref_name** is referenced as **&example_jf_ref**.

4. Specify the tasks that need to be executed to complete the job.

```
%sentence_task(
        i_jf_ref         =&example_jf_ref,
        i_task_name      =sentence_task;

%hello_task(
        i_jf_ref         =&example_jf_ref,
        i_task_name      =hello_task;

%world_task(
        i_jf_ref         =&example_jf_ref,
        i_task_name      =world_task;
```

where:

- The value for ***%name*** is the name of the macro that creates the task. The name is user-defined and is preceded by a percentage sign (%). In this example, the names used to create the tasks are **%sentence_task**, **%hello_task**, and **%world_task**.
- The value for **i_jf_ref** is the reference name to the job flow. Ensure that you precede the name with an ampersand (&). In the script, the value is **&example_ij_ref**.
- The value for **i_task_name** is the name of the task.

Note: SAS Infrastructure for Risk Management determines the order and parallelism in which the program executes tasks that are based on the availability of specified inputs for a task. Therefore, you can define the tasks in your script in any order.

5. Save the job flow definition to the SAS Infrastructure for Risk Management server and generate the parallel program.

```
%irm_sc_save_jobflow(
        i_jf_ref   =&example_jf_ref);
```

6. Execute the parallel program to create a job flow instance.

```
%irm_sc_execute_jobflow(
        i_jf_ref   =&example_jf_ref);
```

7. Click to run the script.

When you execute the job flow script, the script performs the following actions:

1. It generates the Hello World parallel program (also called the job flow definition).
2. It executes the parallel program, which creates a job flow instance of the parallel program (job flow definition).
3. It creates a shortcut link (IRM_FLOW) to the resulting job flow diagram, and it assigns a library reference (RESULTS) to the results library.

Here is an example of the job flow script that was created by the preceding steps:

```
/* Scripting Client Initialization - Required setup */
%irm_sc_init();

/* Add job flow -- Build jobflow container */
%irm_sc_build_jobflow(
        i_jf_name        =hello_world_jobflow,
        o_jf_ref_name    =example_jf_ref);

/* Add task -- sentence_task */
%sentence_task(
        i_jf_ref         =&example_jf_ref,
        i_task_name      =sentence_task);

/* Add task -- hello_task */
%hello_task(
        i_jf_ref         =&example_jf_ref,
        i_task_name      =hello_task);

/* Add task -- world_task */
%world_task(
        i_jf_ref         =&example_jf_ref,
        i_task_name      =world_task);

/* Save this job flow definition to IRM Server */
%irm_sc_save_jobflow(
        i_jf_ref         =&example_jf_ref);

/* Execute job flow instance in IRM Server */
%irm_sc_execute_jobflow(
        i_jf_ref         =&example_jf_ref);
```

View the Results

After you have successfully executed the script, you can view the job flow diagram (see Figure 4.1.) or access the job flow instance results tables in the RESULTS library.

View the Job Flow Diagram in the SAS Infrastructure for Risk Management Web Application

When you successfully execute the hello_world_jobflow script, a file shortcut link named IRM_FLOW is created under **File Shortcuts** in the SAS Studio Navigation pane. To open the job flow diagram in the SAS Infrastructure for Risk Management web application, double-click the IRM_FLOW file shortcut.

Note: The link is only for the current job flow instance that is created in a SAS Studio session. It requires your user password to be stored in metadata.

View the Results Library

When you successfully execute the hello_world_jobflow script, a RESULTS library is created under **My Libraries** in the SAS Studio Navigation pane.

This library contains the job flow instance output tables for the current script execution. For the Hello World program, there is single output file named sentence.sas7bdat.

To produce a simple report using the results output, reference the output table in the RESULTS library in your code:

```
proc print data = results.sentence;
run;
```

To view the report, click the **RESULT** tab in the SAS Studio workspace.

Chapter 5

Using Additional Parallel Programming Features

About Scripting Client Parallel Programming Features 27

Create Tasks 27
About Creating Tasks 27
Doxygen Syntax 28

Generate Doxygen Documentation 30

Use Parameter Substitution 30

Create a Job Flow 30

Create and Add SubFlows 30

Use Partitions 33
Partition Methods 33
By Variable Partitions 33
Partition Tasks 34
Creating a Script That Uses a Simple ByN Partition 34
Creating a Script That Uses a Simple ByVar Partition 37

Use Generic and Temporary Libraries 41

Create and Add a New Federated Area 42

About Scripting Client Parallel Programming Features

This chapter documents several primary scripting client parallel programming features.

Create Tasks

This section describes how to create tasks that you use in your parallel program. Tasks are independent programs that are combined in a job flow to form a job.

About Creating Tasks

Before you begin creating tasks, note the following characteristics:

- Tasks are located in the **source/sas/nodes** folder of your personal federated area.
- Each task is defined in only one file.
- The task filename must be in lowercase, must not exceed 32 characters (excluding the suffix), and must follow the SAS naming convention. In addition, all task filenames must end with the .sas file extension.
- A task file must contain a Doxygen header. The scripting client relies on the syntax in the Doxygen header of a task to create the tasks' packages that you use when developing job flows.

Doxygen Syntax

When you create a task, you must include a Doxygen header in the task file. In the header, you must comply with the syntax rules in the following sections.

Task Name

By default, a task name comes from its filename but without the extension. For example, for a task file named rm_node_enrich_loans.sas, irm_node_enrich_loans is used as the task name. No user action is needed.

Task Input and Output Tables

Input tables and output tables in a task are treated as input and output function parameters in Doxygen. Therefore, in each task source file, you must document the input and output tables, table substitution, or data collection by using the following syntax:

```
\param[dir] parameter-name parameter-description
```

where:

- *dir* — specifies whether the table is an input table or an output table. Choices are **in** and **out**.
- *parameter-name* — specifies the name of the data source that you are defining as input or output for the task.
- *parameter-description* — (Optional) describes the table. The description can contain multiple spaces and tabs.

Table Substitution

To denote table substitution, use the percent sign (%) as the prefix for the table substitution token.

If a default table substitution value is provided, assign it using the equal sign (=) and its corresponding table substitution value. For example, if you have the table substitution token FILTERED_LOANS and you assign its default value to R_STAGE.filtered_loans as an input table for a task, you use the following Doxygen syntax:

```
\param[in] %FILTERED_LOANS=RD_STAGE.filtered_loans    filtered loans file
```

Note: When defining table substitution, ensure that there are no spaces or tabs before or after the equal sign (=).

Collection Input

To denote collection data (such as data library, a file folder, or a format catalog) as input data and output data for a task, follow the SAS libref naming rule. Here is an example:

```
\param[in] %VALIDATION_COLLECTION=validtn     validation input folder
```

Task Documentation

You can document the details of the function of a task while coding the task. When you process the content that you are developing, this documentation is available for users to reference.

To document the details about the calculations a task is performing, use the following syntax:

```
\file
  \brief    Include brief description of the task.

  \details  Include detailed information about the calculations
            that the task is performing.
```

Doxygen Header Sample

Here is an example the Doxygen header for a task file:

```
/*
 Copyright (C) 2017 SAS Institute Inc. Cary, NC, USA
*/

/**
\file

\brief    The task irm_node_enrich_loans enriches the filtered loan data

\details  This task contains dynamics assigned tables names, which enables this task to be used for
both regular Market SCR and Hypothetical Market SCR calculation.


<b> Indentified Inputs and Outputs </b>

\param[in] %FILTERED_LOANS=RD_STAGE.filtered_loans.sas7bdat  Filtered loans table

\param[in] RD_STAGE.ratings.sas7bdat                         Ratings data set

\param[in] RD_CONF.analytics_option.sas7bdat                 Analytics option

\param[out] %LOANS                                           Output enriched loan table


\ingroup Nodes

\author SAS Institute INC.

\date 2016
*/
```

Generate Doxygen Documentation

To generate the documentation that is associated with a task, use the **%irm_sc_gen_doc macro**.

Use Parameter Substitution

All tasks must have declared input and output. However, there is some flexibility in how you declare their inputs and outputs. This flexibility enables you to reuse existing code for different jobs.

Inputs and outputs that are declared in job flow files are passed as defined to the tasks for execution. This prevents a given task from being reused with different inputs or outputs for a different purpose. However, you can specify a parameter substitution for each input and output. When you specify a parameter substitution for an input file or an output file, the code that is associated with the task can check the value of the data state to separate the inputs or outputs.

The convention is the same for all tasks, regardless of the language: the parameter is specified as **%<parameter-name>** in the data state as in **%qrtdata, %datapoint, %excel_template**, and **%validation**.

Create a Job Flow

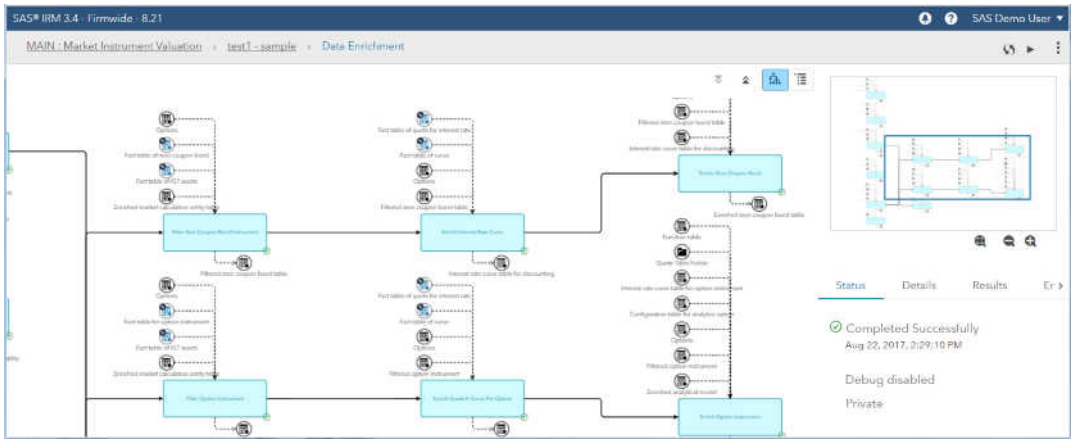
For information about how to create a basic job flow, see Chapter 4, "Creating a Basic Parallel Program," on page 21.

Create and Add SubFlows

Tasks that contain subflows are denoted in the SAS Infrastructure for Risk Management web application as follows:

To open a task, double-click the task in the SAS Infrastructure for Risk Management web application. The subflows are displayed:

Figure 5.1 *Task Subflows*



To create subflows, complete the following tasks:

1. Create a subflow task.

```
%subflow_task(
              i_jf_ref           = &subflow,
              i_task_name       = subflow_task);
```

   where:

   - The value for ***%name*** is the name of the macro that creates the task. The name is user-defined and preceded by a percent sign (%). In this example, the name is subflow_task.
   - The value for **i_jf_ref** is the name of the reference to the job flow for which this task is a subflow. Ensure that you precede the name with an ampersand (&). In this example, the reference name for the job flow is subflow.
   - The value for **i_task_name** is the name of the task.

2. Configure the inputs and outputs of the subflow to be displayed in the UI.

```
%irm_sc_add_visual_data(
              i_jf_ref        = &subflow,
              i_visual_input_data_list
              =MK_CONF:MK_CONF.one.sas7bdat);
```

   where:

   - The value for **i_jf_ref** is the reference name for the subflow, which is preceded by &.
   - The value for **i_visual_input_data_list** is the name of the input.

3. Create and add the subflow to the job flow.

```
%irm_sc_add_sub_jobflow(
              i_jf_ref           =  &test,
              i_sub_jf_ref     = &subflow);
```

   where:

   - The value for **i_jf_ref** is the reference name for the job flow, which is preceded by &.

- The value for **i_sub_jf_ref** is the reference name for the task that you are adding as a subflow, which is preceded by &.

Here is an example of creating and adding a subflow:

```
/*
Copyright (c) 2017 by SAS Institute Inc., Cary, NC, USA.
*/
/* Scripting Client Initialization */
%irm_sc_init(i_system_debug_on=TRUE);
/*************************************************************/
/* Begin: User defined jobflow */
/*************************************************************/
%macro user_def_jobflow();

   /* create top level job flow */
   %irm_sc_build_jobflow(
           i_jf_name      = folder_test_6,
           o_jf_ref_name  = test);

   %irm_sc_build_jobflow(
              i_jf_name      = folder_subflow,
              o_jf_ref_name  = subflow);

   /*Add task nodes*/
   %folder_output(
              i_jf_ref           = &test,
              i_task_name       = folder_output);

   %subflow_node1(
              i_jf_ref           = &subflow,
              i_task_name       = subflow_task);

   /*Add visual data to subflow*/
   %irm_sc_add_visual_data(
              i_jf_ref        = &subflow,
              i_visual_input_data_list
              =MK_CONF:MK_CONF.one.sas7bdat);

   /*Add subflow*/
   %irm_sc_add_sub_jobflow(
              i_jf_ref           =  &test,
              i_sub_jf_ref    = &subflow);

   /* Sample: Save job flow definition to IRM Server */
   %irm_sc_save_jobflow(
           i_jf_ref                   =&test);

   /* Sample: Execute job flow instance in IRM Server */
   %irm_sc_execute_jobflow(
               i_jf_ref                   =&test);


/*************************************************************/
/* End: User defined jobflow */
/*************************************************************/
```

```
%mend;
/*************************************************************/
/* Execute: User defined jobflow */
/*************************************************************/
%user_def_jobflow();
```

Use Partitions

The best way to reduce complexity and to improve performance in SAS Infrastructure for Risk Management is to use partitions. If your problem can be divided into independent chunks of data, then you should use partitions. Using partitions enables large data files to be partitioned into smaller units of data and calculated across multiple cores. Use of partitions improves performance for large tables. After all calculations are complete, a task recombines the results of the partitioned data.

Partition Methods

SAS Infrastructure for Risk Management supports the following two partition methods:

- By Number (ByN) — Data is partitioned by a user-defined number of partitions.
- By Variable (ByVar) — Data is partitioned by user-defined variables.

By Variable Partitions

The easiest way to apply partitions is to use one or more *by variables*. When you use the ByVar partition method, the data is partitioned by user-defined variables. SAS Infrastructure for Risk Management automatically creates the partitions: one partition per unique value of the concatenated values of the BY variables.

SAS Infrastructure for Risk Management also automatically recombines partitions for processing by tasks that do not support multi-threading and for displaying in the SAS Infrastructure for Risk Management web application.

The canonical forms of a partitioned flow follow:

1. One task to calculate the number of partitions

   This task needs to calculate the number of partitions and to put that value in a data object. The data object should contain only one record and one column, which must be called MAX_RANK_NO.

2. One task to partition one or more data objects

   This task creates the partitions. One instance of the partitioning task runs per partition in order to create the partitions. Each task receives a %rank SAS macro variable in order to determine which partition needs to be generated.

3. One or more computational tasks

   One or more of these tasks do not need to do anything special for partitioning. They have inputs and outputs that are partitioned. The partitioned inputs and outputs must be labeled with the partition data state. The computation tasks in a partitioned flow have one important restriction: in one computational task, the task can have only one output per LIBNAME. It is acceptable to have some inputs that are non-partitioned. In that case, all partitions (or instances of the computational tasks) will receive the

same input. In the job flow diagram, see the computation task. Notice the partition data state that is attached to the input and output. (See Figure 5.3 on page 38.)

4. One recombining task

   This task performs the reverse operation of the partitioning task. Only one instance of the recombining task will be executed: the partitioned inputs, which must be labeled with the partition data state. The output cannot be partitioned.

Partition Tasks

The steps to add a task with partitioned data in a job flow script might vary according to the method that you choose. However, at minimum you must use the following four tasks:

1. Calculate Cardinality Task — Calculates the number of partitions (called cardinality) and produces the cardinality output. If using the ByVar method, this task also creates a partition mapping table.

   This task is created by the following scripting client macros:

   - **`%irm_node_get_cardinality_byn`**
   - **`%irm_node_get_cardinality_byvar`**

2. Partition the Data Task — Partitions the input data.

   This task is created by the following scripting client macros:

   - **`%irm_sc_add_partition_byn_task`**
   - **`%irm_sc_add_partition_byvar_task`**

3. Computational Task — Calculates the partitioned data.
4. Recombine Task — Recombines the partitioned results.

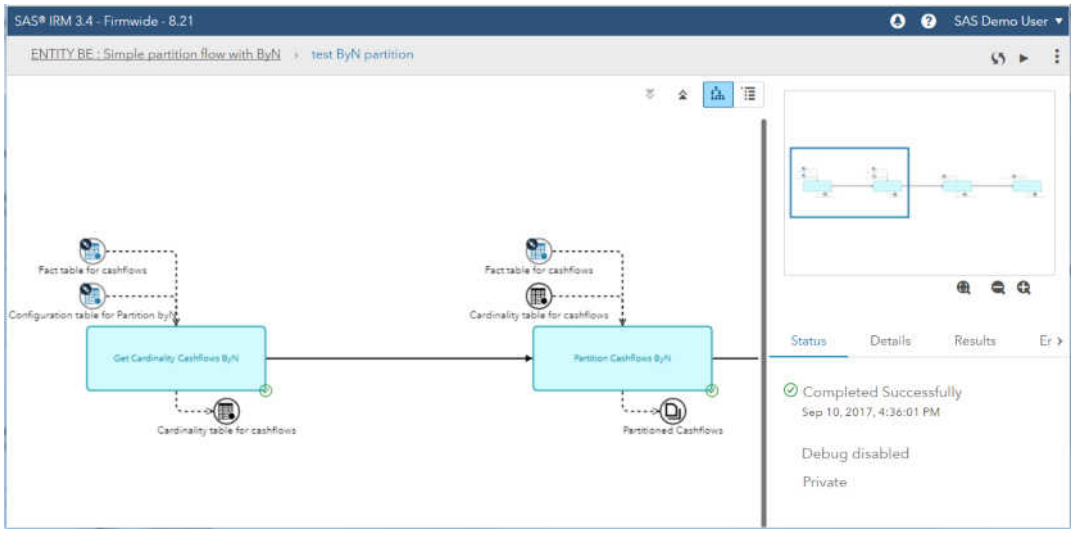
   This task is created by the following scripting client macros:

   - **`%irm_sc_add_recombine_byn_task`**
   - **`%irm_sc_add_recombine_byvar_task`**

Creating a Script That Uses a Simple ByN Partition

Here is how a portion of a job flow that uses ByN partitioning appears in the SAS Infrastructure for Risk Management web application interface:

Figure 5.2 *Jobflow with Simple ByN Partition*



To write a script that generates a job flow, complete the following steps:

1. In SAS Studio, click ✱ and select **SAS Program** to open a new program tab in the work area. In the CODE window of the work area, create your script.
2. Initialize the scripting client.

```
%irm_sc_init();
```

3. Build the container for the script.

```
/* Add job flow -- build job flow container  */
%irm_sc_build_jobflow(
        i_jf_name      =simple_byn_partition_calculation_flow,
        o_jf_ref_name  =simp_byn_flow_ref);
```

4. Create a task that calculates cardinality and produces the cardinality output table.

```
%irm_node_get_cardinality_byn(
          i_jf_ref          =,
          i_task_name       =,
          t_in_ds_part_byn  =,
          t_in_ds_byn_conf  =,
          t_out_card_ds_byn =)
```

where:

- **`i_jf_ref`** is the reference name for the job flow. Ensure that you precede the name with an ampersand (&).
- **`i_task_name`** is the name of the task.
- **`t_in_ds_part_byn`** is the name of the input table to be partitioned.
- **`t_in_ds_byn_conf`** is the name of the ByN configuration table.
- **`t_out_card_ds_byn`** is the name of the output cardinality table.

5. Create a task that partitions the input data.

```
%irm_sc_add_partition_byn_task(
          i_jf_ref           =,
          i_task_nm          =,
```

```
            i_card_tb_byn      =,
            i_part_ds_byn      =,
            i_out_part_tb_byn  =);
```

where:

- **i_jf_ref** is the reference name for the job flow. Ensure that you precede the name with an ampersand (&).
- **i_task_name** is the name of the task.
- **i_card_tb_byn** is the name of the input cardinality table.
- **i_part_ds_byn** is the name of the input table to be partitioned.
- **i_out_part_tb_byn** is the name of the output table for the partitioned data.

6. Create a task that recombines the partitioned data.

```
%irm_sc_add_recombine_byn_task(
            i_jf_ref           =,
            i_task_nm          =,
            i_rec_tb_in_byn    =,
            i_rec_out_tb_byn   =);
```

where:

- **i_jf_ref** is the reference name for the job flow. Ensure that you precede the name with an ampersand (&).
- **i_task_name** is the name of the task.
- **i_rec_tb_in_byn** is the name of the partitioned input table.
- **i_rec_out_tb_byn** is the name of the recombined output table.

Here is a sample script that generates a parallel program that uses a simple ByN partition:

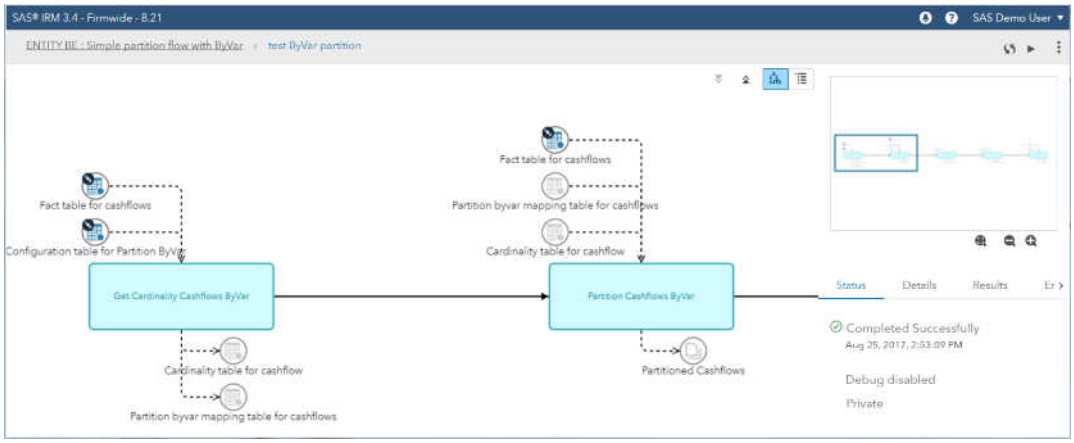
```
/*
Copyright (c) 2017 by SAS Institute Inc., Cary, NC, USA.
*/

/* Scripting Client Initialization */
%irm_sc_init();
/*******************************************************************************/
/* Begin: User defined jobflow */
/*******************************************************************************/
%macro user_def_jobflow();

   /* create job flow */
   %irm_sc_build_jobflow(
           i_jf_name              =simple_byn_partition_calculation_flow,
           o_jf_ref_name          =simp_byn_flow_ref);

   /* call get cardinality API in the job flow*/
   %irm_node_get_cardinality_byn(i_jf_ref =&simp_byn_flow_ref,
           i_task_name            =get_cardinality_cashflows_byn);

   /* call SAS Scripting client macro to add partition task*/
   %irm_sc_add_partition_byn_task(
           i_jf_ref               =&simp_byn_flow_ref,
           i_task_name            =partition_cashflows_byn,
```

```
        i_card_tb_byn          =MK_CARD.card_cashflows_byn.sas7bdat,
        i_part_ds_byn          =STAGING.cashflows.sas7bdat,
        i_out_part_tb_byn      =P_STAGE.cashflows.sas7bdat);

   /* add task to calculation flow */
   %discount_single_cashflow(
        i_jf_ref                =&simp_byn_flow_ref,
        t_in_inst_discount      =P_STAGE.cashflows.sas7bdat,
        t_out_inst_discount     =P_MK_CF.cashflows.sas7bdat,
        i_task_name             =discount_cashflow);

   /* call SAS Scripting client macro to add recombine task */
   %irm_sc_add_recombine_byn_task(
        i_jf_ref                =&simp_byn_flow_ref,
        i_task_name             =recombine_cashflow_byn,
        i_rec_tb_in_byn         =P_MK_CF.cashflows.sas7bdat,
        i_rec_out_tb_byn        =R_MK_CF.cashflows_result.sas7bdat);

   /*Save job flow definition to IRM Server */
   %irm_sc_save_jobflow(
        i_jf_ref                =&simp_byn_flow_ref,
        i_status_code_var_name  =IRM_SC_SAVE_DEF_CODE);


   /*Execute job flow instance in IRM Server */
   %if &IRM_SC_SAVE_DEF_CODE eq 200 %then %do;
   %irm_sc_execute_jobflow(
        i_jf_ref                  =&simp_byn_flow_ref,
        i_status_code_var_name    =IRM_SC_EXE_DEF_CODE);
   %end;
   %else %do;
       %put ERROR: failed to save definition;
   %end;

   %if &IRM_SC_EXE_CODE ne 200 %then %do;
       %put ERROR: failed to execute job flow;
   %end;

/*******************************************************************************/
/* End: User defined jobflow */
/*******************************************************************************/

%mend;
%user_def_jobflow();
```

Creating a Script That Uses a Simple ByVar Partition

Here is how a portion of a job flow that uses ByVar partitioning appears in the SAS Infrastructure for Risk Management web application interface.

Figure 5.3 *Job Flow with Simple ByVar Partition*



To write a script that generates a job flow, complete the following steps:

1. In SAS Studio, click ✱ and select **SAS Program** to open a new program tab in the work area. In the CODE window of the work area, create your script.
2. Initialize the scripting client.

```
%irm_sc_init();
```

3. Build the container for the script.

```
/* Add job flow -- build job flow container  */
%irm_sc_build_jobflow(
        i_jf_name      =simple_byn_partition_calculation_flow,
        o_jf_ref_name  =simp_byn_flow_ref);
```

4. Create a task that calculates the cardinality and that produces the cardinality output table and the ByVar mapping table.

```
%irm_node_get_cardinality_byn(
          i_jf_ref            =,
          i_task_name         =,
          t_in_ds_part_byvar  =,
          t_in_ds_byvar_conf  =,
          t_out_card_ds_byvar =,
          t_out_card_map      =)
```

where:

- **`i_jf_ref`** is the reference name for the job flow. Ensure that you precede the name with an ampersand (&).
- **`i_task_name`** is the name of the task.
- **`t_in_ds_part_byvar`** is the name of the input table to partition.
- **`t_in_ds_byvar_conf`** is the name of the input ByVar configuration table.
- **`t_out_card_ds_byn`** is the name of the output cardinality table.
- **`t_out_card_map`** is the name of the output cardinality mapping table.

5. Create a task that partitions the input data.

```
%irm_sc_add_partition_byn_task(
          i_jf_ref            =,
```

```
          i_task_nm            =,
          i_card_tb_byvar      =,
          i_part_ds_byvar      =,
          i_out_part_tb_byvar  =);
```

where:

- **i_jf_ref** is the reference name for the job flow. Ensure that you precede the name with an ampersand (&).
- **i_task_name** is the name of the task.
- **i_card_tb_byvar** is the name of the input cardinality table.
- **i_part_ds_byvar** is the name of the input data set to be partitioned.
- **i_card_map_byvar** is the name of the input table to be partitioned.
- **i_out_part_tb_byvar** is the name of the output table for the partitioned data.

6. Create a task that recombines the partitioned data.

```
%irm_sc_add_recombine_byvar_task(
          i_jf_ref              =,
          i_task_nm             =,
          i_rec_tb_in_byvar     =,
          i_rec_card_map_byvar  =,
          i_rec_out_tb_byvar    =);
```

where:

- **i_jf_ref** is the reference name for the job flow. Ensure that you precede the name with an ampersand (&).
- **i_task_name** is the name of the task.
- **i_rec_tb_in_byvar** is the name of the partitioned input table.
- **i_rec_card_map_byvar** is the name of the partitioned input table.
- **i_rec_out_tb_byvar** is the name of the recombined output table.

Here is a sample script that generates a parallel program that uses a simple ByVar partition:

```
/*
Copyright (c) 2017 by SAS Institute Inc., Cary, NC, USA.
*/

/* Scripting Client Initialization */
%irm_sc_init();
/*******************************************************************************/
/* Begin: User defined jobflow */
/*******************************************************************************/
%macro user_def_jobflow();

   /* create the job flow */
   %irm_sc_build_jobflow(
           i_jf_name             =simple_byvar_partition_calculation_flow,
           o_jf_ref_name         =simp_byvar_flow_ref);

   /* call get cardinality API in the jobflow*/
   %irm_node_get_cardinality_byvar(
```

```
        i_jf_ref                =&simp_byvar_flow_ref,
        t_in_ds_part_byvar      =STAGING.cashflows.sas7bdat,
        t_in_byvar_conf         =IRM_CFG.byvar_configuration.sas7bdat,
        t_out_card_ds_byvar     =MK_CARD.card_filtered_cashflows.sas7bdat,
        t_out_card_map          =mk_card.part_cashflow_mapping.sas7bdat,
        i_task_name             =get_cardinality_cashflows);

/* call SAS Scripting client macro to add partition task*/
%irm_sc_add_partition_byvar_task(
        i_jf_ref                =&simp_byvar_flow_ref,
        i_task_name             =partition_cashflows_byvar,
        i_card_tb_byvar         =MK_CARD.card_filtered_cashflows.sas7bdat,
        i_part_ds_byvar         =STAGING.cashflows.sas7bdat,
        i_card_map_byvar        =mk_card.part_cashflow_mapping.sas7bdat,
        i_out_part_tb_byvar     =P_STAGE.cashflows.sas7bdat);

/* add tasks to calculation flow */
%discount_single_cashflow(
        i_jf_ref                =&simp_byvar_flow_ref,
        t_in_inst_discount      =P_STAGE.cashflows.sas7bdat,
        t_out_inst_discount     =P_MK_CF.cashflows.sas7bdat,
        i_task_name             =discount_cashflow);

%sum_data(
        i_jf_ref                =&simp_byvar_flow_ref,
        t_in_inst               =P_MK_CF.cashflows.sas7bdat,
        t_out_inst              =P_MK_CF.summed_cashflows.sas7bdat,
        i_task_name             =sum_cashflow);

/* call SAS Scripting client macros to add recombine task */
%irm_sc_add_recombine_byvar_task(
        i_jf_ref                =&simp_byvar_flow_ref,
        i_task_name             =recombine_cashflow_byvar,
        i_rec_tb_in_byvar       =P_MK_CF.summed_cashflows.sas7bdat,
        i_rec_card_map_byvar    =mk_card.part_cashflow_mapping.sas7bdat,
        i_rec_out_tb_byvar      =R_MK_CF.cashflows_result.sas7bdat);


/* Save job flow definition to IRM Server */
%irm_sc_save_jobflow(
        i_jf_ref                =&simp_byvar_flow_ref,
        i_status_code_var_name =IRM_SC_SAVE_DEF_CODE);


/* Execute job flow instance in IRM Server */
%if &IRM-SC_SAVE_DEF_CODE eq 200 %then %do;
%irm_sc_execute_jobflow(
        i_jf_ref                =&simp_byvar_flow_ref,
        i_status_code_var_name =IRM_SC_EXE_DEF_CODE);
%end;
%else %do;
    %put ERROR: failed to save definition;
%end;

%if &IRM_SC_EXE_DEF_CODE ne 200 %then %do;
    %put ERROR: failed to execute job flow;
```

```
   %end;

/*****************************************************************************/
/* End: User defined jobflow */
/*****************************************************************************/

%mend;
%user_def_jobflow();
```

Use Generic and Temporary Libraries

For information about using generic and temporary libraries, see *SAS Infrastructure for Risk Management 3.4: Administrator's Guide*.

```
/*
Copyright (c) 2017 by SAS Institute Inc., Cary, NC, USA.
*/
/*************************************************************/
/* Scripting Client Initialization */
%irm_sc_init(i_system_debug_on=TRUE);
/*************************************************************/
/* Begin: User defined jobflow */
/*************************************************************/
%macro user_def_jobflow();

   /* create top level job flow */
   %irm_sc_build_jobflow(
              i_jf_name      =download_postgres,
              o_jf_ref_name  =postgres);


   /*Add task nodes*/
   %test_platform_pgstable2sasdatase(
              i_jf_ref           =&postgres,
              i_task_name       =download_from_postgres);


   /* Sample: Save job flow definition to IRM Server */
   %irm_sc_save_jobflow(
           i_jf_ref       =&postgres);


   /* Sample: Execute job flow instance in IRM Server */
   %irm_sc_execute_jobflow(
               i_jf_ref   =&postgres);


/*************************************************************/
/* End: User defined jobflow */
/*************************************************************/

%mend;
```

```
/*************************************************************/
/* Execute: User defined jobflow */
/*************************************************************/
%user_def_jobflow();
```

Create and Add a New Federated Area

The platform federated area is installed by SAS Infrastructure for Risk Management. The purpose of the platform federated area is to deliver software that is essential for the platform to run correctly. For example, the code that is necessary for the execution of Java code is delivered in the platform federated area.

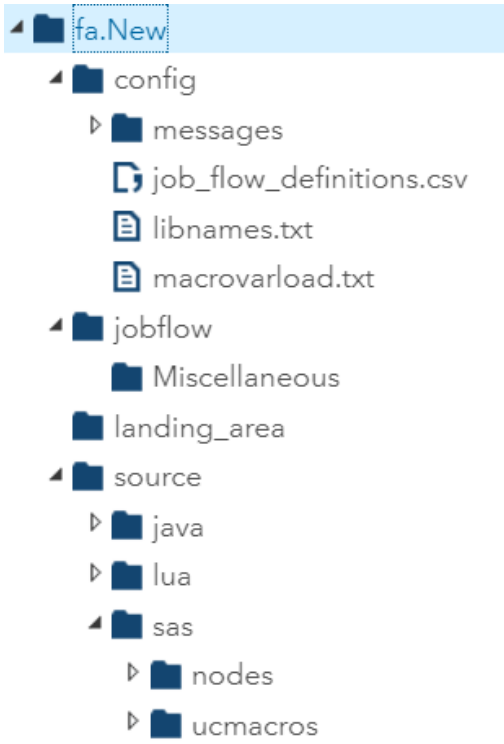
When creating a new federated area, understand that all federated areas must be integrated with the platform federated area.

To add a new federated area, complete the following steps:

1. Stop the SAS Infrastructure for Risk Management web application server. For example, for a non-clustered environment, you might stop SASServer8_1. For a clustered environment, you might also include SASServer8_2, SASServer_3, and so on.

   These servers can be on the same machine or on different machines within the cluster. For information about stopping SAS Web Application Servers, see *SAS Intelligence Platform: Middle-Tier Administration Guide*.

2. Create and populate the new federated area. Grant Read permissions to the SAS General Servers user.



3. Add the property of the new federated area in SAS Management Console by completing the following steps:

a. Open SAS Management Console.

b. Connect to the appropriate metadata server.

c. Enter the password for the SAS Administrator account.

d. On the **Plug-ins** tab, verify that the repository is selected in the **Repository** field. The default repository name is Foundation.

e. In the Navigation pane, select **Application Management** ⇨ **Configuration Manager** ⇨ **SAS Application Infrastructure**.

f. Right-click **SAS IRM Mid-Tier Server** and select **Properties**. The IRM Mid-Tier Server Properties dialog box is displayed.

g. Click **Advanced** and then click **Add**. The Define New Property dialog box is displayed.

h. In the **Property Name** field, enter `com.sas.solutions.risk.irm.fa.`*n*, where *n* is the identifier for the federated area. This identifier must not be zero (0) or start with 0. For example, do not assign the ID fa.0.3.4_myfa to a federated area.

   Note: Typically, you want the new property to have precedence. Therefore, the value for *n* should be greater than previous federated version IDs.

   In the **Property Value** field, enter the path to the new federated area and click **OK**.

i. Click **OK**.

4. Grant Read permissions to the spawned server user on the federated area directory.

5. Restart the SAS Infrastructure for Risk Management web application server.

For detailed information about SAS Management Console, see *SAS Intelligence Platform: Middle-Tier Administration Guide*.

Appendix 1

Scripting Client: A Tutorial

SAS Scripting Client: A Tutorial 45
About This Tutorial 45
Getting Started – Hello World 46
Set Up the Environment for Using SAS Scripting Client 46
View the Job Tasks 47
Write the SAS Scripting Client Program 48
View and Consume Results 50
View the Job Flow Diagram 50
View Results Library and Consume Results 51
Wrap-Up 52

SAS Scripting Client: A Tutorial

The SAS Infrastructure for Risk Management platform is an execution engine for highly parallel code execution. The SAS Infrastructure for Risk Management platform enables the user to take advantage of multi-core, many task computing, and parallel computing technologies using only their existing SAS programming skills.

For code to execute on the SAS Infrastructure for Risk Management platform, the user must create it and lay it out in specific ways, and the user must follow some rules. The SAS Scripting Client (available in SAS Infrastructure for Risk Management 3.4) enables the user to generate highly parallel programs in a format that can be executed by the SAS Infrastructure for Risk Management platform. The SAS Scripting Client is basically a set of SAS macros that provides the developer with a way to use their existing SAS programming skills and to hide the complexity of developing parallel programs.

About This Tutorial

This tutorial provides minimal guidance to the developer in getting started to use the SAS Scripting Client to develop parallel programs on the SAS Infrastructure for Risk Management platform.

In this tutorial, you will use the SAS Scripting client to create a parallel program. Here is the basic process:

1. Write and execute the SAS Scripting Client program (job flow script).

   User writes and executes a SAS Scripting Client script in SAS Studio. Execution of this script will generate a parallel program (job flow definition).

2. Execute the parallel program (job flow definition) that was generated in Step 1.

   The script written in Step 1 includes the SAS Scripting Client API macro that submits the generated parallel program for execution. Execution of the generated parallel program creates an instance (job flow instance) of the job flow definition.

3. View and consume the results of the parallel program execution.

   The results of executing the parallel program include the Job Flow Diagram (a visual representation of the job flow instance) and any other final outputs (data sets, reports, and so on).

Getting Started – Hello World

You will start using the SAS Scripting Client with a Hello World example. Your objective is to create a parallel program that combines three tasks from the tutorial task library.

Here is an overview of the steps:

1. Set up our environment in SAS Studio to run the SAS Scripting Client.
2. View the tasks that you will include in your parallel program.
3. Write a SAS Scripting Client program (*job flow script*) that has these substeps:
   a. Generate the Hello World parallel program (also called a *job flow definition*).
   b. Execute the generated parallel program, which will create an instance (*job flow instance*) of the parallel program definition.
   c. Create a shortcut link (IRM_FLOW) to the resulting job flow diagram and assign a library reference (RESULTS) to the results library.
4. View and consume the results of the parallel program execution by producing a simple report.

Set Up the Environment for Using SAS Scripting Client

Note: The SAS Scripting Client runs out of the box in SAS Studio, but it does require that the system is properly configured.

1. Log on to SAS Studio and connect to the SAS Infrastructure for Risk Management 3.4 server with a valid user account (for example, sasdemo).
2. Select **Navigation Pane Display**.

In the example, it is helpful to have Folder Shortcuts, Libraries, and File Shortcuts that are visible in the SAS Studio navigation pane. (See Figure A1.1 on page 47.).

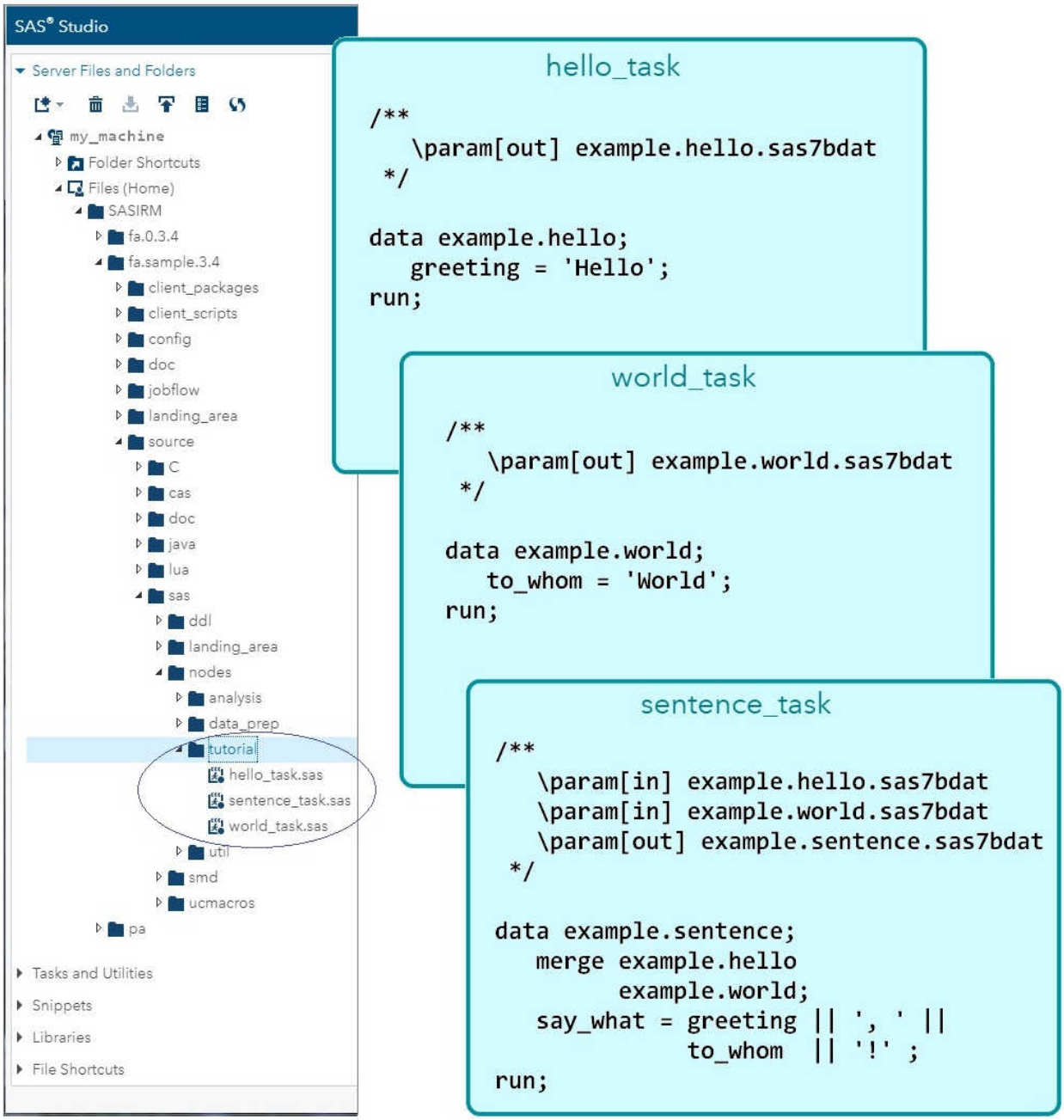
Figure A1.1 *Navigation Pane Display*

View the Job Tasks

In SAS Infrastructure for Risk Management, a task is an atomic unit of work organized into a program. The task must define its inputs and outputs. You can combine individual tasks as needed in order to complete a job. When you put multiple tasks into a parallel program, the SAS Infrastructure for Risk Management platform determines the task dependencies and the order or the tasks based on the inputs that are needed. The platform then runs tasks in parallel, based on the number of cores that are available for processing. The headers for the task that are shown in the tutorial contain the minimum required documentation tags **`(\param[in] and \param[out])`**. These tags are used by the SAS Scripting Client during job flow development, but are not sufficient for documentation purposes.

The tasks that are needed by the example are available in the SAS Infrastructure for Risk Management sample task library. Three tasks are used in the example. (See Figure A1.2 on page 48.)

Figure A1.2 *Hello World Tasks*



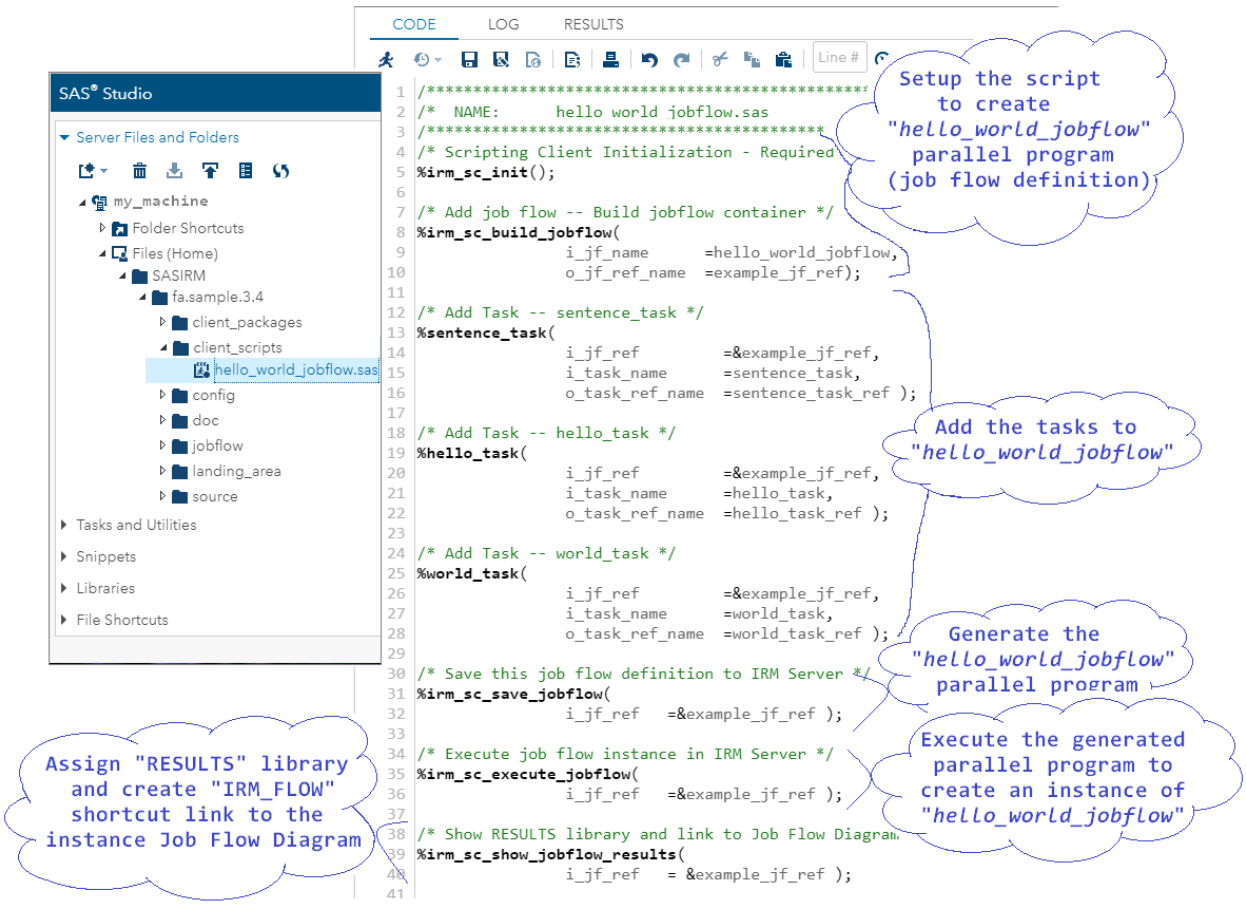
Write the SAS Scripting Client Program

A *job flow script* is the SAS program that you write. It combines the SAS Scripting Client macros and the job flow tasks that are needed to create your parallel program. SAS Infrastructure for Risk Management platform determines the order and the parallelism in which the parallel program executes tasks based on the availability of the task's specified inputs. Therefore, you can add the tasks to the script in any order.

In the SAS Studio New SAS Program window, enter the job flow script that will create your parallel program. The script (hello_world_jobflow.sas) can be copied from the SAS Infrastructure for Risk Management 3.4 Sample Client Scripts. (See Figure A1.3 on page 49.). Here are the tasks that the script performs:

1. Generates the Hello World parallel program (also called a job flow definition)
2. Executes the generated parallel program, which will create an instance (job flow instance) of the parallel program definition
3. Creates a shortcut link (IRM_FLOW) to the resulting job flow diagram and assign a library reference (RESULTS) to the results library

Figure A1.3 *Job Flow Script*



Submit the job flow script for execution in SAS Studio session. Here is the sample script for the “hello world example:

```
/* Scripting Client Initialization - Required setup */
%irm_sc_init();

/* Add job flow -- Build jobflow container */
%irm_sc_build_jobflow(
        i_jf_name        =hello_world_jobflow,
        o_jf_ref_name    =example_jf_ref);

/* Add task -- sentence_task */
%sentence_task(
```

```
        i_jf_ref         =&example_jf_ref,
        i_task_name      =sentence_task,
        o_task_ref_name  =sentence_task_ref);

/* Add task -- hello_task */
%hello_task(
        i_jf_ref         =&example_jf_ref,
        i_task_name      =hello_task,
        o_task_ref_name  =hello_task_ref);

/* Add task -- world_task */
%world_task(
        i_jf_ref         =&example_jf_ref,
        i_task_name      =world_task,
        o_task_ref_name  =world_task_ref);

/* Save this job flow definition to IRM Server */
%irm_sc_save_jobflow(
        i_jf_ref         =&example_jf_ref);

/* Execute job flow instance in IRM Server */
%irm_sc_execute_jobflow(
        i_jf_ref         =&example_jf_ref);

/* Show results library and link to PFD IRM Server */
%irm_sc_show_jobflow_results(
        i_jf_ref         =&example_jf_ref);
```

View and Consume Results

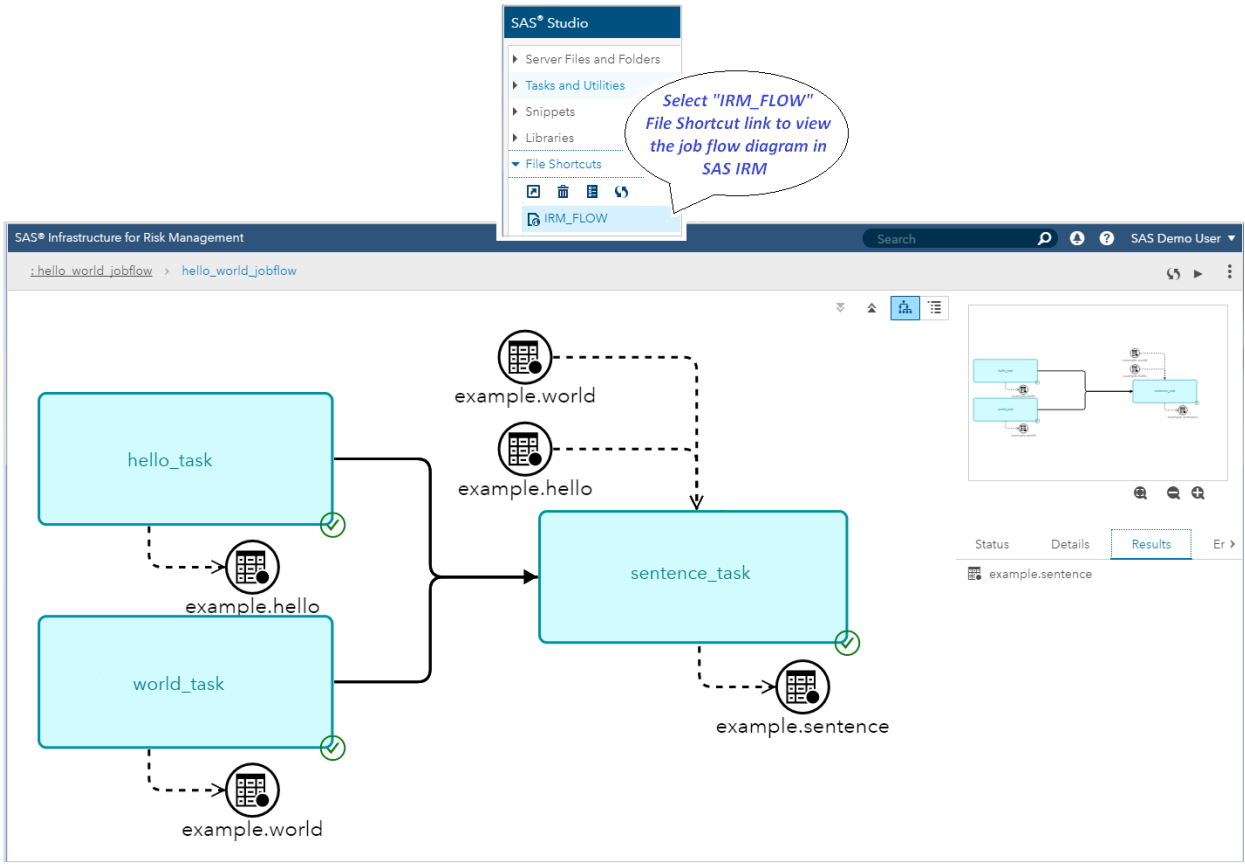
After successful execution of the hello_world_jobflow script, you can view the resulting job flow diagram (the graphical representation of the job flow) or access the job flow instance results tables in the RESULTS library.

View the Job Flow Diagram

Successful execution of the hello_world_jobflow script creates a File Shortcut link (IRM_FLOW) under **File Shortcuts** in the SAS Studio navigation pane

Double-click the IRM_FLOW file shortcut to open the job flow diagram in a SAS Infrastructure for Risk Management window. (See Figure A1.4 on page 51.) The link is only for the current job flow instance that is created in this SAS Studio session.

Figure A1.4 *Job Flow Shortcut*



View Results Library and Consume Results

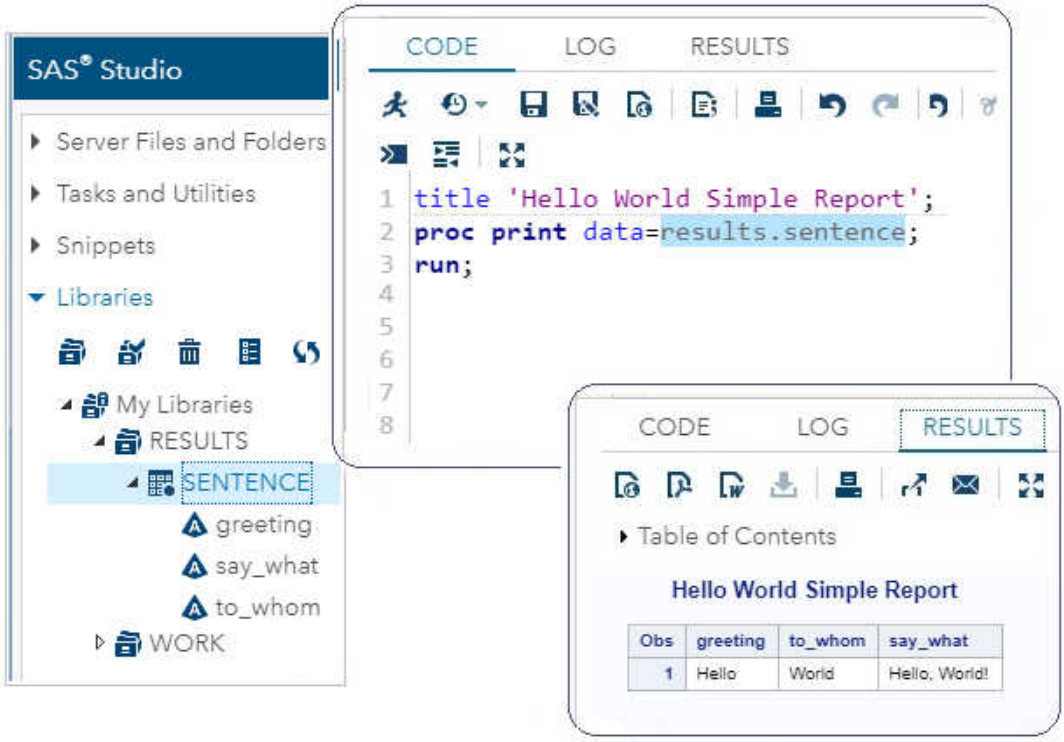
Successful execution the hello_world_jobflow script creates a Library (RESULTS) under **My Libraries** in the SAS Studio navigation pane.

This library contains the job flow instance terminal output tables for the current script execution. In the example, the single output file is sentence.sas7bdat.

The final task in the example is to produce a simple report using the results output. You simply reference the output table in the RESULTS library in your code. (See Figure A1.5 on page 52.):

```
proc print data = results.sentence;
run;
```

Figure A1.5 *Library Results*

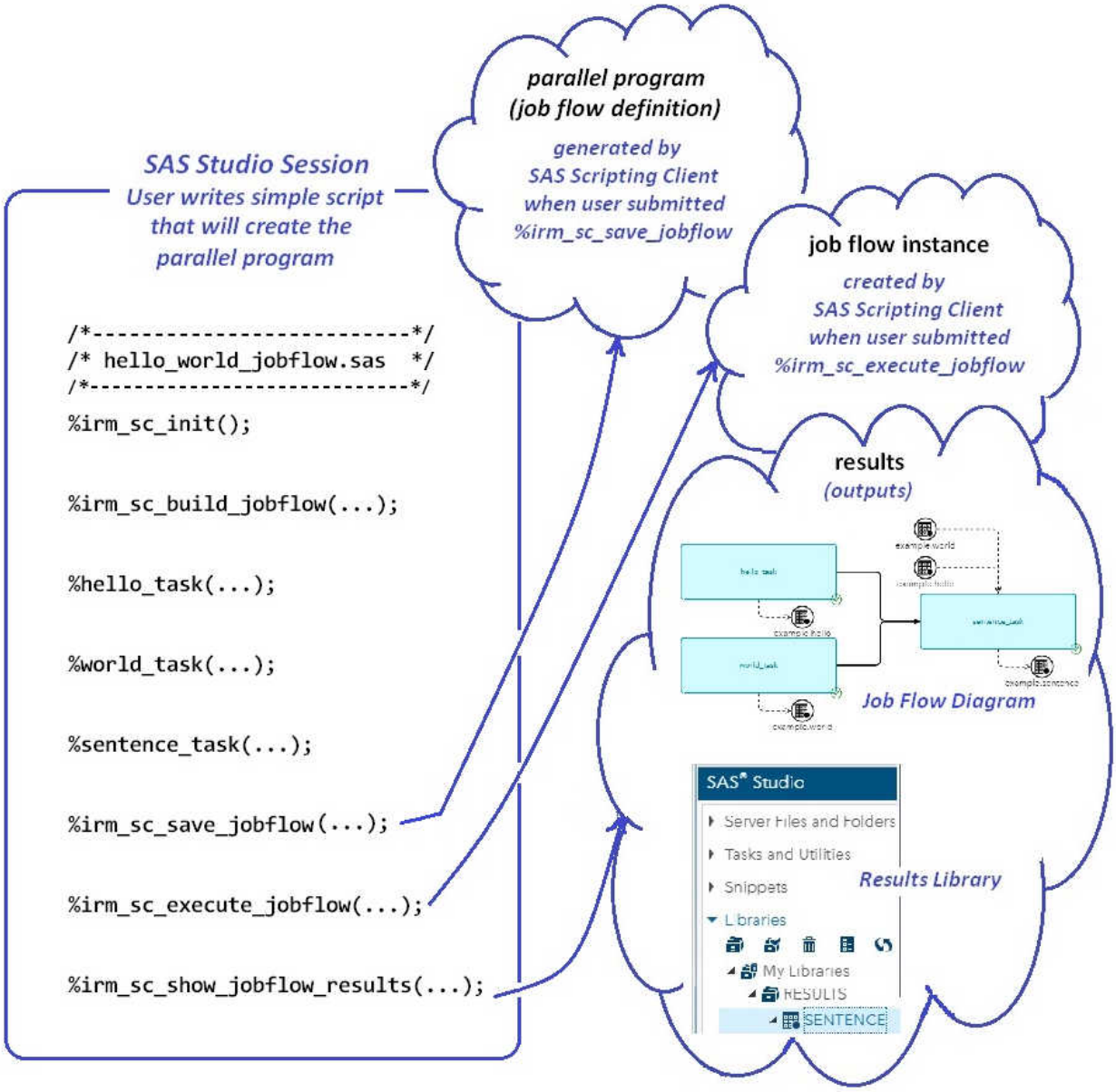


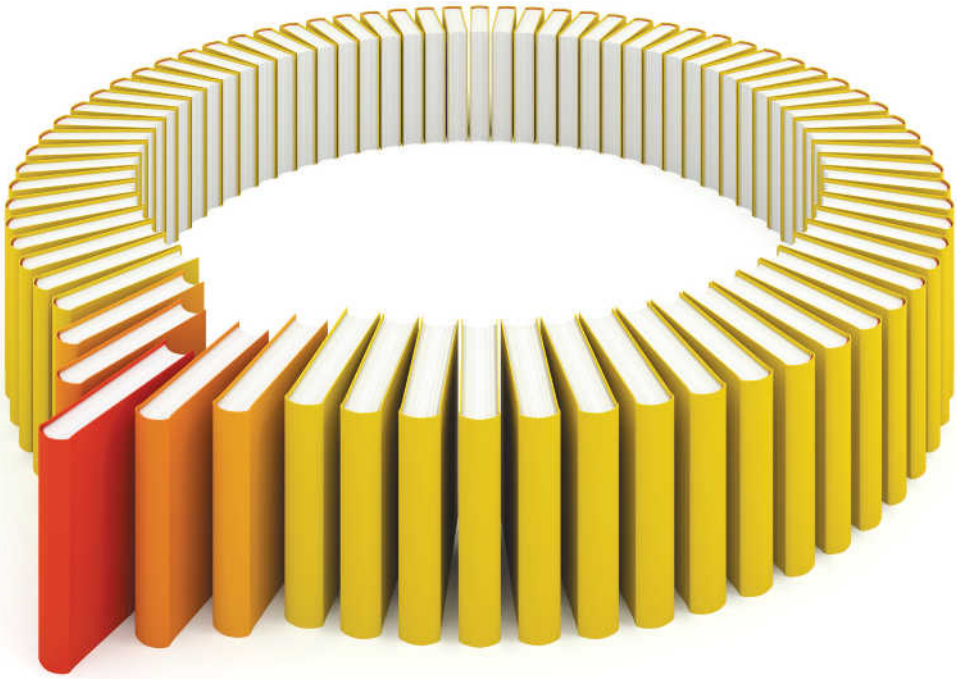
Wrap-Up

In this tutorial, you used the SAS Scripting Client to quickly and easily create a parallel program to be executed on the SAS Infrastructure for Risk Management platform. (SeeFigure A1.6 on page 53.) Here is the process:

1. Wrote a SAS Scripting Client job flow script to define the parallel program.
2. Executed the SAS Scripting Client job flow script, which performed these tasks::
   a. Generated the parallel program that defined the job flow of the three tasks that were specified to complete the job.
   b. Executed the generated parallel program (job flow definition) to create an instance of the job flow.
   c. Created a file shortcut to the job flow diagram and assigned a library reference to the RESULTS library.

Figure A1.6 *Wrap-up*





Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
for additional books and resources.

sas
THE POWER TO KNOW.

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. © 2013 SAS Institute Inc. All rights reserved. S107969US.0613