SAS® Infrastructure for Risk Management 3.6: Programmer’s Guide for SAS

SAS® Documentation
April 21, 2022

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2019. *SAS® Infrastructure for Risk Management 3.6: Programmer’s Guide for SAS*. Cary, NC: SAS Institute Inc.

SAS® Infrastructure for Risk Management 3.6: Programmer’s Guide for SAS

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

April 2022

3.6-P1:irmscug

Contents

About This Book

Audience

This book is intended for SAS programmers who want to learn how to easily create parallel programs that run on the SAS Infrastructure for Risk Management platform.

What’s New in SAS Infrastructure for Risk Management 3.6

New Features and Enhancements

SAS Infrastructure for Risk Management 3.6 provides the following new SAS programming features and enhancements:

- Support for automatic conversion of SAS data sets to and from Pandas DataFrames
- Ability to download SAS data sets in JSON (%PA and %LA)
- Automatic task execution when code is changed in a programmer’s personal federated area
- Support for the configuration of support for ticket-granting tickets (TGT) in SAS tasks
- Support to create partitions based on the number of cores
- Support for SAS views

Chapter 1

Before You Begin

About SAS Infrastructure for Risk Management

The SAS Infrastructure for Risk Management platform is a high-performance job execution engine that leverages the many-task computing model. SAS Infrastructure for Risk Management enables SAS programmers to quickly and easily develop highly parallel programs.

Use SAS Infrastructure for Risk Management to create SAS programs that function as follows:

- execute computationally intensive analytics as fast as possible
- can easily be developed with SAS Studio
- automate the handling of the complexity of creating parallel programming, task scheduling, and job distribution
- automatically generate documentation

Key Concepts

As a programmer who uses the SAS Infrastructure for Risk Management platform to create parallel programs, you must understand the following SAS Infrastructure for Risk Management key concepts and components:

Task

Tasks are pre-constructed analytical or reporting programs. Inputs and outputs are defined for these programs, and each program performs an independent unit of work. Tasks are the basic building blocks of SAS Infrastructure for Risk Management.

Job Flow (parallel program)

A *job flow*, or parallel program, is the method that SAS Infrastructure for Risk Management uses to organize the tasks that make up a job.

A *job flow definition* specifies the tasks and subflows that are contained in the job flow.

A *job flow instance* is a single occurrence of a job flow definition.

In this guide, the term, job flow, might be used to refer to both a job flow definition and a job flow instance. The exact meaning depends on the context in which it is used.

Subflow

A subflow is a job flow definition that is contained within another job flow definition.

Federated Area

A federated area is a folder structure that conforms to specific SAS Infrastructure for Risk Management rules. It is a storage area for content and it is the mechanism by which programmers add custom content to SAS Infrastructure for Risk Management. SAS Infrastructure for Risk Management provides a platform federated area and a sample federated area. In addition, programmers have their own personal federated areas, which are writable only to them. A personal federated area is the location in which a single programmer develops custom content.

Content

Content refers to any file that can be delivered in SAS Infrastructure for Risk Management, but is not necessarily part of the SAS Infrastructure for Risk Management installation. Examples of content include data, code, and configuration information. Because programmers deploy content for SAS Infrastructure for Risk Management in a federated area, federated areas and content are synonymous in SAS Infrastructure for Risk Management.

Data Objects

In SAS Infrastructure for Risk Management, all data is defined as a data object. A data object can be data input to one or more tasks, but it can be the output data for only one task. Data objects are immutable. A data object can model any data that can be contained in a file.

Scripting Client

The SAS Infrastructure for Risk Management scripting client is a collection of SAS macros that simplify the process of creating job flows (parallel programs).

Typical Development Scenario

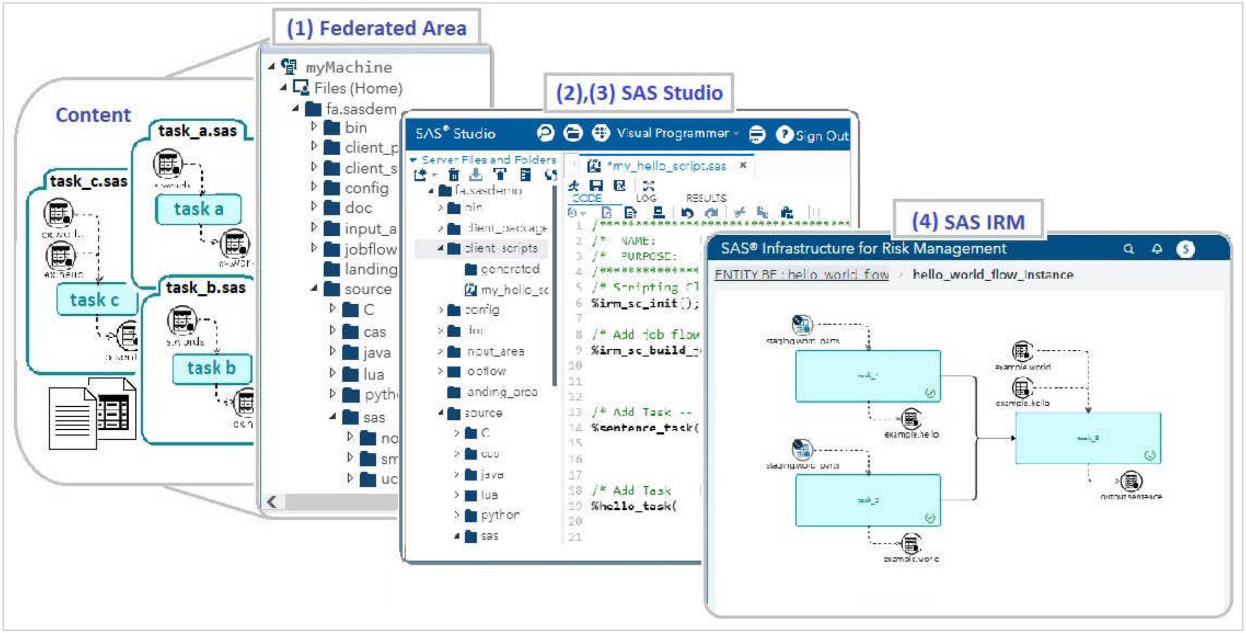
Here is a typical development scenario for a SAS Infrastructure for Risk Management programmer:

1. Create your content (task files and related content) and save to your personal federated area.
2. In SAS Studio, develop a program (job flow script) using scripting client and task macros to create a job flow.
3. In SAS Studio, also run the job flow script.

When you run the job flow script, SAS Infrastructure for Risk Management automatically handles the complexity of creating the parallel program, scheduling the tasks, and job distribution.

4. View the job flow instance in the SAS Infrastructure for Risk Management web application.

Here is a visualization of the development scenario that was previously described:



```
/************************************
/*  NAME:
/*  PURPOSE:
/**************
/* Scripting Cl
%irm_sc_init();

/* Add job flow
%irm_sc_build_j



/* Add Task --
%sentence_task(



/* Add Task
%hello_task(
```

Set Up Your Development Environment

Before you can develop parallel programs in SAS Infrastructure for Risk Management or follow the development examples that are included in this guide, you must complete the following steps:

1. Ensure that you have access to a SAS Infrastructure for Risk Management 3.6 server, the SAS Infrastructure for Risk Management web application, and SAS Studio 3.7.
2. Consult with your system administrator about meeting the following requirements:
   - Enable development mode on the SAS Infrastructure for Risk Management server.
   - Create a developer's account for you.

     The configuration of your programmer's account enables files and folders that you create to be discovered by stored process servers and SAS workspace servers.

   For information about these administrative tasks, see "Configure the Development Environment" in *SAS Infrastructure for Risk Management: Administrator's Guide*.
3. Create your personal federated area.
   - Using your programmer's account, log on to the SAS Infrastructure for Risk Management web application.

When you log on to the SAS Infrastructure for Risk Management web application for the first time, your personal federated area is automatically created in the following location:

`/sas-configuration-directory/Lev1/AppData/SASIRM/pa/fas`

By default, the personal federated area is named after your user name. For example, if your user name is “user1,” the name of your personal federated area is as follows:

`/sas-configuration-directory/Lev1/AppData/SASIRM/pa/fas/fa.user1`

- After you create your personal federated area, contact your system administrator to obtain Write permissions to all folders in your personal federated area.

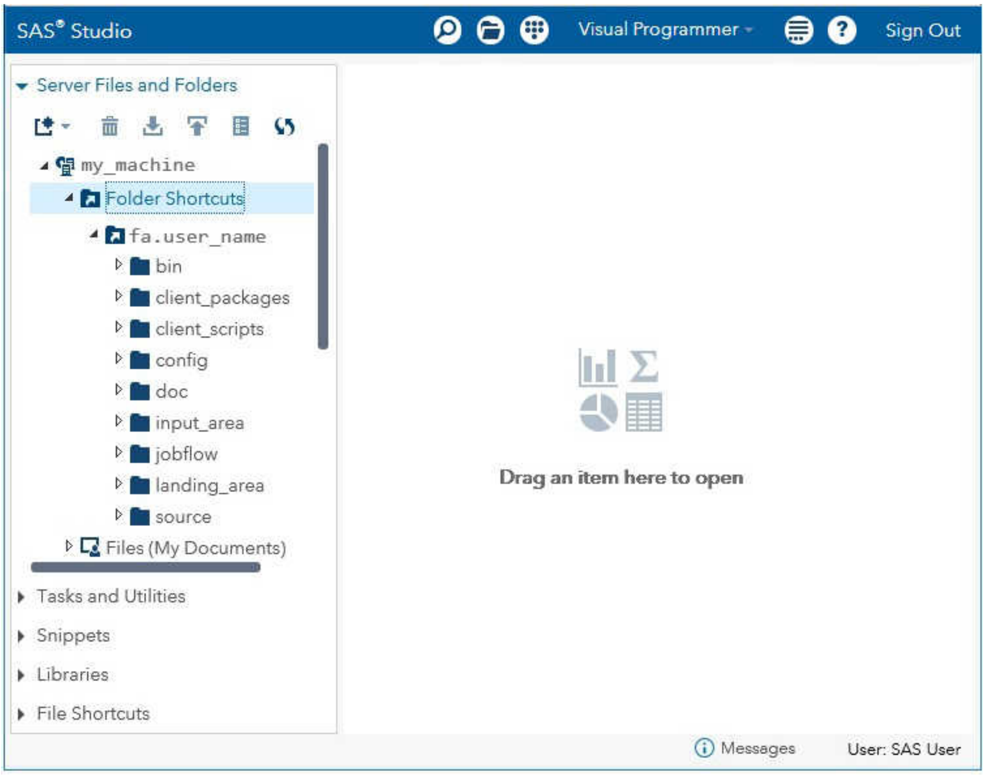
For more information about your personal federated area, see “Your Personal Federated Area ”.

4. Set up your SAS Studio interface.

Using your programmer’s account:

- Log on to SAS Studio.
- Create a folder shortcut to your personal federated area on the SAS Infrastructure for Risk Management server.

Here is an example of how your screen should look:



Note: You can also create folder shortcuts to federated areas other than your own. However, access to these federated areas must be Read-Only. You must not modify any federated area other than your personal federated area. The only exceptions to this rule 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.

Chapter 2
Create Your First Parallel Program in 60 Seconds

About Creating Your Parallel Program Example

Note: Before you can complete the parallel program example, your development environment must be correctly set up. (See "Set Up Your Development Environment".)

SAS Infrastructure for Risk Management provides a scripting client that enables programmers to quickly develop job flows (parallel programs).

Here are the steps to create a parallel program:

1. Log on to SAS Studio.
2. Develop a program (job flow script) using scripting client and task macros in order to create a job flow.
3. Run the program.

To follow the example, note the following tips:

- You use tasks that are available to you in the SAS Infrastructure for Risk Management sample federated area (fa sample.3.6). Therefore, the sample federated area must be installed on the SAS Infrastructure for Risk Management server. The tasks are named sentence, hello, and world.
- When you develop a job flow script, you can enter the script manually. However, an easier way to create your job flow script is to use a copy of the script template (script_template.sas). The script template is located in the `client_packages/sas` folder in your personal federated area.

  Note: The script template is generated when you initialize the scripting client by running the %irm_sc_init macro. Therefore, if you do not see the script template, run the %irm_sc_init macro to generate it.

Creating Your First Parallel Program

High-Level View of Creating the Job Flow

1. In SAS Studio, click to open a new SAS program.
2. Type (or copy and paste) the following job flow script (program), which consists of scripting client macros and tasks.

```
/******************************************************************************/
/*  NAME:      hello_world_60second_jobflow.sas                              */
/*  PURPOSE:   Create hello_world_jobflow in 60 seconds using sample tasks   */
/******************************************************************************/

/*Step 1  Scripting Client Initialization - Required setup */
%irm_sc_init();

/*Step 2  Create a job flow - Name the jobflow definition */
%irm_sc_build_jobflow(
          i_jf_name      =hello_world_flow,
          o_jf_ref_name  =example_jf_ref);

/*Step 3  Add Tasks from fa.sample.3.6 - sentence, hello, world */
%sentence(
          i_jf_ref       =&example_jf_ref,
          i_task_name    =sentence);

%hello(
          i_jf_ref       =&example_jf_ref,
          i_task_name    =hello);

%world(
          i_jf_ref       =&example_jf_ref,
          i_task_name    =world);

/*Step 4  Save this job flow definition to IRM Server */
%irm_sc_save_jobflow(
          i_jf_ref       =&example_jf_ref);

/*Step 5  Execute job flow instance in IRM Server */
%irm_sc_execute_jobflow(
          i_jf_ref       =&example_jf_ref);

/*Step 6  Show RESULTS library and link to Job Flow Diagram in IRM Server*/
%irm_sc_show_jobflow_results(
          i_jf_ref       =&example_jf_ref);
```

3. Click to run your job flow script.

Detailed View of Creating the Job Flow

Here is a detailed description of each step that is highlighted in the job flow script that you just created.

1. Initialize the scripting client.

   ```
   %irm_sc_init();
   ```

   The %irm_sc_init macro is the required first step of a job flow script. It initializes the scripting client, and it defines the scripting client macros and the configuration parameters to the SAS Infrastructure for Risk Management server.

   Note: This example uses default values for the job flow configuration parameters (for example, the configuration set ID and base date). However, in a typical development scenario, you will likely override the default configuration parameters and enable verbose debug messages.

2. Define the job flow definition.

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

   The %irm_sc_build_jobflow macro creates the job flow definition.

   In this example, the job flow definition is named *hello_world_flow* and the assigned job flow reference name is *example_jf_ref*.

   The job flow reference name is a required parameter. You use this name to refer to the job flow definition in subsequent steps of the job flow script. When you use a reference name, always precede the reference name with an ampersand (&).

   Note: One advantage of using a reference name in the job flow script is that it enables you to reuse the job flow script with minimal code changes. For example, you can reuse a job flow script to create a new job flow definition that has a different name by updating the name once in the %irm_sc_build_job macro instead of having to change it for every reference to the job flow definition in the job flow script.

3. Add tasks to the job flow definition.

   Add tasks to the *hello_world_flow* job flow definition (which was assigned a job flow reference name example_jf_ref in Step 2) by calling the task macros.

   In this example, the following three tasks are added to the hello_world_flow job flow definition using default values for the task inputs and outputs:

   ```
   %sentence(
               i_jf_ref          =&example_jf_ref,
               i_task_name       =sentence);
   %hello(
               i_jf_ref          =&example_jf_ref,
               i_task_name       =hello);
   %world(
               i_jf_ref          =&example_jf_ref,
               i_task_name       =world);
   ```

   You can add the tasks in any order to the job flow definition. SAS Infrastructure for Risk Management automatically deduces data dependencies from the tasks and schedules the parallel execution of the tasks.

Note: The three tasks (sentence, hello and world) are available in the SAS Infrastructure for Risk Management sample federated area (fa.sample.3.6). Their corresponding task macros were automatically created when you executed the %irm_sc_init macro in Step 1.

4. Save the job flow definition.

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

The %irm_sc_save_jobflow macro saves the job flow definition to the SAS Infrastructure for Risk Management server.

5. Create and execute the job flow definition.

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

The %irm_sc_execute_jobflow macro creates and executes a job flow instance. In addition to creating and executing the job flow instance, SAS Infrastructure for Risk Management generates a job flow diagram that you can view in the SAS Infrastructure for Risk Management web application.

6. View the results of your first parallel program.

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

The %irm_sc_show_jobflow_results macro creates the RESULTS library in the **Libraries** section in the SAS Studio navigation pane. The RESULTS library provides a quick way to view the results of executing job flow script.

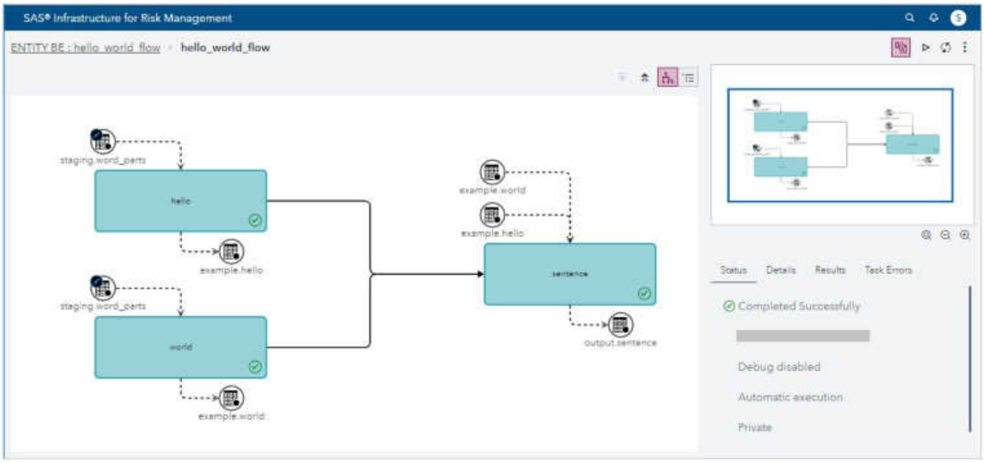
The macro also creates a file shortcut (IRM_FLOW) to the job flow instance in the **File Shortcuts** section in the SAS Studio navigation pane. The IRM_FLOW shortcut provides a quick way to view the job flow instance in the SAS Infrastructure for Risk Management web application.

Note: The RESULTS library and IRM_FLOW file shortcut apply only to the SAS Studio session in which you submitted the job flow script to create the instance.

Viewing Your First Parallel Program

To view the job flow instance that you created, select the IRM_FLOW file shortcut in the **File Shortcuts** section in the same SAS Studio session in which you just submitted the job flow script. Alternatively, you can view the job flow instance from the SAS Infrastructure for Risk Management web application.

Here is how your first parallel program appears in the SAS Infrastructure for Risk Management web application:



Completed Successfully

Debug disabled

Automatic execution

Private

Chapter 3
Federated Areas

About Federated Areas

A federated area is a folder structure that conforms to specific SAS Infrastructure for Risk Management rules. It is a storage area for content. As a SAS Infrastructure for Risk Management programmer, you must organize the content that you create in a federated area.

Federated areas enable content developers to deploy content independently of the SAS Infrastructure for Risk Management platform.

In addition, federated areas provide reproducibility: the ability to run existing content repeatedly. Deploying new content or a new platform should never break existing content.

The content of a federated area consists of the following elements:

- job flow definition — a file that contains the tasks and subflows required to complete a job flow
- code — task files (nodes), string message data, and macros
- input files — 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

When you install SAS Infrastructure for Risk Management 3.6, the following federated areas are installed:

- `fa.0.3.6` — contains only elements that are required to make the platform run. There is no content in the platform federated area.
- `fa.sample.3.6` — contains SAS sample content that you can use to test the SAS Infrastructure for Risk Management installation, create SAS parallel programs, and use as a reference.
- `fa.sample.3.6.py` — contains Python sample content that you can use to test the SAS Infrastructure for Risk Management installation, create Python parallel programs, and use as a reference.
- **fa.*user_name*** — a personal federated area that is created on demand. You must have a personal federated area in order to develop your own content.

It is important to note the following rules about federated areas:

- Are independent, but they must be designed to not conflict with other federated areas.
- They must not be altered, modified, or deleted after they are deployed.

  The exceptions to this rule are:

  - You can upload data to the input area or the landing area of a federated area other than the platform federated area.
  - You can modify your personal federated area.
- Tasks, job flows, and input data are *federated content*, which can be shared across all federated areas. Therefore, do not change the definition of tasks, job flows and data. Doing so can cause unpredictable results.

  All other content is local to a federated area and cannot be shared with other federated areas.
- As a SAS Infrastructure for Risk Management programmer (content developer), you can modify your personal federated area. Typically, you use the scripting client to make modifications. The SAS Infrastructure for Risk Management 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.

  Note: Before you delete the contents of your personal federated area, you must delete any job flows that you created using the content from your personal federated area.

Your Personal Federated Area

Your personal federated area is where you develop your content.

It is important to note the following concepts about your personal federated area:

- SAS Infrastructure for Risk Management can read all personal federated areas. However, you cannot see other programmer's personal federated areas — just your own.
- Your personal federated area is at the top of the federated area precedence chain. It is always the first federated area that is searched by SAS Infrastructure for Risk Management for federated content, unless you select the federated area below your personal federated area (time machine) when you create a job flow instance in the SAS Infrastructure for Risk Management user interface. For more information about

creating a job flow instance based on a federated area that is not at the top of the inheritance lineage, see "Create an Instance" in *SAS Infrastructure for Risk Management: User's Guide*.

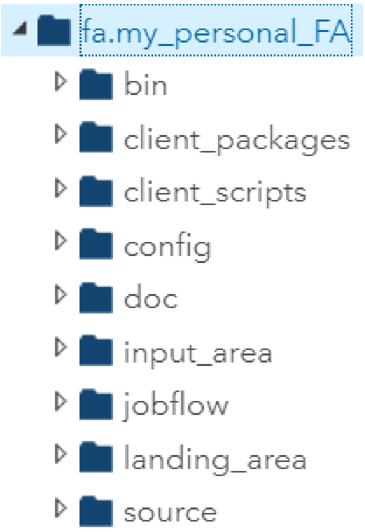
- You cannot publish any job flow instances that you create from your personal federated area.

For information about generating your personal federated area, see "Set Up Your Development Environment".

Folders in a Federated Area

All federated areas have the same basic folder structure. Custom federated areas might contain additional folders or fewer folders than the ones described in this section. SAS Infrastructure for Risk Management does not require all folders to be included in a federated area. However, specific types of content must be in the appropriate folder.

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



Here is a partial list of the folders that are in your personal federated area:

- `client_scripts` — contains the job flow scripts that you have created.
- `config` — contains files that a programmer uses to configure the behavior of job flows. Specifically, this folder contains the following files and folders:
  - `messages` — contains the labels to the nodes, job flows, and the inputs and outputs that are visible in the SAS Infrastructure for Risk Management web application.
  - `job_flow_definitions.csv` — lists the job flows that are available in the SAS Infrastructure for Risk Management web application.
  - `libnames.txt` — maps libraries for the input data that you use in your job flows.
  - `macrovarload.txt` — lists the SAS data sets that define the global macro variables that must be loaded before a task executes.

- **doc** — contains the task documentation that is generated when you invoke the scripting client %irm_sc_gen_doc macro.
- **input_area** — is the area in which you can load data directly into a federated area when the SAS Infrastructure for Risk Management server is running. For information about loading data, see “Loading Data into Your Personal Federated Area Using Live ETL”.
- **jobflow** — contains job flow definitions or subdirectories that contain job flow definitions. Subfolders within the job flow folder are displayed as categories in the SAS Infrastructure for Risk Management web application. Only a single level of folder can contain a subflow folder. A job flow subfolder can contain a subflow folder.
- **landing_area** — is the read-only data mart of a federated area. The landing area contains the data objects (for example, SAS data sets) that are required for the job flows that are defined in that federated area. The SAS Infrastructure for Risk Management server must be shut down before you load data into the landing area unless you use Live ETL to upload data. For information about loading data, see “Loading Data into Your Personal Federated Area Using Live ETL”.
- **source** — contains the source code for your content by language (for example, C, CAS, Java, Lua, Python, and SAS).

  Each task type folder contains a nodes folder that contains the task file (for example, **source/sas/nodes**). Uncompiled SAS macros for a federated area must be stored in **source/sas/ucmacros**.

  The folders that appear in a language-specific folder are dependent on the language. However, each language-specific folder contains a **nodes** folder that contains the task files. For example, **source/sas/nodes** contains the .sas task files that are directly called by job flows. To make code more manageable, you can create a single-level hierarchy of subfolders in the nodes folder.

Base Date Folder Entity Table

At least one federated area in a SAS Infrastructure for Risk Management deployment must contain a **base date** folder in its landing_area. The base date folder contains input data for a period of time. The naming convention for a base date folder is MMDDYYYY (for example. 03312019).

Each base date folder must contain an *entity table*. SAS Infrastructure for Risk Management uses entity tables to configure aspects of the SAS Infrastructure for Risk Management web application. For example, the options that are available on the drop-down menus are determined by the configuration of the entity table.

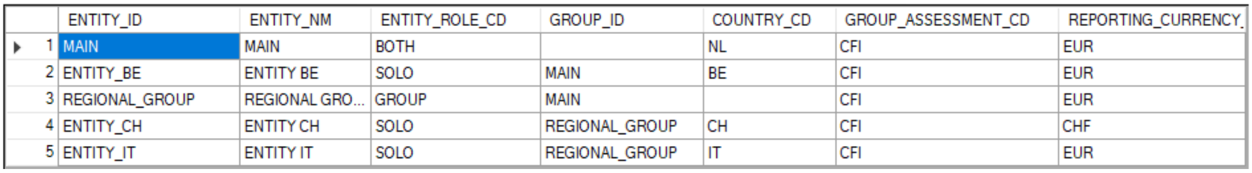
Typically, the data in the entity table is static. However, data can be updated via Loading Data into Your Personal Federated Area Using Live ETL.

CAUTION:

> **Entities that are being used cannot be deleted.** Deletion of an entity row being used would render existing job flow instances invalid.

Note: Entity tables in the highest federated areas take precedence over the entity tables in lower federated areas.

Here is an example of the entity table:



	ENTITY_ID	ENTITY_NM	ENTITY_ROLE_CD	GROUP_ID	COUNTRY_CD	GROUP_ASSESSMENT_CD	REPORTING_CURRENCY.
▸ 1	MAIN	MAIN	BOTH		NL	CFI	EUR
2	ENTITY_BE	ENTITY BE	SOLO	MAIN	BE	CFI	EUR
3	REGIONAL_GROUP	REGIONAL GRO...	GROUP	MAIN		CFI	EUR
4	ENTITY_CH	ENTITY CH	SOLO	REGIONAL_GROUP	CH	CFI	CHF
5	ENTITY_IT	ENTITY IT	SOLO	REGIONAL_GROUP	IT	CFI	EUR

where:

- ENTITY_ID — (Required) alphanumeric identifier that specifies the organizational or operational unit of an organization. A value that contains spaces or any special characters except for an underscore (_) is rejected when SAS Infrastructure for Risk Management is started.
- ENTITY_NM — specifies the name of the organizational or operational unit of an organization. If a value is not specified for this attribute, the value for the ENTITY_ID displays in the SAS Infrastructure for Risk Management web application.
- ENTITY_ROLE_CD — (Required) specifies which calculation level options are available in an entity. Here are the possible values:
  - BOTH — configures SAS Infrastructure for Risk Management to enable calculations at the solo and group level.
  - GROUP — configures SAS Infrastructure for Risk Management to enable calculations at a group level, which includes the subsidiary units within an entity.
  - SOLO — configures SAS Infrastructure for Risk Management to enable calculations for the chosen entity as a single unit.
- GROUP_ID — (Required) specifies the identification of the group to which a particular entity belongs. Note that a solo entity can belong to a group (ENTITY_BE) and also a group can belong to another group (REGIONAL_GROUP).

Note: The entity table can contain any number of variables, but the ENTITY_ID, GROUP_ID, and ENTITY_ROLE_CD table attributes are required.

Loading Data into Your Personal Federated Area Using Live ETL

Typically, SAS Infrastructure for Risk Management reads initial input data objects from the landing_area folder of a federated area. Because SAS Infrastructure for Risk Management can simultaneously run multiple job flows and tasks that might access the input data objects from the landing area, you should avoid manually loading your data to the landing area while the SAS Infrastructure for Risk Management server is running. Loading data to the landing_area folder of a federated area while the SAS Infrastructure for Risk Management server is running can cause errors (for example, file locking issues or incorrect results).

To avoid potential issues and errors, load your input data to the input_area folder of your federated area. Then SAS Infrastructure for Risk Management automatically uses live ETL to copy the input data objects from the input_area folder to the landing_area folder.

When using live ETL to upload data, note that you can upload all existing data using live ETL. There is no restriction on the number of tables that you can upload at the same time. In addition, you can create entities, configuration sets, and new base dates via live

ETL. Live ETL supports the creation of new input data objects. However, it does not support deleting input data objects.

To load input data when the SAS Infrastructure for Risk Management server is running, follow these steps:

1. Load your input data objects to the input_area folder of your personal federated area.
2. Modify the marker file, which is named last_update.txt and is located in the input_area folder, to update the file’s timestamp. This triggers live ETL, and SAS Infrastructure for Risk Management copies the input data from the input_area folder to the landing_area folder when it determines that input data objects are not being accessed by any job flow tasks.

Here is a brief SAS program that loads a SAS data set (MYDATA.mywords) to the landing area of a personal federated area. You execute this sample program in SAS Studio.

```
/**************************************************************************/
/* NAME:     LoadMyDataViaLiveETL.sas                                     */
/* PURPOSE:  Create MYWORDS dataset and load to landing area via Live ETL */
/* NOTE:     Replace ~pfa with path to my personal FA                     */

/**************************************************************************/
/*Step 1 - Put my dataset in my personal FA's input_area */
libname ina "~pfa/input_area";
data ina.mywords;
   attrib  WORD length= $50
           PART length= $32;
   word="HELLO"    ;  part="interjection" ;  output;
   word="WORLD"    ;  part="noun"         ;  output;
run;

/*Step 2 - Create the library entry in libnames.txt */
data _null_;
    FILE "~pfa/config/libnames.txt";
    PUT 'MYDATA=%la';
run;

/*Step 3 - Touch last_update.txt to trigger live ETL and refresh the libnames */
data _null_;
    FILE "~pfa/input_area/last_update.txt";
    PUT;
run;
```

The following steps provide details about each action in the preceding sample program.

1 Copies the input data set to the input_area folder of the personal federated area. SAS Infrastructure for Risk Management copies the input data set to the input_area folder of the personal federated area by first assigning a libref (INA) to the input_area folder. Then, the data set (named mywords) is created in the INA library.

In this example, the program dynamically creates the data set. However, a typical use case would be a process to copy or extract the data to this location.

2 Adds an entry for the new landing area library to the libnames.txt file.

SAS Infrastructure for Risk Management owns all library assignments. These library assignments must be defined in the libnames.txt file in the config folder of a federated area so that SAS Infrastructure for Risk Management knows that the library exists.

In this example, the libref MYDATA is created and refers to the landing_area (%la) folder of the personal federated area.

Note: For more information about assigning library mappings in the libnames.txt file and configuration variable notation (for example, %la for landing area), see "Map Libraries" in *SAS Infrastructure for Risk Management: Administrator's Guide*.

3 Modifies the last_update.txt file in the input_area folder.

Modifying the last_update.txt file in the input area folder triggers SAS Infrastructure for Risk Management to invoke live ETL. The live ETL process copies the input data objects from the input_area folder to the landing_area folder.

After you execute the sample program, you should see the following output data and files in SAS Studio:

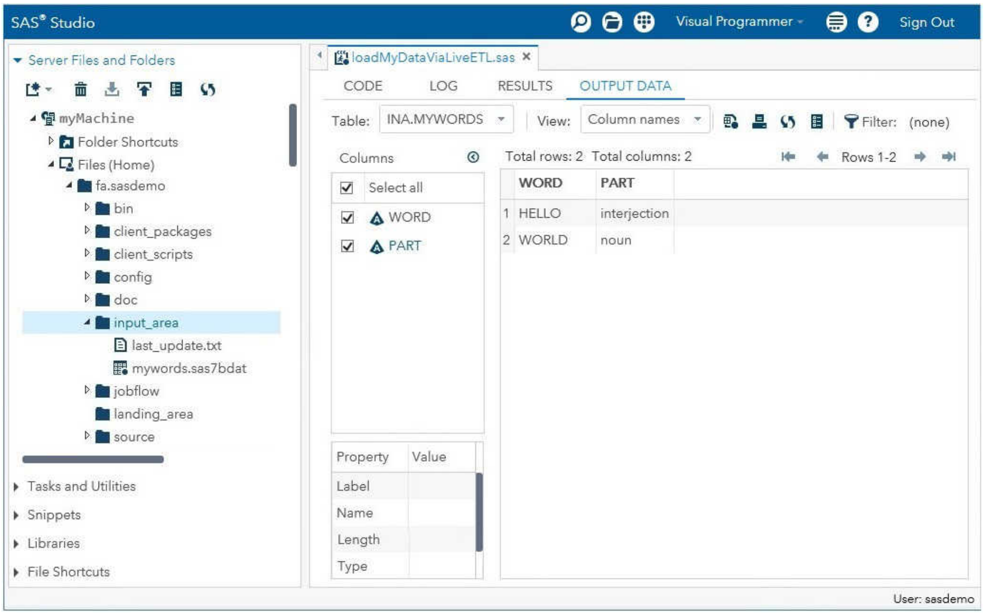


Table: INA.MYWORDS View: Column names Filter: (none)

Total rows: 2 Total columns: 2 Rows 1-2

	WORD	PART
1	HELLO	interjection
2	WORLD	noun

After a short delay (up to 30 seconds), the landing area is populated with your data:

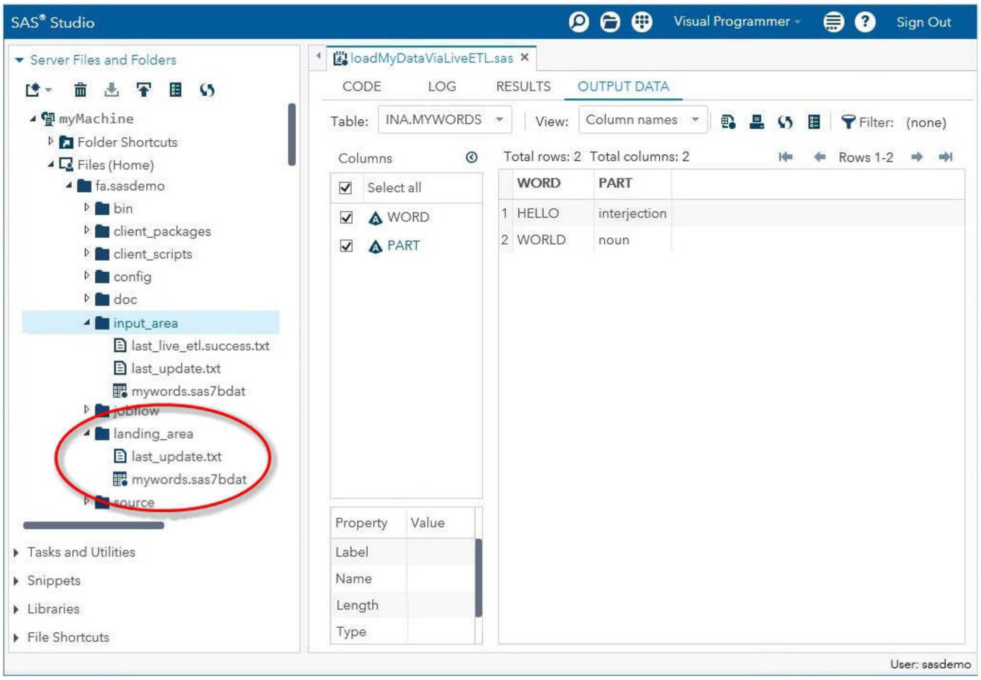


Table: INA.MYWORDS View: Column names Filter: (none)

Total rows: 2 Total columns: 2

	WORD	PART
1	HELLO	interjection
2	WORLD	noun

Note: SAS Infrastructure for Risk Management adds an additional file (last_live_etl_success.txt) to the input_area folder to notify you of the success or failure of the live ETL process.

Loading SAS Macros into Your Personal Federated Area

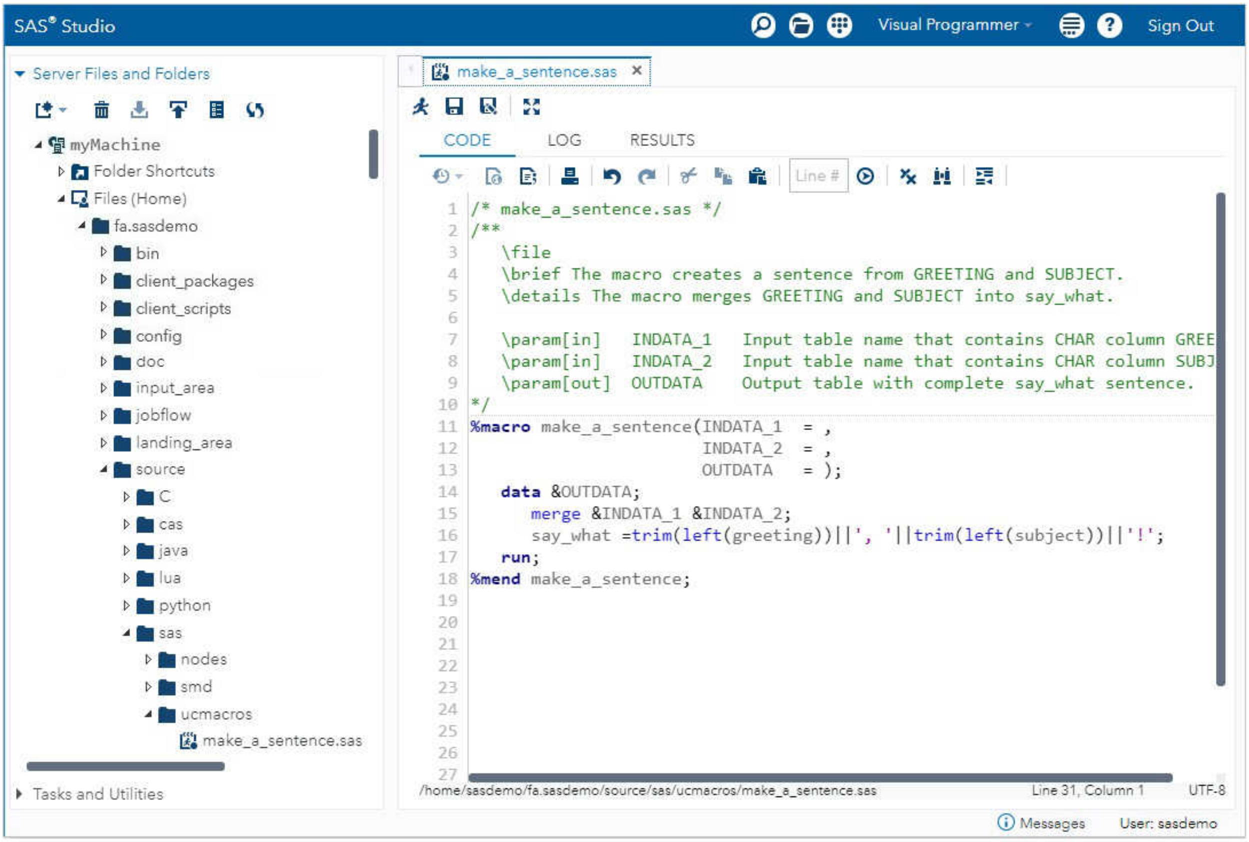
You might choose to create SAS task programs that call uncompiled SAS macros (.sas files). If you choose this programming approach, you must load the uncompiled SAS macros in the ucmacros folder in your personal federated area.

Here is a sample SAS macro program that creates a SAS macro that is named make_a_sentence.sas. This macro is called by the Example 3 task. (See "Example 3: Task with Parameter Substitution and a SAS Macro Call".)

```
/* make_a_sentence.sas */
/**
   \file
   \brief The macro creates a sentence from GREETING and SUBJECT.
   \details The macro merges GREETING and SUBJECT into say_what.

   \param[in]   INDATA_1   Input table name that contains CHAR column GREETING
   \param[in]   INDATA_2   Input table name that contains CHAR column SUBJECT
   \param[out]  OUTDATA    Output table with complete say_what sentence.
*/
%macro make_a_sentence(INDATA_1  = ,
                       INDATA_2  = ,
                       OUTDATA   = );
   data &OUTDATA;
      merge &INDATA_1 &INDATA_2;
      say_what =trim(left(greeting))||', '||trim(left(subject))||'!';
   run;
%mend make_a_sentence;
```

Note: The SAS macro program file is not a SAS task file that is stored in the **`source/sas/nodes`** folder. Therefore, you do not have to include a file header in the program file. However, it is a recommended practice to include a file header that documents the function of the macro and identifies the inputs and outputs.

After creating the SAS macro program file (make_a_sentence.sas), save it to the ucmacros folder that is located in **`fa.`*`username`*`/source/sas/ucmacros`**:



```
/* make_a_sentence.sas */
/**
   \file
   \brief The macro creates a sentence from GREETING and SUBJECT.
   \details The macro merges GREETING and SUBJECT into say_what.

   \param[in]   INDATA_1   Input table name that contains CHAR column GREE
   \param[in]   INDATA_2   Input table name that contains CHAR column SUBJ
   \param[out]  OUTDATA    Output table with complete say_what sentence.
*/
%macro make_a_sentence(INDATA_1  = ,
                       INDATA_2  = ,
                       OUTDATA   = );
   data &OUTDATA;
      merge &INDATA_1 &INDATA_2;
      say_what =trim(left(greeting))||', '||trim(left(subject))||'!';
   run;
%mend make_a_sentence;
```

/home/sasdemo/fa.sasdemo/source/sas/ucmacros/make_a_sentence.sas

Sharing the Content in Your Personal Federated Area

After developing and testing your content in your personal federated area, you might want to make your content available to others.

As a SAS Infrastructure for Risk Management programmer, here is the process for sharing the content that you developed in your personal federated area:

1. Develop and test your content in your personal federated area.
2. From the top folder of your personal federated area, compress the federated area with any percentage of compression to create a ZIP file of your personal federated area.
3. Ask your system administrator to install your zipped personal federated area as a stand-alone federated area on the SAS Infrastructure for Risk Management server.

For more information about installing a stand-alone federated area, see "Install a Stand-Alone Federated Area without a Server Restart" in *SAS Infrastructure for Risk Management: Administrator's Guide*.

Chapter 4
Tasks

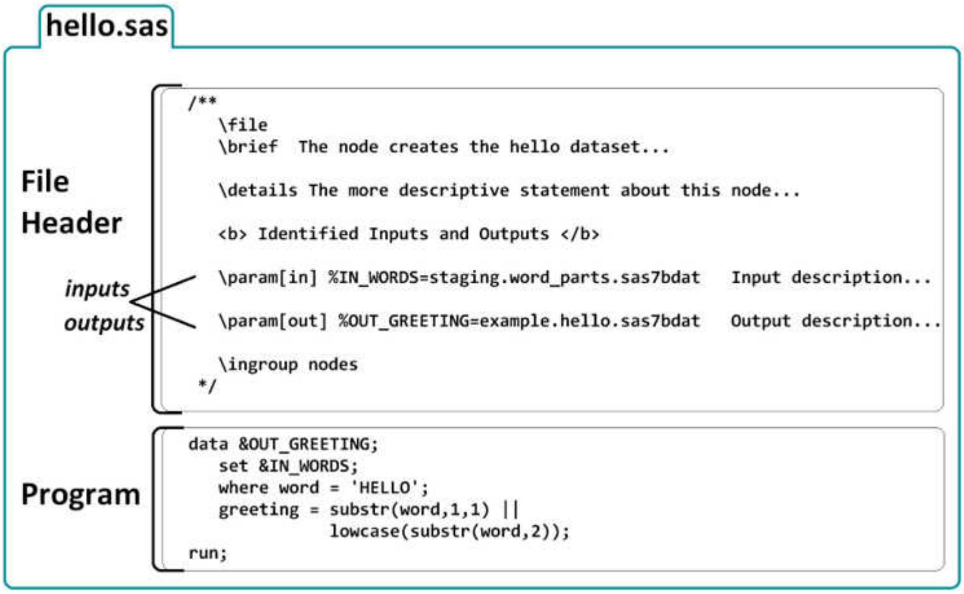
About Tasks

The basic building block of a SAS Infrastructure for Risk Management job flow is the *task*. Tasks are independent programs that you combine in a job flow to complete a job.

A task is contained in only one file (task file). The task file consists of two required parts:

- file header — a syntax-specific header section in which you must declare the task-specific inputs, outputs, or both inputs and outputs. It is also a good practice to document the function of the task in the header. The file header is used by SAS Infrastructure for Risk Management to identify your program as a task and create task packages that are available for you to use when developing job flows.
- program — the analytical or reporting element of processing that is packaged as a complete program. This is your code that accomplishes a complete unit of work.

For example, here are the file header and program parts of the hello.sas task file that is located in the SAS Infrastructure for Risk Management sample federated area (fa.sample.3.6):



```
/**
   \file
   \brief  The node creates the hello dataset...

   \details The more descriptive statement about this node...

   <b> Identified Inputs and Outputs </b>

   \param[in] %IN_WORDS=staging.word_parts.sas7bdat   Input description...

   \param[out] %OUT_GREETING=example.hello.sas7bdat   Output description...

   \ingroup nodes
 */
```

```
data &OUT_GREETING;
   set &IN_WORDS;
   where word = 'HELLO';
   greeting = substr(word,1,1) ||
              lowcase(substr(word,2));
run;
```

Before you begin to use or create tasks, note the following:

- Tasks must declare all inputs and outputs.
- You can combine individual tasks in a job flow.
- When you use multiple tasks in your job flow, SAS Infrastructure for Risk Management determines the order and parallelism in which the tasks are executed based on the availability of the inputs and the number of cores that are available for processing.
- You can use predefined tasks (for example, tasks that are delivered in a federated area) in your job flow. In addition, you can use tasks that you create in your own personal federated area.
- The task file name must be in lowercase letters and not exceed 32-characters (excluding the .sas file extension). The name must follow the standard SAS naming conventions. This name, excluding the file extension, is the name of the task as it appears in the SAS Infrastructure for Risk Management web application.

Anatomy of the Task File Header

SAS Infrastructure for Risk Management requires that all tasks contain a properly constructed header that conforms to Doxygen syntax guidelines. (See http://doxygen.nl/.) At a minimum, the file header must contain the task's inputs and outputs. The task header is a type of comment header that SAS Infrastructure for Risk Management uses to create tasks. It is not a comment header that is ignored by SAS Infrastructure for Risk Management.

Documenting the Task

In the file header, you can create user documentation that describes the function of a task. When you process the content that you are developing, this documentation is parsed and made available in HTML format for users to access. You must document a task for tooltips and node-level help.

Here is an example:

```
\file
\brief    Include a brief description of the task.
\details  Include detailed information about the task, what it does.
```

Defining Task Inputs and Outputs

The inputs and outputs for a task are treated as input and output function parameters. Therefore, in each task file, you must document inputs and outputs using the following Doxygen syntax:

```
\param[in|out] data-set-name data-set-description
```

where:

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

Here is an example:

```
\param[in]  example1.dataset.sas7bdat  example input data set
\param[out] example2.dataset.sas7bdat  example output data set
```

If you specify specific input data or output data (as shown in this example), the task is inflexible because the task must always use the data input or data output that you specified. If you want to reuse the task for different inputs or outputs, use parameter substitution instead.

Defining Task Inputs and Outputs Using Parameter Substitution

Parameter (or token) substitution enables you to reuse existing tasks for different jobs. When you specify a parameter substitution for input data or output data, the code that is associated with the task references the input data or output data by the parameter. The convention is the same for all tasks, regardless of the programming language that you use.

To use parameter substitution, specify the parameter name as follows:

```
\param[dir] %parameter-name data-set-description
```

where the percent sign (%) denotes parameter substitution.

Here is an example:

```
\param[in]  %IN_WORDS   example input data set or data state
\param[out] %OUT_HELLO  example output data set or data state
```

Defining Task Inputs and Outputs Using Parameter Substitution with Default Value Specification

If there is a default substitution value, you assign the value using the equal sign (=) and the corresponding parameter substitution value.

For example, if you have substitution parameters named IN_WORDS and OUT_HELLO and you assign their default values to STAGING.word_parts.sas7bdat and EXAMPLE.hello.sas7bdat, you use the following syntax:

```
\param[in]  %IN_WORDS=STAGING.word_parts.sas7bdat  example input data set
\param[out] %OUT_HELLO=EXAMPLE.hello.sas7bdat      example output data set
```

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

Defining Task Inputs and Outputs Using Collections

To denote collection data (such as a data library, a file folder, or a format catalog) as input data or output data for a task, follow the SAS LIBNAME naming rules.

For example, assume that you have a substitution token VALIDATION_COLLECTION, and you want to assign the token to a folder, you use the following syntax:

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

Note that the collection value (*validtn*) follows the SAS LIBNAME naming rule and is not a multi-part named value.

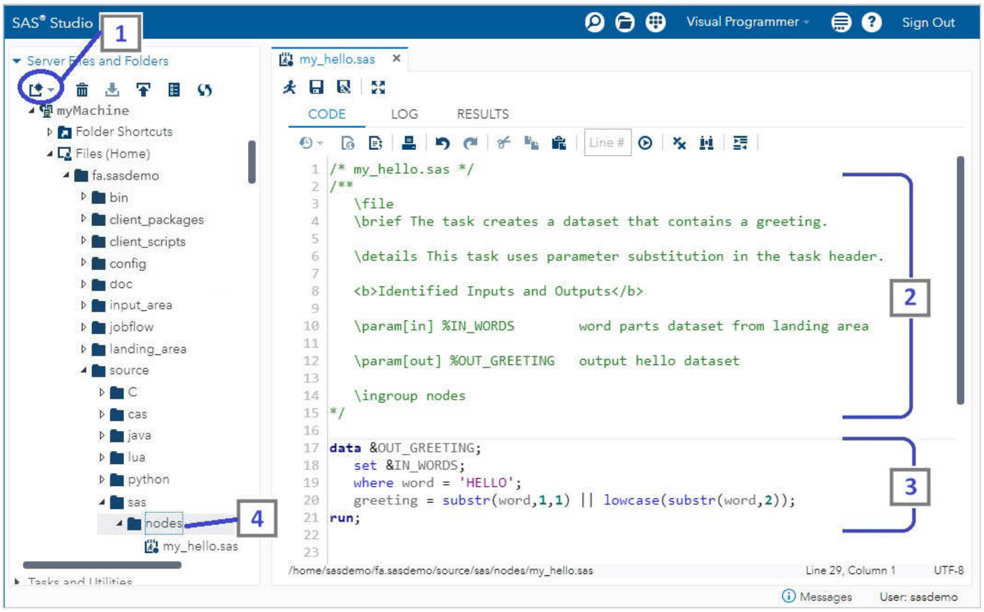
Creating Tasks

About Creating Tasks

1. In your SAS Studio session, open a new SAS program.
2. Enter the task file header. You might need to edit the header after you create the program and identify all your inputs and outputs.
3. Enter the task program. Make sure that you have tested and verified your program code.

   Note: For illustration purposes, the program in the following example has been entered as open SAS code. Typically, your program code is not open code, but instead is a macro call to macros that you have created in your personal federated area.
4. Save the task file (using a valid name) to the nodes folder in your personal federated area. The task file name must conform to the standard SAS naming conventions.

Here is an illustration of the four steps:



```
/* my_hello.sas */
/**
   \file
   \brief The task creates a dataset that contains a greeting.

   \details This task uses parameter substitution in the task header.

   <b>Identified Inputs and Outputs</b>

   \param[in] %IN_WORDS        word parts dataset from landing area

   \param[out] %OUT_GREETING   output hello dataset

   \ingroup nodes
*/

data &OUT_GREETING;
   set &IN_WORDS;
   where word = 'HELLO';
   greeting = substr(word,1,1) || lowcase(substr(word,2));
run;
```

/home/sasdemo/fa.sasdemo/source/sas/nodes/my_hello.sas Line 29, Column 1 UTF-8

Here are three SAS task files that you will use later in this guide to create a job flow. You can copy the code samples for the three example tasks in order to create your tasks. These are tasks that you will create and save in your personal federated area.

If you choose to follow the examples in this guide, ensure that the task file names that you use and the task file names in the examples are identical. Here are the task file names:

- my_hello.sas
- my_world.sas
- my_sentence.sas

As a benefit, you can copy and paste the code samples without having to edit the code.

Ensure that the following requirements have been met:

- The my_words data set has been loaded to the landing area. (See "Loading Data into Your Personal Federated Area Using Live ETL".)
- The make_a_sentence macro has been created in the ucmacros folder of your personal federated area. (See "Loading SAS Macros into Your Personal Federated Area".)

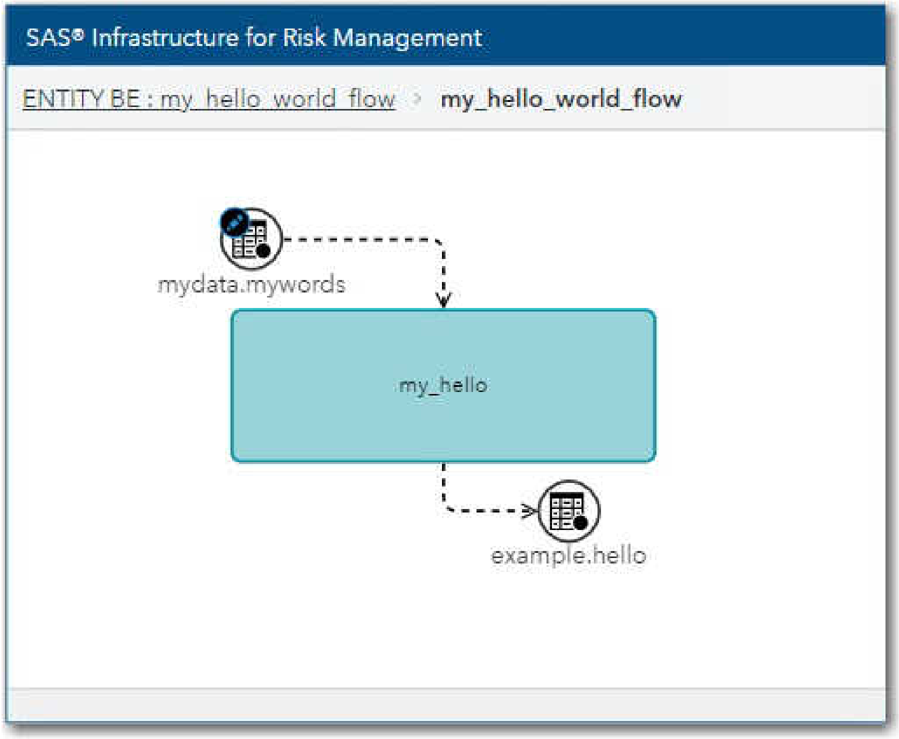
Example 1: Task with Parameter Substitution

This task uses parameter substitution to identify its input and output data sets. (See "Defining Task Inputs and Outputs Using Parameter Substitution".)

Here is a code sample that you can use to create a SAS task that is named my_hello.

```
/* my_hello.sas */
/**
   \file
   \brief The task creates a dataset that contains a greeting.

   \details This task uses parameter substitution in the task header.

   <b>Identified Inputs and Outputs</b>
   \param[in] %IN_WORDS       word parts dataset from landing area

   \param[out] %OUT_GREETING  output hello dataset

   \ingroup nodes
*/

data &OUT_GREETING;
   set &IN_WORDS;
   where word = 'HELLO';
   greeting = substr(word,1,1) || lowcase(substr(word,2));
run;
```

Here is how the newly created task appears in the SAS Infrastructure for Risk Management web application when you use it in a job flow.


SAS® Infrastructure for Risk Management
ENTITY BE : my_hello_world_flow
my_hello_world_flow
mydata.mywords
my_hello
example.hello

Example 2: Task with Parameter Substitution and Default Values

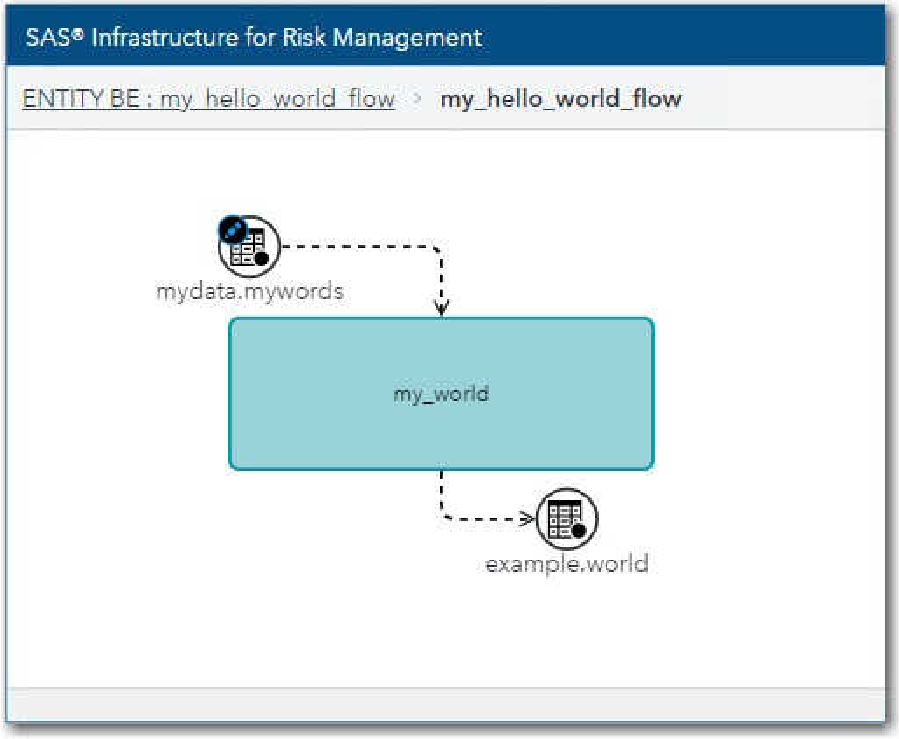
This task uses parameter substitution with default value specifications to identify its input and output data sets. (See “Defining Task Inputs and Outputs Using Parameter Substitution with Default Value Specification”.)

Note: When defining substitution with default values in the header, ensure that there are no spaces or tabs before or after the equal sign (=).

Here is a code sample that creates a SAS task that is named my_world.

```
/* my_world.sas */
/**
   \file
   \brief The task creates a dataset that contains a subject for a sentence.
   \details The task creates a dataset that contains a subject for a sentence.
            Task uses parameter substitution with default value specification
            in the task header. Note there is no spaces before or after the =.

   <b>Identified Inputs and Outputs</b>

   \param[in] %IN_WORDS=mydata.mywords.sas7bdat      input from landing area

   \param[out] %OUT_SUBJECT=example.world.sas7bdat   output subject dataset

   \ingroup nodes
*/

data &OUT_SUBJECT;
   set &IN_WORDS;
   where word = 'WORLD';
   subject = lowcase(word);
run;
```

Here is how the task appears in the SAS Infrastructure for Risk Management web application when you use it in a job flow.


SAS® Infrastructure for Risk Management
ENTITY BE : my_hello_world_flow
›
my_hello_world_flow
mydata.mywords
my_world
example.world

Example 3: Task with Parameter Substitution and a SAS Macro Call

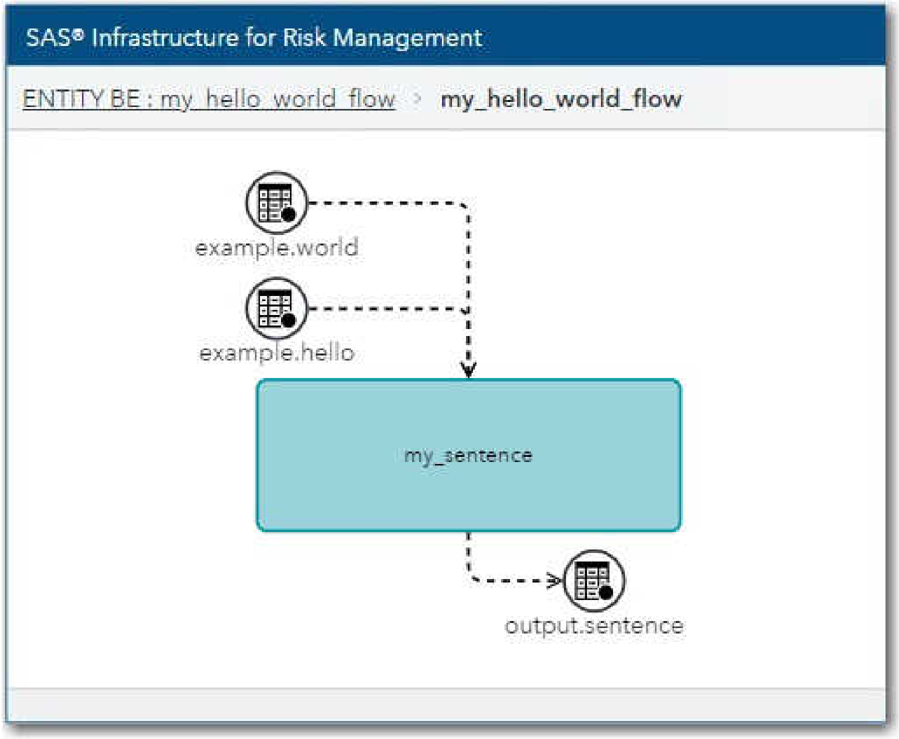
In this task, a SAS macro is used instead of open code in the program section of the task file.

For information about how to create the SAS macro that is used in this example, see "Loading SAS Macros into Your Personal Federated Area".

Here is a code sample that creates a SAS task that is named my_sentence.

```
/* my_sentence.sas */
/**
   \file
   \brief    The task creates the sentence dataset, calls a macro.
   \details  The task merges the greeting and the subject datasets and creates
             an output dataset the contains the final sentence.

   <b>Identified Inputs</b>
   \param[in] %IN_GREETING=example.hello.sas7bdat    input hello dataset
   \param[in] %IN_SUBJECT=example.world.sas7bdat     input world dataset

   <b>Identified Outputs</b>
   \param[out] %OUT_SENTENCE=output.sentence.sas7bdat  output sentence dataset

   \ingroup nodes
*/

%make_a_sentence(INDATA_1 =&IN_GREETING ,
                 INDATA_2 =&IN_SUBJECT  ,
                 OUTDATA  =&OUT_SENTENCE);
```

Here is how the task appears in the SAS Infrastructure for Risk Management web application when you use it in a job flow.



Chapter 5
Job Flows

About Job Flows

A job flow, or parallel program, is a way to organize tasks. One can use the term job flow to mean a job flow definition or a job flow instance; the exact meaning depends on the context in which it is used. (See “Key Concepts”.)

In SAS Infrastructure for Risk Management, to create a job flow definition:

1. Define your job flow and assign it a name.
2. Add tasks to your job flow.

   These SAS Infrastructure for Risk Management tasks declare inputs and outputs and are stored in the language-specific nodes folder of a federated area.

Note: The order in which you add tasks to a job flow is unimportant. SAS Infrastructure for Risk Management automatically deduces data dependencies from the tasks, schedules the parallel execution of the tasks, and draws a job flow diagram for you.

Scripting Client Overview

To simplify the process of creating job flows, SAS Infrastructure for Risk Management provides a *scripting client*. The scripting client is a collection of SAS macros that you combine into a single program (job flow script), which you then save and execute to create your job flow.

Note: Before you can use the scripting client, you must ensure that your development environment has been set up correctly. (See "Set Up Your Development Environment".)

Some important characteristics to note about the scripting client macros include:

- Two types of SAS macros are used by the scripting client: system-reserved macros, which start with prefix %irm_sc, and the task macro, which is derived from the user-defined task node.
- The scripting client macro syntax uses the following parameter prefixes:
  - `i_` identifies an input
  - `o_` identifies an output
  - `t_` identifies parameter (token) substitution. If you want to override the default value, this convention requires that you provide a parameter (token) substitution value.

Here are the most commonly used scripting client macros:

Table 5.1 *Common Scripting Client Macros*

Macro	Description
%irm_sc_init	Initializes the scripting client and creates your task API in the client packages folder in your personal federated area.
%irm_sc_build_jobflow	Names and builds the job flow definition.
%irm_sc_save_jobflow	Saves the job flow definition to the jobflow folder in your personal federated area.
%irm_sc_execute_jobflow	Executes the job flow definition and creates an instance of the job flow.

Note: For a complete list of the scripting client macros, see Appendix 2, "Scripting Client Macro Reference".

Example 1: Creating a Simple Job Flow with Your Three Tasks

Here is an example of how to create a simple job flow that contains the three tasks that you created in Chapter 4, "Tasks".

High-Level View of Creating the Job Flow

1. In SAS Studio, click to open a new SAS program.

2. Enter your job flow script (program), which consists of scripting client macros and your tasks:

```
/*****************************************************************/
/*  NAME:      my_hello_world_jobflow.sas                        */
/*  PURPOSE:   SAS Scripting Client program to create my job flow */
/*****************************************************************/
/*Step 1  Scripting Clien Initialization - Required setup */
%irm_sc_init();

/*Step 2  Name the job flow and create a jobflow definition */
%irm_sc_build_jobflow(
          i_jf_name        =my_hello_world_flow,
          o_jf_ref_name    =example_jf_ref );

/*Step 3 Add Tasks - my_sentence, my_hello, my_world */
%my_hello(
          i_jf_ref         =&example_jf_ref,
          i_task_name      =my_hello,
          t_in_words       =mydata.mywords.sas7bdat,
          t_out_greeting   =example.hello.sas7bdat );

%my_world(
          i_jf_ref        =&example_jf_ref,
          i_task_name     =my_world );

%my_sentence(
          i_jf_ref         =&example_jf_ref,
          i_task_name      =my_sentence );

/*Step 4  Save this job flow definition to IRM Server */
%irm_sc_save_jobflow(
          i_jf_ref         =&example_jf_ref );

/*Step 5  Execute job flow instance in IRM Server */
%irm_sc_execute_jobflow(
          i_jf_ref         =&example_jf_ref );
```

3. Click to run your job flow script.

Detailed View of Creating the Job Flow

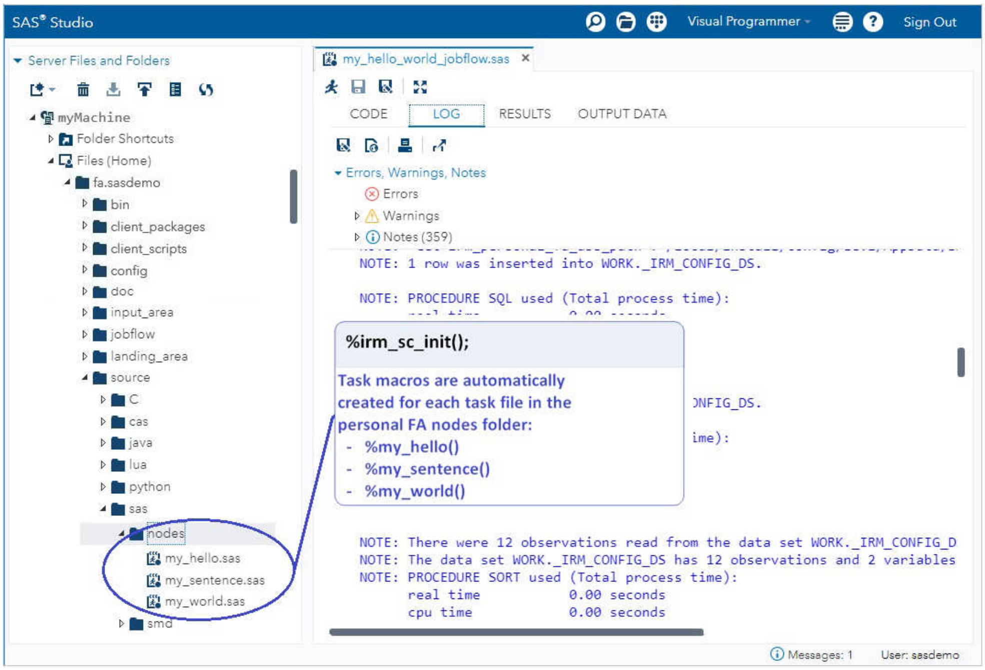
Here is a detailed description of each step in the job flow script that you just created. The examples do not display all the options or arguments that are available for a given scripting client macro. Default settings are used.

1. Initialize the scripting client.

```
%irm_sc_init();
```

The %irm_sc_init macro initializes the scripting client. It also reads the task files in your personal federated area nodes folder and creates task macros with the same name as those task files, but without the .sas extension.

In this example, the %irm_sc_init macro creates three task macros (%my_hello, %my_sentence, and %my_world) shown as follows:



```
NOTE: 1 row was inserted into WORK._IRM_CONFIG_DS.

NOTE: PROCEDURE SQL used (Total process time):
```

%irm_sc_init();

Task macros are automatically created for each task file in the personal FA nodes folder:

- %my_hello()
- %my_sentence()
- %my_world()

```
NOTE: There were 12 observations read from the data set WORK._IRM_CONFIG_D
NOTE: The data set WORK._IRM_CONFIG_DS has 12 observations and 2 variables
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

2. Name and build the job flow definition.

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

You use the %irm_sc_build_jobflow macro to name the job flow definition that you are creating as well as assign it a reference. You use the reference that you assign to the job flow definition that you will use to refer to the job flow definition in subsequent steps of your scripting client program. Always precede a reference name with an ampersand (&) when you use the reference name.

In this example, the job flow definition name is my_hello_world_flow and the reference to this job flow definition is example_jf_ref.

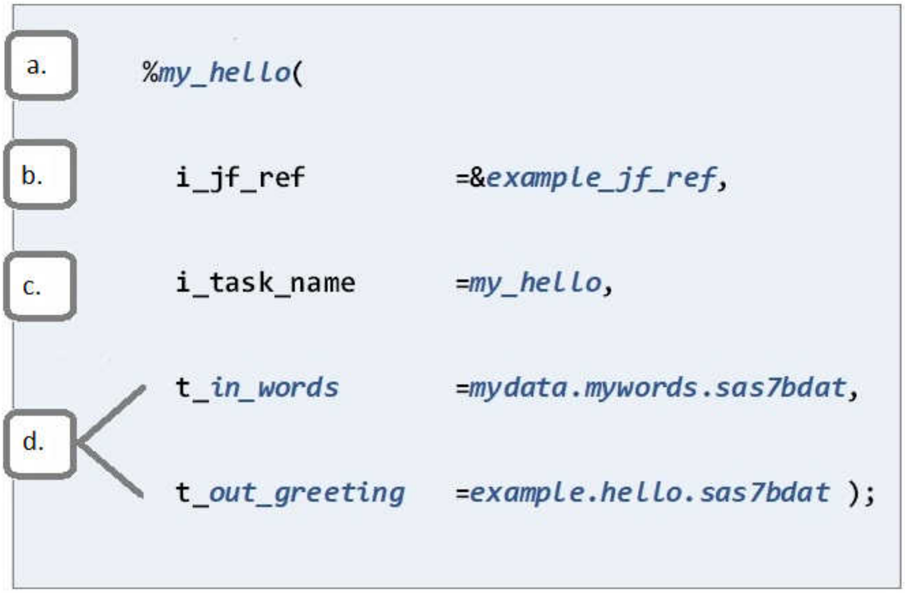
3. Add the tasks to the job flow definition.

```
%my_hello(
          i_jf_ref          =&example_jf_ref,
          i_task_name       =my_hello,
          t_in_words        =mydata.mywords.sas7bdat,
          t_out_greeting    =example.hello.sas7bdat );

%my_world(
          i_jf_ref          =&example_jf_ref,
          i_task_name       =my_world  );

%my_sentence(
          i_jf_ref          =&example_jf_ref,
          i_task_name       =my_sentence );
```

You can add your task macros to the job flow script in any order.

The syntax of each task follows the same basic structure, but it might vary in how the inputs and outputs are named based on the task header.

For example, here is an explanation of the syntax that is used for the my_hello task macro in this example job flow script:



```
%my_hello(
   i_jf_ref          =&example_jf_ref,
   i_task_name       =my_hello,
   t_in_words        =mydata.mywords.sas7bdat,
   t_out_greeting    =example.hello.sas7bdat );
```

a. Invoke the task macro (%task_name).

   The task macro is created when you execute %irm_sc_init macro and is the same name as the task program file without the file extension.

b. Specify the job flow definition to which you are adding the task.

   Use the job flow reference name that you specified for the %irm_sc_build_jobflow macro, preceded by an ampersand (&).

c. (Optional) Specify a user-assigned task name.

   It might be useful to assign a descriptive task name to identify tasks when you are using the same task multiple times with different inputs. If you do not specify a descriptive task name, the task name defaults to the name of the task macro.

d. (Optional) Specify the task inputs and outputs based on what is defined in the task file header.

   There are several methods for specifying the inputs and outputs in the task file header. These different specifications determine the syntax that you use in the job flow script for the task macro:

***Table 5.2** Data Input and Output Specifications*

Task File Header Specification	Job Flow Script Configuration
Parameter substitution without a default table name specified: **`\param[in] %IN_WORDS`**	You must specify the name in the script: **`t_IN_WORDS=mydata.mywords.sas7bdat`** *Note:* The **`t_`** prefix is required syntax to indicate that parameter (token) substitution is used, followed by the parameter name (**`IN_WORDS`**) as defined in the task header.
Parameter substitution, with a default table name: **`\param[in] %IN_WORDS=mydata.mywords.sas7bdat`**	You can include the table name in the script. To accept the default table name, do not specify the parameter. An example of this syntax is shown in the Example 1 j ob flow script where the my_world and my_sentence tasks were added to the job flow. To override the default table name, specify the parameter with the name of the override table: **`t_IN_WORDS =mydata.override.sas7bdat`**
No parameter substitution used: **`\param[in] mydata.mywords.sas7bdat`**	The table name is fixed in the task header. Therefore, it cannot be changed or substituted in the job flow script.

4. Save the job flow definition to the SAS Infrastructure for Risk Management server.

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

   The %irm_sc_save_jobflow macro saves the my_hello_world_flow job flow definition to the SAS Infrastructure for Risk Management server. Use the job flow reference name (&example_jf_ref) that you assigned to the job flow in the %irm_sc_build_jobflow macro when you save the job flow definition. Ensure that you precede the job flow reference name with an ampersand (&).

   Once saved, the job flow definition can be used to create job flow instances in SAS Infrastructure for Risk Management.

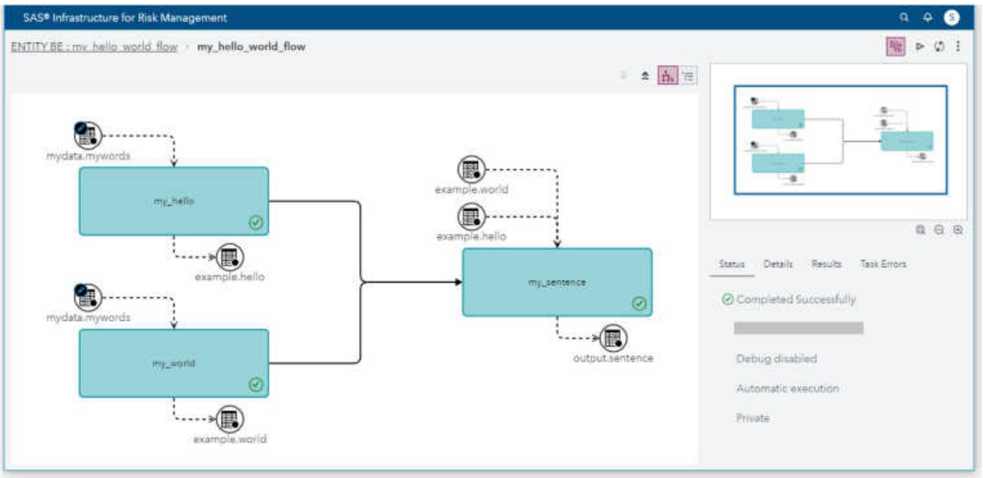
5. Create and execute a job flow instance.

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

   The %irm_sc_execute_jobflow macro creates and executes an instance of the job flow that is referenced in the macro. Use the job flow reference name (&example_jf_ref) that you assigned to the job flow in the %irm_sc_build_jobflow macro when you specify the job flow definition to be executed. Precede the job flow reference name with an ampersand (&).

View the Job Flow Instance

After creating the job flow instance, you can view it from the SAS Infrastructure for Risk Management web application. Here is how the job flow instance appears in the SAS Infrastructure for Risk Management web application.



Example 2: Creating a Job Flow Containing a Subflow

Here is an example of a job flow script that creates a job flow definition that performs the same function as the simple job flow in Example 1. However, in this job flow definition, the my_hello task and the my_world task are part of a subflow that is named sentence_subflow.

High-Level View of Creating the Job Flow

1. In SAS Studio, click to open a new SAS program.

2. Enter your job flow script (program), which consists of scripting client macros and your tasks. The program is the same as in Example 1, but with the addition of a subflow and different job flow name.

   In this job flow script, the subflow (sentence_subflow) is created and added to the top-level flow (my_hello_world_sub_and_flow).

```
/*******************************************************************/
/*  NAME:      my_hello_world_jobflow_with_subflow.sas            */
/*******************************************************************/
/*Step 1  Scripting Client Initialization - Required setup */
%irm_sc_init();

/*Step 2  Create the top-level job flow - create job flow definition */
%irm_sc_build_jobflow(
          i_jf_name      =my_hello_world_sub_and_flow,
          o_jf_ref_name  =example_jf_ref );

/*Step 3  Create the subflow named sentence_subflow */
%irm_sc_build_jobflow(
          i_jf_name      =sentence_subflow,
          o_jf_ref_name  =example_subflow_ref );

/*Step 4  Add the subflow to the top-level job flow */
%irm_sc_add_subflow(
          i_jf_ref       =&example_jf_ref,
          i_sub_jf_ref   =&example_subflow_ref );

/*Step 5  Add tasks (my_hello, my_world) to the subflow */
%my_hello(
          i_jf_ref       =&example_subflow_ref,
          i_task_name    =my_hello,
          t_in_words     =mydata.mywords.sas7bdat,
          t_out_greeting =example.hello.sas7bdat );

%my_world(
          i_jf_ref       =&example_subflow_ref,
          i_task_name    =my_world );

/*Step 6  Add task (my_sentence) to the top-level job flow */
%my_sentence(
          i_jf_ref       =&example_jf_ref,
          i_task_name    =my_sentence );

/*Step 7  Save this job flow definition to IRM Server */
%irm_sc_save_jobflow(
          i_jf_ref       =&example_jf_ref );

/*Step 8  Execute job flow instance in IRM Server */
%irm_sc_execute_jobflow(
          i_jf_ref       =&example_jf_ref );
```

3. Click 🏃 to run your job flow script.

Detailed View of Creating the Job Flow

Here is a detailed description of steps 3, 4, and 5 in the job flow script that you just created. The remainder of the script is the same as that which is explained in the simple job flow script in Example 1.

1. (Step 3) Create a subflow that is named sentence_subflow and assign it a reference.

```
%irm_sc_build_jobflow(
         i_jf_name     =sentence_subflow
         o_jf_ref_name =example_subflow_ref );
```

   Use the %irm_sc_build_jobflow macro to name the subflow job flow definition (sentence_subflow) and to assign it a reference name (example_subflow_ref).

2. (Step 4) Add the subflow to the top-level job flow.

```
%irm_sc_add_subflow(
          i_jf_name     =&example_jf_ref,
          i_sub_jf_ref  =&example_subflow_ref );
```

   The top-level job flow (named hello_world_sub_and_flow with the reference example_jf_ref) was created in step 2 and the subflow (named sentence_subflow with reference example_subflow_ref) was created in step 3.

   Use the %irm_sc_add_subflow macro to add the subflow to the top-level flow. This step designates sentence_subflow as a subflow (a job flow contained within a job flow) of hello_world_sub_and_flow.

3. (Step 5) Add tasks to the subflow.

```
%my_hello(
          i_jf_ref       =&example_subflow_ref,
          i_task_name    =my_hello,
          t_in_words     =mydata.mywords.sas7bdat,
          t_out_greeting =example.hello.sas7bdat );

%my_world(
          i_jf_ref       =&example_subflow_ref,
          i_task_name    =my_world );
```

   The two tasks (my_hello and my_world) are added to the subflow (sentence_subflow that has the reference name example_subflow_ref).

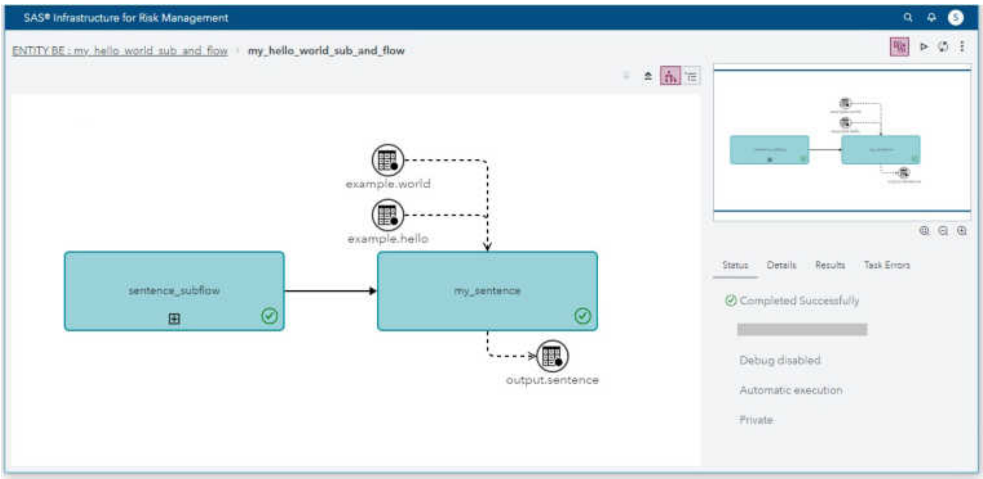
4. (Step 6) Add the remaining task (my_sentence) to the top-level job flow.

```
%my_sentence(
          i_jf_ref     =&example_jf_ref
          i_task_name  =my_sentence );
```

   The single task (my_sentence) is added to the top-level job flow (my_hello_world_sub_and_flow job flow that has the reference name example_jf_ref).

View the Job Flow Instance

After creating the job flow instance, you can view it from the SAS Infrastructure for Risk Management web application. Here is how the job flow instance appears in the SAS Infrastructure for Risk Management web application.


SAS® Infrastructure for Risk Management
ENTITY BE : my_hello_world_sub_and_flow
my_hello_world_sub_and_flow
example.world
example.hello
sentence_subflow
my_sentence
output.sentence
Status
Details
Results
Task Errors

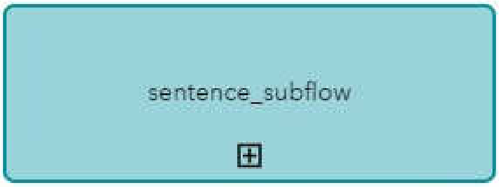
Completed Successfully

Debug disabled

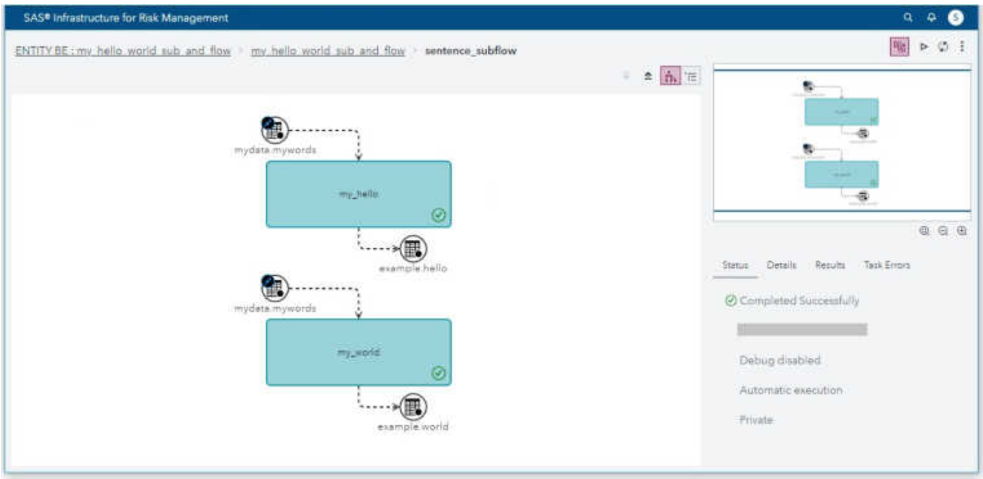
Automatic execution

Private

The subflow (sentence_subflow) is denoted in the SAS Infrastructure for Risk Management web application by a plus (+) sign to indicate that it is not a single task, but a subflow that contains multiple tasks:



To expand the subflow and view the tasks that it contains, double-click the subflow icon in the SAS Infrastructure for Risk Management web application:



Completed Successfully

Debug disabled

Automatic execution

Private

Example 3: Creating a Job Flow That Simplifies the View of a Complex Job Flow

You can configure the inputs and outputs for subflows to display in the job flow diagram in the SAS Infrastructure for Risk Management web application. Here is an example of a job flow script that creates a job flow that performs the same function as your job flow with a subflow in Example 2. However, in this job flow definition, the job flow instance diagram in the SAS Infrastructure for Risk Management web application displays a visual of the inputs and outputs that are associated with a subflow from the top-level, or parent flow, view.

High-Level View of Creating the Job Flow

1. In SAS Studio, click to open a new SAS program.

2. Enter your job flow script (program), which consists of scripting client macros and your tasks. The program is the same as in Example 2, but with the addition of step 5b.

```
/*******************************************************************/
/*  NAME:      my_hello_world_jobflow_with_subflow_visual.sas     */
/*******************************************************************/
/*Step 1  Scripting Client Initialization - Required setup */
%irm_sc_init();

/*Step 2  Create the top-level job flow - create job flow definition */
```

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

/*Step 3  Create the subflow named sentence_subflow */
%irm_sc_build_jobflow(
          i_jf_name      =sentence_subflow,
          o_jf_ref_name  =example_subflow_ref );

/*Step 4  Add the subflow to the top-level job flow */
%irm_sc_add_subflow(
          i_jf_ref       =&example_jf_ref,
          i_sub_jf_ref   =&example_subflow_ref );

/*Step 5  Add tasks (my_hello, my_world) to the subflow */
%my_hello(
          i_jf_ref       =&example_subflow_ref,
          i_task_name    =my_hello,
          t_in_words     =mydata.mywords.sas7bdat,
          t_out_greeting =example.hello.sas7bdat );

%my_world(
          i_jf_ref       =&example_subflow_ref,
          i_task_name    =my_world );

/*Step 5b Add visual input and output to the subflow */
%irm_sc_add_visual_data(
   i_jf_ref                   =&example_subflow_ref,
   i_visual_input_data_list   =mydata.mywords.sas7bdat,
   i_visual_output_data_list  =example.hello.sas7bdat:
                               example.world.sas7bdat );

/*Step 6  Add task (my_sentence) to the top-level job flow */
%my_sentence(
          i_jf_ref       =&example_jf_ref,
          i_task_name    =my_sentence );

/*Step 7  Save this job flow definition to IRM Server */
%irm_sc_save_jobflow(
          i_jf_ref       =&example_jf_ref );

/*Step 8  Execute job flow instance in IRM Server */
%irm_sc_execute_jobflow(
          i_jf_ref       =&example_jf_ref );
```

3. Click 🏃 to run your job flow script.

Detailed View of Creating the Job Flow

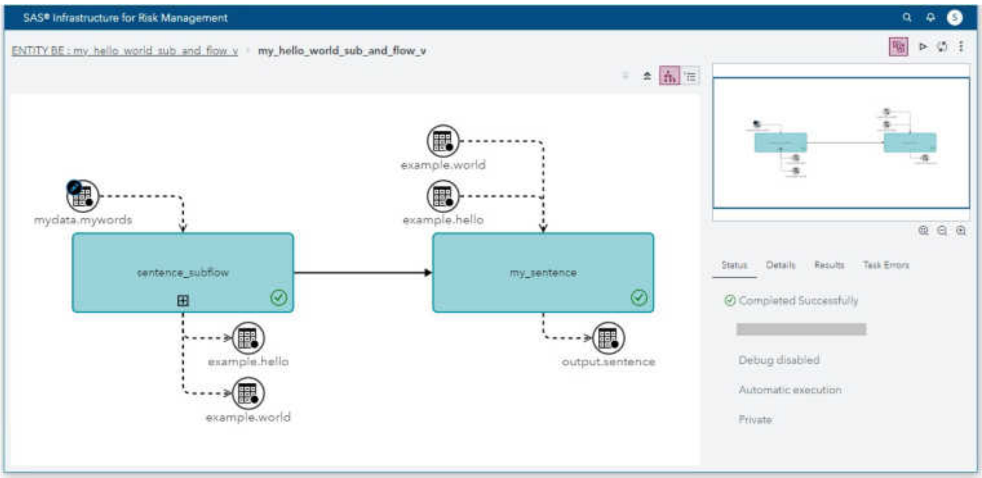
Here is a detailed description of the %irm_sc_add_visual_data macro that you used in step 5b.

Use the %irm_sc_add_visual_data macro to make the subflow’s inputs and outputs visible from the top-level flow:

```
/*Step 5b Add visual input and output to the subflow */
%irm_sc_add_visual_data(
   i_jf_ref                   =&example_subflow_ref,
   i_visual_input_data_list   =mydata.mywords.sas7bdat,
   i_visual_output_data_list  =example.hello.sas7bdat:
                               example.world.sas7bdat );
```

In the %irm_sc_irm_add_visual_data macro, you list the inputs and outputs that you want to display for the subflow task in the SAS Infrastructure for Risk Management web application. Each input (i_visual_input_data_list) and output (i_visual_output_data_list) in the list is delimited by a colon (:).

Here is an example of how the job flow appears in the SAS Infrastructure for Risk Management web application after you create the job flow instance with the %irm_sc_irm_add_visual_data macro that lists the subflow inputs (mydata.mywords) and the subflow outputs (example.hello and example.world):



Completed Successfully

Debug disabled

Automatic execution

Private

Chapter 6
Partitions

About Partitioning

Partitioning enables you to separate large data files into smaller units of data and to dynamically distribute the calculation processes of these smaller units of data across multiple cores. Using partitions increases the speed and efficiency of calculation processing of large tables.

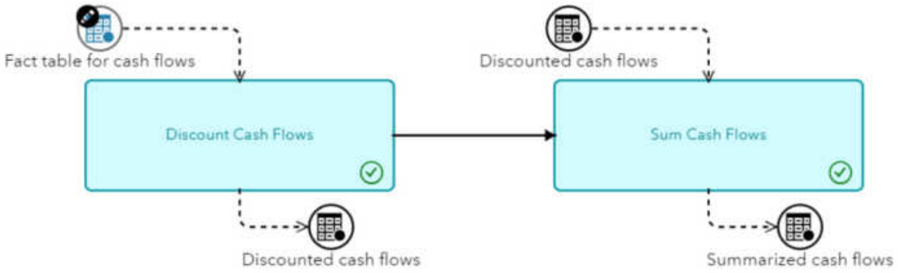
SAS Infrastructure for Risk Management supports the following partition methods:

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

Using Partitions in a Job Flow

About Using Partitions in a Job Flow

Here is a view of a job flow diagram that contains a simple single-threaded job flow with two tasks. No partitioning of data is involved. In this job flow, the first task discounts cash flows and the second task calculates the sum total of the discounted cash flow values.



The steps to create a partition job flow might differ between the partition methods (ByN and ByVar) and the number of computational and analysis tasks in your job flow. Regardless of the partition method that you use, your job flow must perform the following tasks:

1. Calculate the cardinality task.
2. Partition the data task.
3. Perform the computational task.
4. Recombine task.

The partition task files used in the following examples are located in the platform federated area that is installed with SAS Infrastructure for Risk Management. In addition, two sample job flow scripts (simple_byn_partition_flow.sas and simple_byvar_partition_flow.sas) are also provided that you can use as a template for your job flow script.

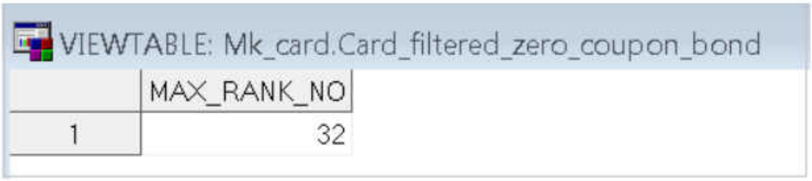
To access the task files, navigate to **`\SASIRM\fa.0.3.6\source\sas\nodes`**. To access the sample job flow scripts, navigate to **`\SASIRM\fa.sample.3.6\client_scripts\sample_basic`**.

Step 1: Calculate the Cardinality Task

About the Cardinality Task

The cardinality task calculates the cardinality of input data based on a user-defined value that is specified in an input configuration data set. The output of this task is a cardinality table. The cardinality table contains one row and one column (MAX_RANK_NO). SAS Infrastructure for Risk Management uses the value of MAX_RANK_NO to determine the number of partitions to dynamically create with the partition tasks.

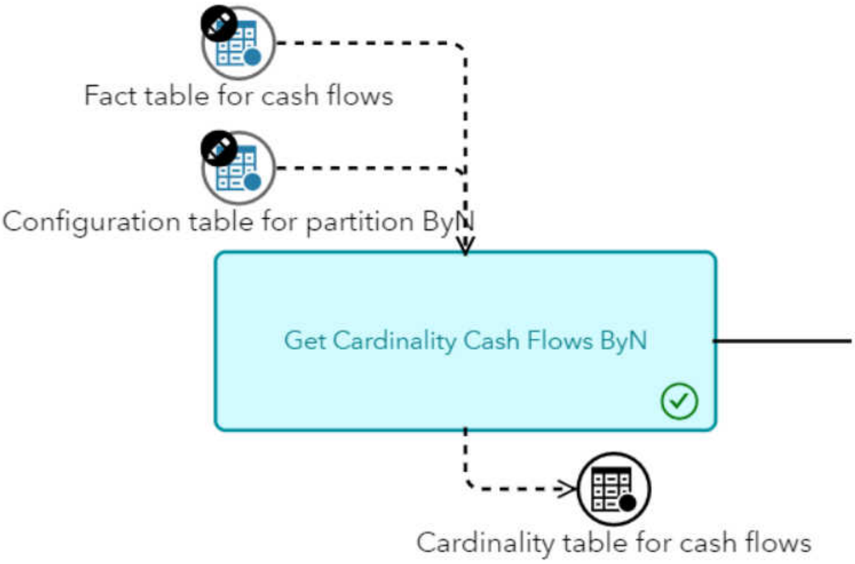
Here is an example of the output cardinality table:


VIEWTABLE: Mk_card.Card_filtered_zero_coupon_bond

	MAX_RANK_NO
1	32

ByN Partition Method

With ByN partitioning, this task calculates the cardinality of input data objects based on a user-defined number that is specified in an input configuration data set. If the number of partitions is not specified, the task sets the cardinality to 1. The task file name is irm_node_get_cardinality_byn.sas.


Fact table for cash flows
Configuration table for partition ByN
Get Cardinality Cash Flows ByN
Cardinality table for cash flows

Here are the identified input data objects:

- `%IN_DS_PART_BYN=STAGING.cashflows.sas7bdat`

  The parameter that specifies the data set to be partitioned.

- `%IN_BYN_CONF=IRM_CFG.byn_configuration.sas7bdat`

  The parameter that specifies the configuration data set that contains the user-defined number of partitions.

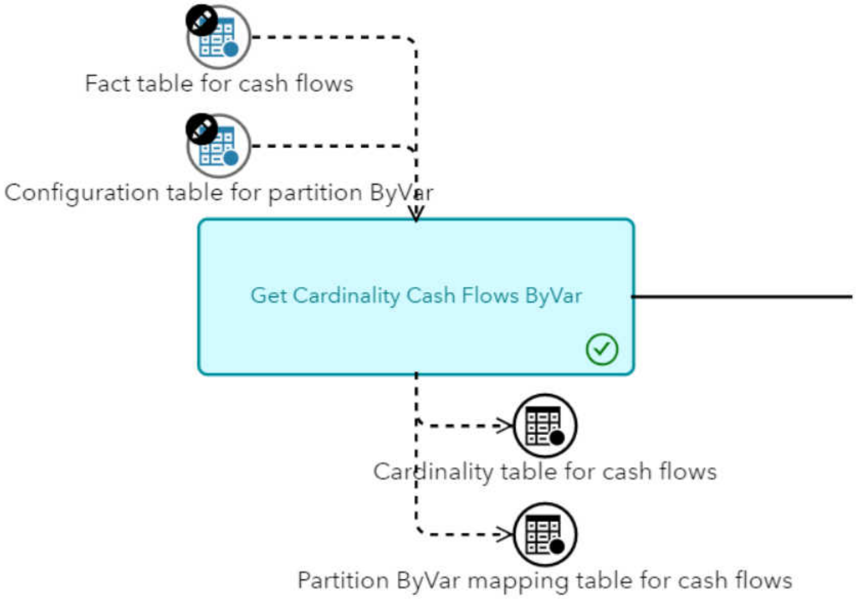
Here is the identified output data object:

- `%OUT_CARD_DS_BYN=MK_CARD.card_cashflows_byn.sas7bdat`

  The parameter that specifies the cardinality table.

ByVar Partition Method

With ByVar partitioning, this task calculates the cardinality of input data objects based on user-defined BY variables that are specified in an input configuration file. If a BY variable is not specified, or no valid BY variable is provided, the task sets the cardinality to 1 and creates an empty ByVar mapping table. The task file name is irm_nod_get_cardinality_byvar.sas.


Fact table for cash flows
Configuration table for partition ByVar
Get Cardinality Cash Flows ByVar
Cardinality table for cash flows
Partition ByVar mapping table for cash flows

Here are the identified input data objects:

- `%IN_DS_PART_BYVAR=MK_OPT.filtered_option_instrument.sas7bdat`

  The parameter that specifies the data set to be partitioned.
- `%IN_BYVAR_CONF=IRM_CFG.byvar_configuration.sas7bdat`

  The parameter that specifies the configuration data set that contains the user-defined BY variable or variables.

Here are the identified output data objects:

- `%OUT_CARD_DS_BYVAR=MK_CARD.card_filtered_option_instrument.sas7bdat`

  The parameter that specifies the cardinality table.
- `%OUT_CARD_MAP=MK_CARD.part_option_inst_mapping.sas7bdat`

  The parameter that specifies the BY variable mapping table.

Step 2: Partition the Data Task

About the Partition Task

When the SAS Infrastructure for Risk Management server identifies a partition task, it reads the output cardinality table to create MAX_RANK_NO threads. In every thread, the partition task slices the data set using the ByN or ByVar method.

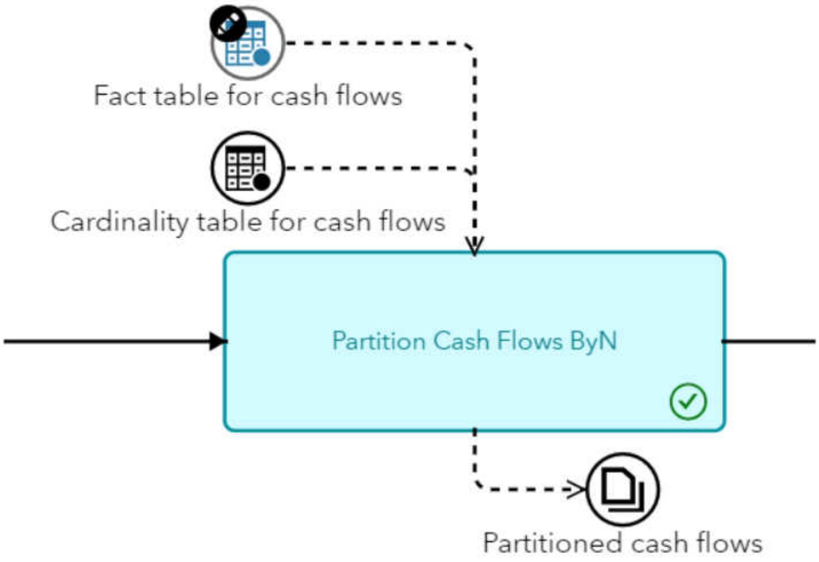
At a minimum, the partition task must have the following inputs and the single output:

- Input 1: The cardinality table that is created as output by Step 1 on page 42.
- Input 2: The table to partition.
- Input 3: The partition mapping table that is created by Step 1 on page 42.

  Note: This applies to the ByVar method only.
- Output: The partitioned results data.

ByN Partition Method

With ByN partitioning, this task partitions the input data based on the ByN cardinality table. The task file name is irm_node_partition_byn.sas.



Here are the identified input data objects:

- `%DS_TO_PART=mk_opt.filtered_option_instrument.sas7bdat`

  The parameter that specifies the data set to be partitioned.

- `%CARDINALITY_DS=IRM_CFG.byn_configuration.sas7bdat`

  The parameter that specifies the cardinality data that indicates how many partitions to create.

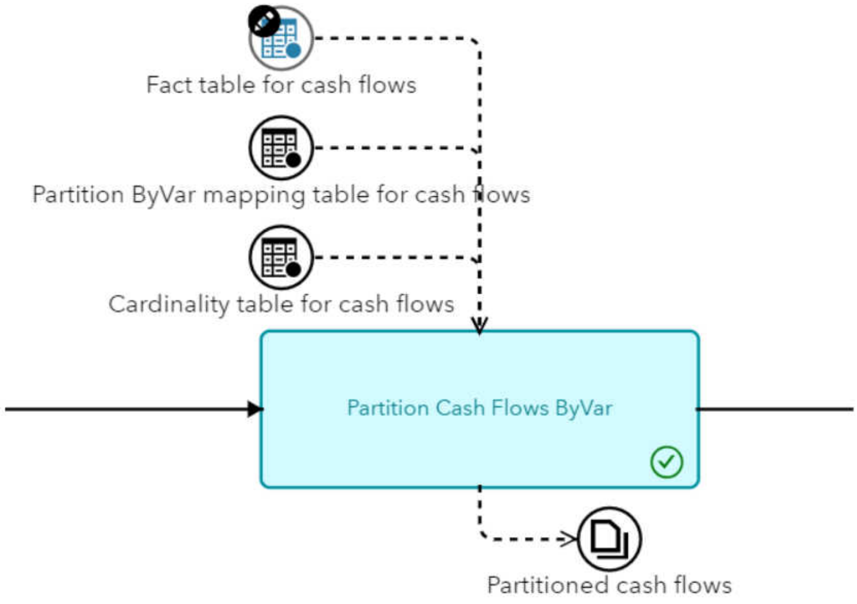
Here is the identified output data object:

- `%PART_DS=P_MK_OPT.filtered_option_instrument.sas7bdat`

  The parameter that specifies the partitioned data.

ByVar Partition Method

With ByVar partitioning, this task partitions the input data based on the ByVar cardinality table and the mapping table. The task file name is irm_node_partition_byvar.sas.


Fact table for cash flows
Partition ByVar mapping table for cash flows
Cardinality table for cash flows
Partition Cash Flows ByVar
Partitioned cash flows

Here are the identified input data objects:

- `%IN_PART_DS=MK_BOND.filtered_zero_coupon_bond.sas7bdat`

  The parameter that specifies the data set to be partitioned.

- `%IN_CARD_MAP=MK_CARD.part_zeroc_inst_mapping.sas7bdat`

  The parameter that specifies the ByVar mapping table that contains names and values of the BY variables.

- `%IN_CARD_DATA=MK_CARD.card_filtered_cashflows.sas7bdat`

  The parameter that specifies cardinality data that indicates how many partitions to create.

Here is the identified output data object:

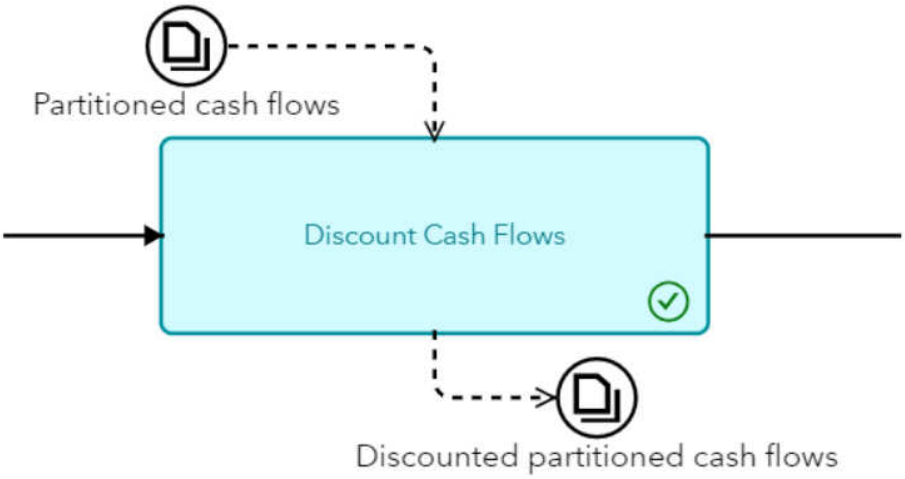
- `%OUT_PART_DS=P_MK_BD.filtered_zero_coupon_bond.sas7bdat`

  The parameter that specifies the partitioned data.

Step 3: Compute the Partitioned Data Task

The computational task is a standard computing task except for the following additional characteristics:

- It must have partitioned input data and output data.
- If more than one input data object exists, some data objects can be non-partitioned, but all instances of the computational task receive the same input.
- The partitioned outputs must have different library names.



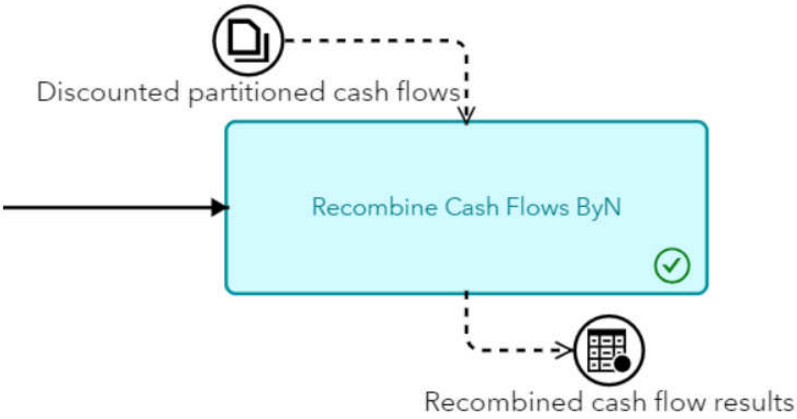
Step 4: Recombine the Results Task

About the Recombine Task

With the Recombine task, partitioned data is recombined as one non-partitioned data output. The recombine task is single-threaded.

ByN Partition Method

With ByN partitioning, this task recombines a set of ByN partitioned input data as one non-partitioned data output. The task file name is irm_node_recombine_byn.sas.


Discounted partitioned cash flows
Recombine Cash Flows ByN
Recombined cash flow results

Here is the identified input data object:

- `%tablein=P_MK_OPT.option_instrument_bi_result.sas7bdat`

  The parameter that specifies the dictionary of identical structure data frames that represent partitioned input data.

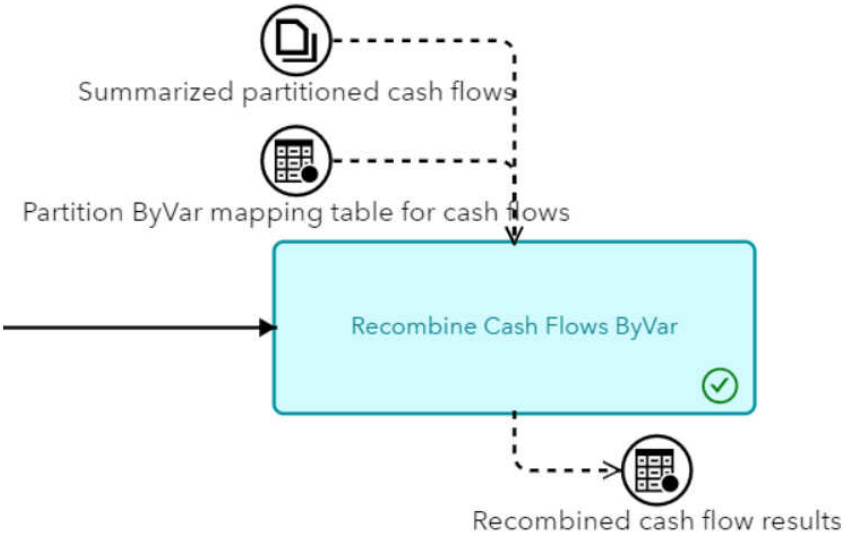
Here is the identified output data object:

- `%tableout=R_MK_OPT.option_instrument_bi_result.sas7bdat`

  The parameter that specifies the data frame that is obtained by appending all input data frames. The output table has the same structure as an input partitioned data frame.

ByVar Partition Method

With ByVar partitioning, this task recombines a set of ByVar partitioned input data as one non-partitioned data output. During the recombine process, the ByVar variables are added back to the output data.


Summarized partitioned cash flows
Partition ByVar mapping table for cash flows
Recombine Cash Flows ByVar
Recombined cash flow results

Here are the identified input data objects:

- `%table_in_nm=P_MK_BD.zero_coupon_bond_result.sas7bdat`

  The parameter that specifies the dictionary of identical structure data frames that represent partitioned input data.

- `%IN_CARD_MAP_DS=MK_CARD.part_zeroc_inst_mapping.sas7bdat`

  The parameter that specifies the ByVar mapping table that contains names and values of the BY variables.

Here is the identified output data object:

- `%table_out_nm=R_MK_BD.zero_coupon_bond_result.sas7bdat`

  The parameter that specifies the data frame that is obtained by appending all input data frames. The output table has the same structure as an input partitioned data frame.

Chapter 7

Accessing SAS Infrastructure for Risk Management Information Using WebDAV

About Using WebDAV to Access Your SAS Infrastructure for Risk Management Information

SAS Infrastructure for Risk Management uses Web Distributed Authoring and Versioning (WebDAV) to provide an easy way for users to access the following types of SAS Infrastructure for Risk Management information:

- job flow definition files
- input and output SAS data sets and their corresponding Microsoft Excel files
- the execution status of job flow instances and sub-flow instances
- the execution status of tasks
- the navigation hierarchy of the data

Before using WebDAV to access SAS Infrastructure for Risk Management data, note the following restrictions:

- The scope of data that a user can access is controlled by permissions that are associated with the user’s logon credentials.
- The content of a flow instance that has not been shared is accessible only to the owner of the flow. If the instance is shared, the content is accessible to users who have access to the business entities of the flow instance.
- The contents of the SAS Infrastructure for Risk Management WebDAV servlet are Read-Only and cannot be deleted.

SAS Infrastructure for Risk Management users can access the data store in a centralized location on a remote server using WebDAV by using either of the following methods:

- using a WebDAV URL in a browser
- using a WebDAV drive that is mapped to your computer

- using the LIBNAME statement to directly map a libref to a WebDAV URL

Using a WebDAV URL in a Browser

You can access your SAS Infrastructure for Risk Management information using a WebDAV URL in your browser.

Here is the general form of the WebDAV URL that you enter in your browser:

```
protocol//server_name:port/SASIRMServer/irmwebdav/userid
```

In the following example, you (sasdemo) create the hello_world job flow instance. (See “Creating Your First Parallel Program”.).

To view the job flow information using the WebDAV URL in the browser:

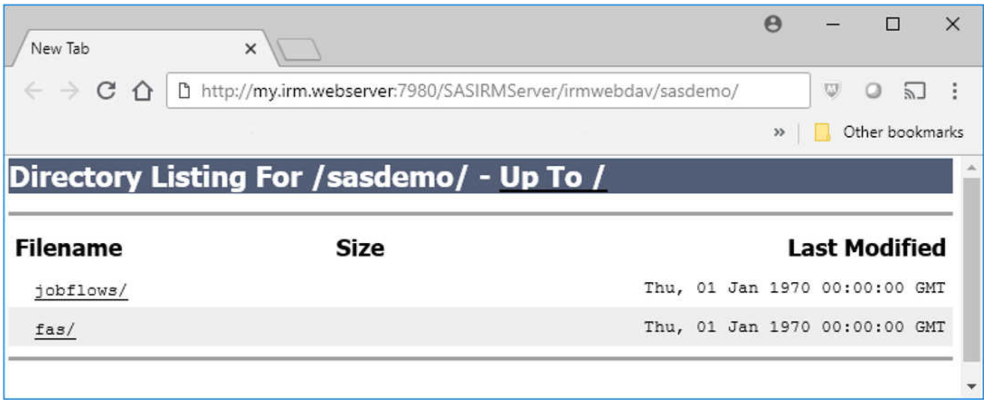
1. Enter the WebDAV URL in your browser.

   In this example, the URL for the SAS Infrastructure for Risk Management WebDAV server is:

   ```
   http://my.irm.webserver:7980/SASIRMServer/irmwebdav/sasdemo/
   ```

   Note: You will be prompted to log on to the SAS Infrastructure for Risk Management server.

   Here is how the WebDAV directories appear once you have logged on to the server:

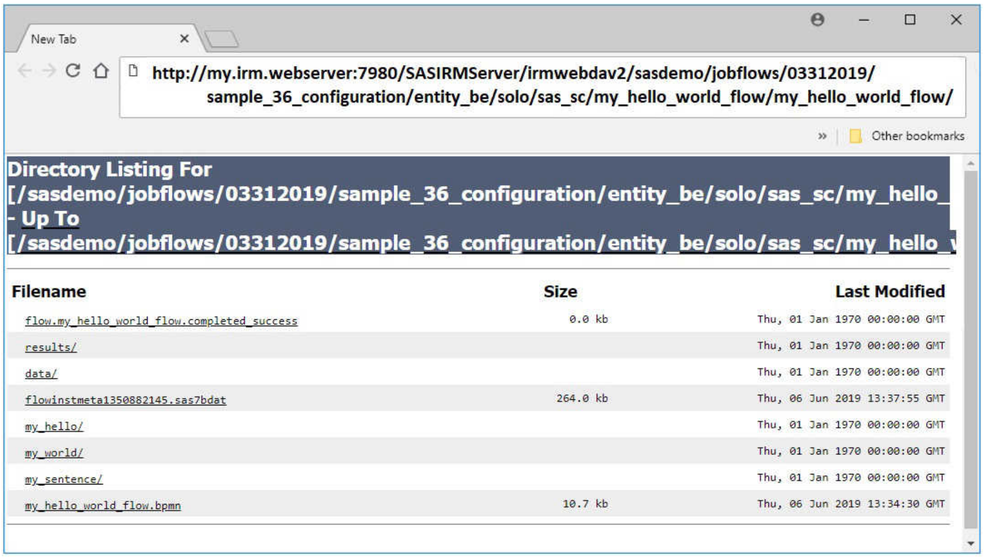

New Tab
http://my.irm.webserver:7980/SASIRMServer/irmwebdav/sasdemo/
»
Other bookmarks

Directory Listing For /sasdemo/ - Up To /

Filename	Size	Last Modified
jobflows/		Thu, 01 Jan 1970 00:00:00 GMT
fas/		Thu, 01 Jan 1970 00:00:00 GMT

2. As needed, drill down to find information. Click **Up To** to drill up the directory tree.

   If any instances have been created in your personal federated area, you can view the logs, status, data, and other files by drilling down through the job flow directory. For example, to see the hello_world job flow instance log and other information, follow the path **`jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/hello_world_flow/hello_world`**.



http://my.irm.webserver:7980/SASIRMServer/irmwebdav2/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/my_hello_world_flow/my_hello_world_flow/

Directory Listing For [/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/my_hello_
- Up To [/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/my_hello_

Filename	Size	Last Modified
flow.my_hello_world_flow.completed_success	0.0 kb	Thu, 01 Jan 1970 00:00:00 GMT
results/		Thu, 01 Jan 1970 00:00:00 GMT
data/		Thu, 01 Jan 1970 00:00:00 GMT
flowinstmeta1350882145.sas7bdat	264.0 kb	Thu, 06 Jun 2019 13:37:55 GMT
my_hello/		Thu, 01 Jan 1970 00:00:00 GMT
my_world/		Thu, 01 Jan 1970 00:00:00 GMT
my_sentence/		Thu, 01 Jan 1970 00:00:00 GMT
my_hello_world_flow.bpmn	10.7 kb	Thu, 06 Jun 2019 13:34:30 GMT

Mapping a WebDAV Drive to Your Computer

You can access your SAS Infrastructure for Risk Management information using a network drive on your computer that is mapped to the WebDAV server.

To access the data from a WebDAV drive that is mapped to your computer:

1. Map the SAS Infrastructure for Risk Management WebDAV servlet drive to your computer. For information about mapping the WebDAV servlet drive to your computer, refer to the documentation for your operating system.

   Here is the general form of the network drive folder name:

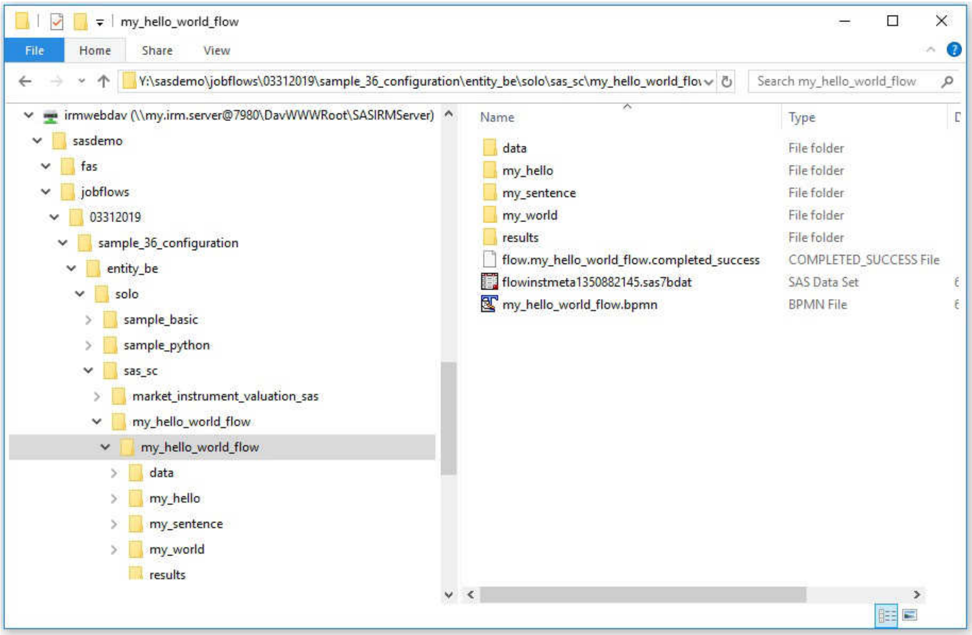
   ```
   protocol//server_name:port/SASIRMServer/irmwebdav
   ```

   Note: You will be prompted to log on to the SAS Infrastructure for Risk Management server.

2. Access the drive on your local system and click your user ID to navigate to the data that you want to access.

In the following example, you (sasdemo) create the hello_world job flow instance. (See “Creating Your First Parallel Program”.) You map a network drive to the WebDAV server. The network drive folder name used in this example is **`http://my.irm.webserver:7980/SASIRMServer/irmwebdav/`**.

After you map the network drive, you can view the job flow instance logs, tasks, data, and other information:



Y:\sasdemo\jobflows\03312019\sample_36_configuration\entity_be\solo\sas_sc\my_hello_world_flo

```
irmwebdav (\\my.irm.server@7980\DavWWWRoot\SASIRMServer)
  sasdemo
    fas
    jobflows
      03312019
        sample_36_configuration
          entity_be
            solo
              sample_basic
              sample_python
              sas_sc
                market_instrument_valuation_sas
                my_hello_world_flow
                  my_hello_world_flow
                    data
                    my_hello
                    my_sentence
                    my_world
                    results
```

Name	Type
data	File folder
my_hello	File folder
my_sentence	File folder
my_world	File folder
results	File folder
flow.my_hello_world_flow.completed_success	COMPLETED_SUCCESS File
flowinstmeta1350882145.sas7bdat	SAS Data Set
my_hello_world_flow.bpmn	BPMN File

In addition to viewing the files, you can access them directly for reporting or other tasks.

Note: If your SAS Infrastructure for Risk Management server is running HTTP and you are running Windows 10 on your local machine, you might have to modify the Windows Registry to allow basic authentication to WebDAV over an HTTP connection before you can map a network drive to the SAS Infrastructure for Risk Management WebDAV servlet . For more information, refer to the Microsoft Windows documentation.

Directly Mapping a Libref to a WebDAV URL

Note: To be able to directly map a libref to a WebDAV URL, the user credentials that you use in your SAS session must be stored in the SAS Metadata Server.

You can associate a libref with a SAS library to enable access to directories on a WebDAV server. You can then use this library to access the data that is associated with your job flow instance in SAS Studio or in a local SAS session.

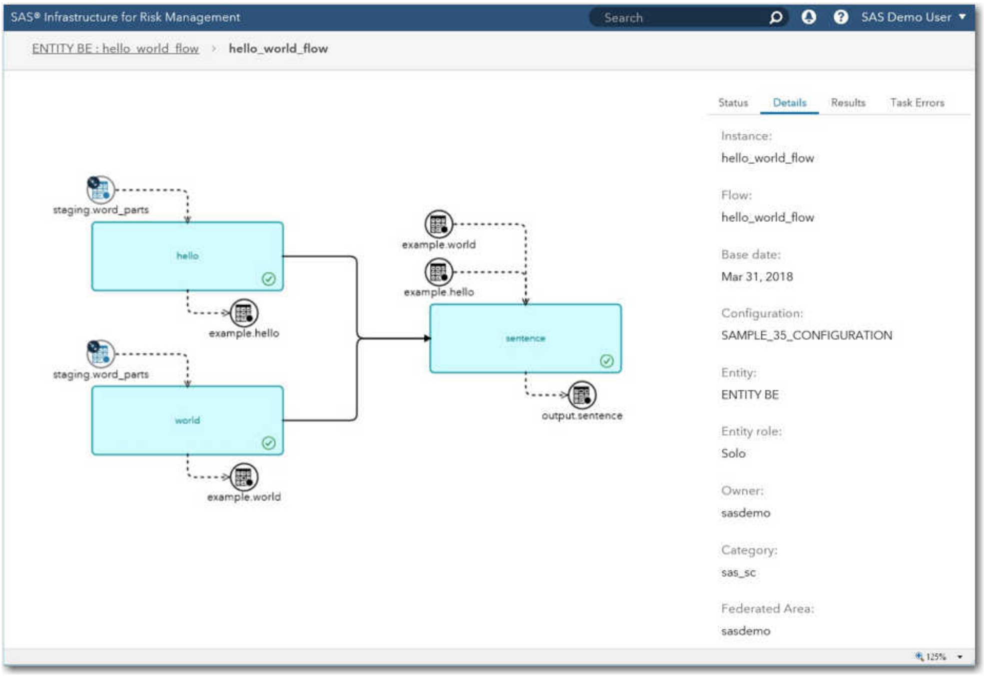
Here is the format of the LIBNAME statement to access the WebDAV server for your job flow instance data:

```
libname mylibref "protocol//server_name:port/SASIRMServer/
irmwebdav/userid/jobflows/base_date/config_set_id/entity/
entity_role/flow_category/flow_bpmn_name/instance_name" WEBDAV
AUTHDOMAIN="DefaultAuth" user="username";
```

Here is an example of how you would use a WebDAV library to produce a simple SAS report from the output data of a job flow instance.

In this example, you (sasdemo) created the hello_world job flow instance. (See “Creating Your First Parallel Program”.) Now, you want to produce a simple SAS report from the instance output data (output.sentence).

Here is the job flow instance diagram and the details that are displayed in the SAS Infrastructure for Risk Management web application:



Instance:
hello_world_flow

Flow:
hello_world_flow

Base date:
Mar 31, 2018

Configuration:
SAMPLE_35_CONFIGURATION

Entity:
ENTITY BE

Entity role:
Solo

Owner:
sasdemo

Category:
sas_sc

Federated Area:
sasdemo

The **Details** tab for this job flow instance displays the information that you need to supply in the LIBNAME statement to connect to the WebDAV data directory for this job flow instance: **Instance** (*instance_name*), **Flow** (*flow_bpmn_name*), **Base date** (*base_date*), **Configuration** (*config_set_id*), **Entity** (*entity*), **Entity role** (*entity_role*), **Owner** (*userid*), and **Category** (*flow_category*).

To create a SAS library to access the job flow instance WebDAV directory and produce the simple report using one of the output data sets:

1. Invoke a SAS session.

   Note: This example uses a local SAS session, but you can make the LIBNAME assignment in any type of SAS session that has access to the SAS Infrastructure for Risk Management server.

2. Submit the LIBNAME statement to access the WebDAV data directory for the instance.

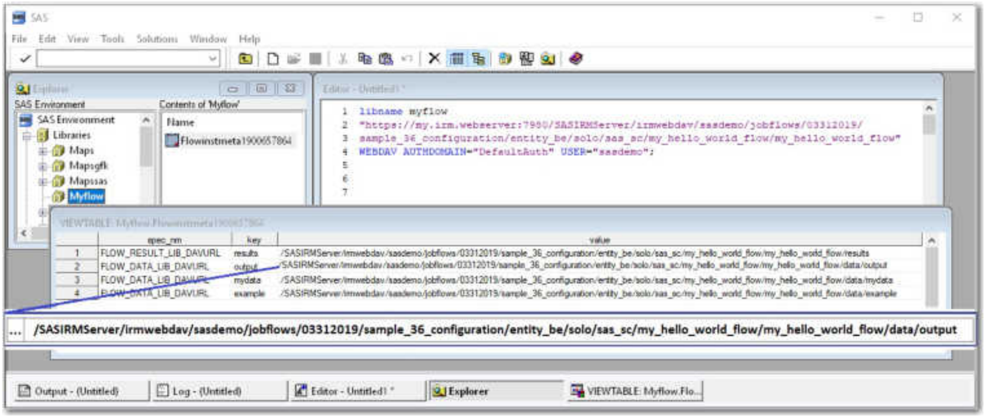
   The **Details** tab of the job flow instance in the SAS Infrastructure for Risk Management web application displays the information that you need to assign the WebDAV library. In this example, the library MYFLOW is created for the hello_world_flow job flow instance data as follows:

   ```
   libname myflow "http://my.irm.webserver:7980/SASIRMServer/
   irmwebdav/sasdemo/jobflows/03312019/sample_36_configuration/
   entity_be/solo/sas_sc/hello_world_flow/hello_world_flow"
   WEBDAV AUTHDOMAIN="DefaultAuth" USER="sasdemo";
   ```

   Note: Depending on your SAS session, you might be prompted to log on to the SAS Infrastructure for Risk Management server when you submit this LIBNAME statement.

3. The MYFLOW library that was created in step 2 contains the job flow instance metadata table (flowinstmeta*.sas7bdat).

   To see a listing of the WebDAV URLs for the job flow instance data, view the contents of the job flow instance metadata table:



```
libname myflow
"https://my.irm.webserver:7980/SASIRMServer/irmwebdav/sasdemo/jobflows/03312019/
sample_36_configuration/entity_be/solo/sas_sc/my_hello_world_flow/my_hello_world_flow"
WEBDAV AUTHDOMAIN="DefaultAuth" USER="sasdemo";
```

	spec_nm	key	value
1	FLOW_RESULT_LIB_DAVURL	results	/SASIRMServer/irmwebdav/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/my_hello_world_flow/my_hello_world_flow/results
2	FLOW_DATA_LIB_DAVURL	output	/SASIRMServer/irmwebdav/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/my_hello_world_flow/my_hello_world_flow/data/output
3	FLOW_DATA_LIB_DAVURL	mydata	/SASIRMServer/irmwebdav/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/my_hello_world_flow/my_hello_world_flow/data/mydata
4	FLOW_DATA_LIB_DAVURL	example	/SASIRMServer/irmwebdav/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/my_hello_world_flow/my_hello_world_flow/data/example

... /SASIRMServer/irmwebdav/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/my_hello_world_flow/my_hello_world_flow/data/output

The data set (OUTPUT.sentence) that is needed for the report is in the WebDAV output directory. The WebDAV directory path for the output directory shown in the job flow instance metadata table is **`SASIRMServer/irmwebdav/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/hello_world_flow/hello_world_flow/data/output`**.

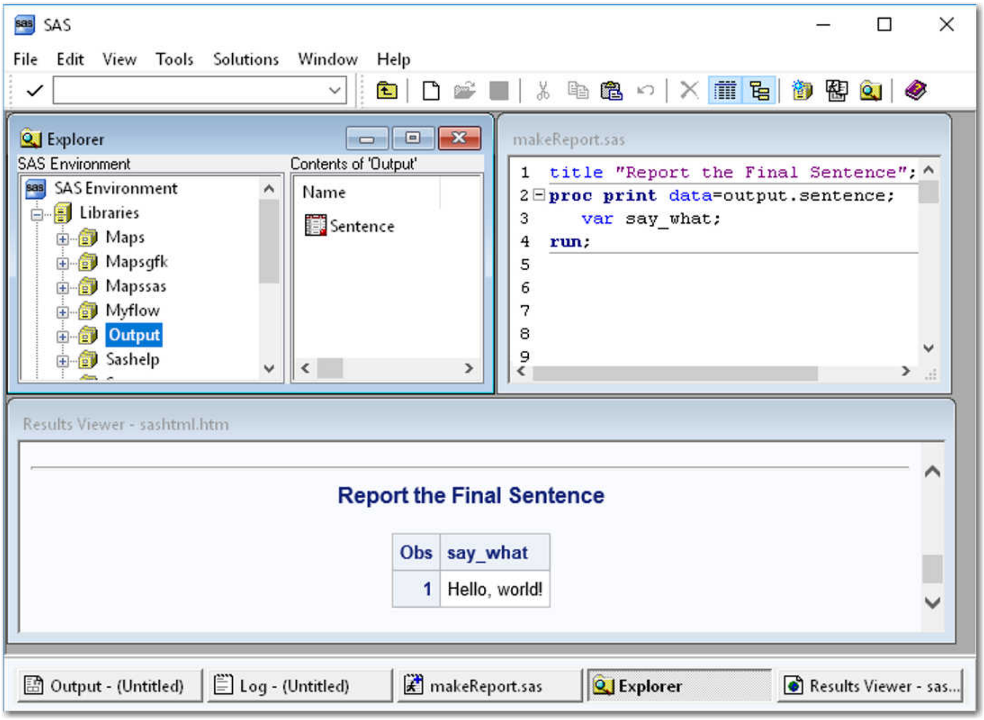
4. Submit the LIBNAME statement to access the WebDAV data directory for this job flow instance's output library:

   ```
   libname output "http://my.irm.webserver:7980/SASIRMServer/
   irmwebdav/sasdemo/jobflows/03312019/sample_36_configuration/
   entity_be/solo/sas_sc/hello_world_flow/hello_world_flow/
   data/output" WEBDAV AUTHDOMAIN="DefaultAuth" USER="sasdemo";
   ```

5. Produce the report accessing the job flow instance data via the WebDAV library in this local SAS session:

   ```
   title "Report the Final Sentence";
   proc print data=output.sentence;
      var say_what;
   run;
   ```

Here is how the report appears in the local SAS session:



```
title "Report the Final Sentence";
proc print data=output.sentence;
   var say_what;
run;
```

Report the Final Sentence

Obs	say_what
1	Hello, world!

For detailed information about using the LIBNAME statement for WebDAV server access, see *SAS Global Statements: Reference*.

Chapter 8
Additional Programming Features

Enabling a Task to Request a Service Ticket

A SAS task might need to make REST calls to other SAS applications. To do so, a task needs access to a ticket-granting ticket (which contains the logged in user's credentials). The tasks requests a service ticket (ST) from the TGT. The ST provides access to the user's credentials. This enables a user to log on to SAS Infrastructure for Risk Management using their credentials and then log on to additional SAS web applications (for example, SAS Studio) without being challenged for log on credentials. When using TGT, user credentials do not need to be stored in the SAS Metadata Server.

Note: STs can be used for only one service.

To enable a SAS task to request a service ticket, declare the following SAS Infrastructure for Risk Management `%irm_tgt` reserved substitution parameter as task input in the task file header:

```
\param[in] %irm_tgt=expiration-value
```

where *expiration-value* specifies how access to TGT expires. The two possible values are:

- `EXPIRE_WITH_TASK`

%irm_tgt=EXPIRE_WITH_TASK, TGT will expire when the task execution is completed. This is the most secure option, because it ensures that a ticket cannot be used by another process later.

- **EXPIRE_WITH_TICKET**

  If set to **%irm_tgt=EXPIRE_WITH_TICKET**, TGT will expire when the ST expires or immediately following the execution of the task, if the execution completed with errors.

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

To make the TGT support more useful, SAS Infrastructure for Risk Management provides the following new macro variables and SAS auto call macro:

- **IRM_MIDTIER_HOST** — contains the host name of the SAS Infrastructure for Risk Management mid-tier server.
- **IRM_MIDTIER_PORT** — contains the port number that the SAS Infrastructure for Risk Management server is running on.
- irm_rest_service_ticket_from_tgt.sas — creates an ST from a TGT and URL.

Deleting a Job Flow Definition

To delete a job flow definition from your personal federated area, use the %irm_sc_del_flow_def macro.

For details about the %irm_sc_del_flow_def macro parameters, see Appendix 2, “Scripting Client Macro Reference”.

Reusing Subflows

You can use the %irm_sc_add_flow_from_fa macro to reuse existing subflows in new job flow definitions.

Suppose you have created the job flow (F_A) and a subflow (SUB_COMMON) in your personal federated area. In a new job flow script, you want to create a new job flow (F_B) that reuses the existing subflow (SUB_COMMON).

When you reuse existing job flow definitions and subflow definitions, the following rules apply:

- You can reuse only job flows that are created by the scripting client.
- Job flows can come from your personal federated area or a Read-only federated area.

Configuring the Task-Level Call Back and Cleanup Method

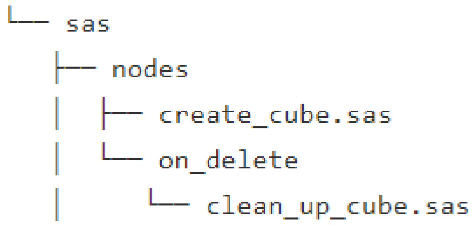
The SAS Infrastructure for Risk Management scripting client provides a call back and clean up method that you can configure at the task level. When the method has been configured for a task, and you delete a job flow instance that the task is a part of, SAS Infrastructure for Risk Management cleans up the external resources associated with the task before it deletes the job flow instance.

When using the call back and clean up method, note the following details about the feature:

- The method is configured per task and is performed by SAS Infrastructure for Risk Management before it deletes the job flow instance.
- The method is implemented by using a SAS Infrastructure for Risk Management task node that is named clean_up_*file_name*.sas, where *file_name* is X. The clean_up.sas task is located under the system-reserved task category that is named on_delete.

  Note: You cannot use tasks that are located in the system-reserved task category in job flow definitions.
- You configure the call back and clean up method in a task by adding the Doxygen syntax **`\ref on_delete/call_back_task_file_name.sas`** to the task code, where *file_name* is the name of the ???. You cannot declare inputs or outputs in the clean_up task.

Here is an example of the call back and clean up method that uses the following tasks.



```
└── sas
    ├── nodes
    │   ├── create_cube.sas
    │   └── on_delete
    │       └── clean_up_cube.sas
```

Table 8.1 *Call Back and Cleanup Method Tasks*

Task	**Description**
create_cube.sas	User-defined task that creates an external RD data cube in the file system that is not managed by SAS Infrastructure for Risk Management.
clean_up_*file_name*.sas task	System-reserved task that cleans up the resources created by the create_cube.sas task, where *file_name* is the name of ???. This

Here is the Doxygen file header syntax in the create_cube.sas task that registers the **`on_delete/clean_up_cube.sas`** task as its call back and clean up task. Call back and cleanup parameters and commands are highlighted in the example.

```
/*
 Copyright (C) 2018 SAS Institute Inc. Cary, NC, USA
*/

/**
   \file

   \brief The node filters cash flow and creates RD cube in file system.

   This will invoke call back clean up task before deletion of job flow
   instance \ref on_delete/clean_up_cube.sas


   <b> Identified Inputs and outputs </b>

   \param[in]  %IN_DS=MK_CF.cashflows.sas7bdat                  input data

   \param[out] %OUT_DS=MK_CF.filtered_cashflow.sas7bdat         output data

  \ingroup nodes

   \author SAS Institute INC.

   \date 2018

 */
```

Here is the Doxygen file header syntax in the clean_up_cube.sas task in the system reserved on_delete folder.

```
/*
 Copyright (C) 2018 SAS Institute Inc. Cary, NC, USA
*/

/**
   \file

   \brief The node cleans up RD cube in file system.

  \ingroup on_delete

   \author SAS Institute INC.

   \date 2018

 */
```

Using Generic Libraries to Bypass Live ETL

SAS Infrastructure for Risk Management supports generic library mapping definitions. Generic library mapping definitions enable access to third-party data that is located outside of a SAS Infrastructure for Risk Management federated area. For example, this data might be located in a relational database management system, Hadoop, CAS, and so on.

Generic libraries provide an easy method to bypass live ETL or to upload data objects into job flow instances. For detailed information about generic library mapping definitions, see "Generic Library Mapping" in *SAS Infrastructure for Risk Management: Administrator's Guide*.

Although the use of a generic library is convenient and useful during development, they are not recommended for deployments on production systems. Generic libraries break an essential SAS Infrastructure for Risk Management design premise, which is that SAS Infrastructure for Risk Management controls the landing area (%LA), the input area (%IA), and the persistent area (%PA).

This design premise is essential in guaranteeing Atomicity, Consistency, Isolation, Durability (ACID) transactions during the execution of a job flow instance. Therefore, if you use generic libraries for the design of your federated area, you must provide manual or automated workflows to work around the lack of transactional integrity for your job flows.

Here is a partial list of the challenges that you or your users could experience if you use generic libraries in your job flows:

- Correctness
  - You must consider that the input data at the beginning of the execution of the flow is not guaranteed to be immutable for the duration of the execution. For example, if MK_OPT.positions.sas7bdat is the input data set to two tasks, the two tasks could read different values for the content of MK_OPT.positions.sas7bdat when they execute. With generic libraries, SAS Infrastructure for Risk Management cannot provide isolation or consistency.
  - For downloads, there is no guarantee that the data being downloaded is the result of the computation of the job flow instance. With generic libraries, SAS Infrastructure for Risk Management cannot guarantee durability.
  - To manually provide isolation during flow execution, consider that care must be exercised to avoid sharing data objects across flow instances.
  - When you use generic libraries, data conversions are automatically handled by SAS/ACCESS, which maps conversions to a string and a number. With this behavior in mind, it is impossible to write a general-purpose task that filters any table in a relational database management system (RDBMS).
  - For library engines that do not support concurrent access (for example, file systems), flow executions could fail when the data is produced by SAS Infrastructure for Risk Management (for example, output to a file) or accessed outside of SAS Infrastructure for Risk Management (for example, a file as input to a job flow).

    For example, if you have a job flow definition that creates a report.txt file outside of the persistent area, two or more concurrent executions of an instance of that flow could fail with an I/O error.
  - With generic libraries, the SAS Infrastructure for Risk Management task scheduling algorithm is no longer guaranteed to trigger task executions at the correct time. In particular, SAS Infrastructure for Risk Management will not detect cycles that are triggered by a dependency in the third-party system.
  - The auditing for uploads and downloads might not be accurate since the auditing might occur in the third-party system.
- Security

- Without SAS Infrastructure for Risk Management asserting control over data that is read or written, there is no guarantee that the data is filtered according to the permissions of the user.
- A separate authentication domain might have to be created for each user to access SAS Infrastructure for Risk Management securely.

- Performance
  - Tasks with generic libraries cannot leverage data object pooling. A lack of data object pooling typically results in an order of magnitude of performance degradation.
  - The SAS Infrastructure for Risk Management scheduling algorithm can no longer optimize disk access for tables that are held in a third-party system. This is true for a single machine or a grid deployment.
  - During flow execution, multiple copies of the input and output data might occur between SAS Infrastructure for Risk Management and the third-party system (in both directions). It is recommended that you do not write SAS Infrastructure for Risk Management job flows that keep data in a third-party system rather than in the persistent area. Flow execution of data in a third-party system is relatively slow and expensive.

In addition, there is no live ETL with generic libraries since they do not use the input area or the landing area. The result of no live ETL is that when you use generic libraries, SAS Infrastructure for Risk Management can no longer reliably determine whether your job flow instances are up-to-date. Consequently, your users must ignore version information in the SAS Infrastructure for Risk Management web application about whether a job flow instance is current.

Before you adopt generic libraries for your job flow and task designs, carefully review these limitations and make sure that they will not interfere with the operations of your solution or that you have appropriate workarounds in place.

Configuring a One-Time Password for a Task

SAS Infrastructure for Risk Management provides system-reserved parameters that you can use to tag a task to indicate that a one-time password is required to execute the task. A one-time password might be needed to execute a task when the task requires credentials to access other applications outside of the SAS Infrastructure for Risk Management server.

To configure a one-time password for a task, include the following parameter in the task file header:

```
\param[in] %IRM_NUM_OF_ONE_TIME_PASSWORD=1
```

The parameter value specifies the number of one-time passwords that can be generated for the task.

After specifying the one-time password parameter in the task header, you reference the user ID and one or more generated passwords in the task code as &IRM_USER_ID and &IRM_USER_PASSWORD.

Here are examples of task code statements that reference the user ID and one or more one-time passwords:

- **`%query_metadatalibs(userid=&IRM_USER_ID., password=&IRM_USER_PASSWORD.);`**

- `%query_metadatalibs(userid=&IRM_USER_ID., password=&IRM_USER_PASSWORD_2.);`

Note: The one-time password feature is typically used with generic libraries during development and is not usually a part of deployments on production systems. Using generic libraries and one-time passwords violates the SAS Infrastructure for Risk Management job flow task's use of immutable inputs and outputs. For information about this approach, see "Using Generic Libraries to Bypass Live ETL".

Configuring an Output URL

About Configuring Output URLs

You can create a task that produces a URL as an associated attribute of its output data object. When the URL attribute is available, you can follow the URL and also download the output from the SAS Infrastructure for Risk Management web application.

An output URL file is not a declared task output of the task, but is an attribute file of a declared task output.

Here are the characteristics of the URL output file:

- The output data object is a name-value pair text file whose name ends with the extension .url.
- The library and base name must match the output data object with which the output is associated. If the output is a data collection, the file name of the URL file matches the library name and is stored in the collection folder.
- If the library is a generic library, the URL is in a folder under the job flow instance at the same level as the other non-generic libraries. If the library is referenced as output, then the name of the URL file matches the folder name (same as a collection URL). If the output references the generic library and a table in the library, the URL file name matches the table name.
- An output URL can contain substitution parameters.

Output URL Data Object Details

The output data object for the output URL is a text file that consists of name-value pairs. The keys (data identifier, which is on the left side of the equal sign) cannot contain spaces.

Here are the keys that SAS Infrastructure for Risk Management supports:

Table 8.2 *Output URL Supported Keys*

Key	Description	Example
`url`	(Required) Generated URL to follow	`url=http://exampleurl.com/`

Key	Description	Example
`description`	(Optional) Description to use in the user interface	**`description=URL with additional information`**

An output URL file and the output data object share the same base name along with the .url extension.

Here is an example where LIB1.OUT1.sas7bdat is the output data object, the data set is stored in the relative persistent area path, and the instance ID is 123456789.

`.../data/123456789/lib1/out1.url`

Here is an example of the file’s content:

```
url=http://someserver:someport/somepath
description=Take Me to Some Place
```

Data Collection

Here is an example for a collection as the output data object. In this example, the collection (named COLL1) is stored in the relative persistent area path and the instance ID is 123456789.

`.../data/123456789/coll1/`

`.../data/123456789/coll1/coll1.url`

Note: SAS Infrastructure for Risk Management creates collections folders with lowercase names in the file system.

Generic Libraries

In the case of a generic library, there is no storage on disk in the persistent area. Therefore, there is no library path that can be parsed to determine the path for the output URL text file. To provide access to the path, SAS Infrastructure for Risk Management scripting client provides the IRM_INSTANCE_FOLDER_ROOT macro variable IRM_INSTANCE_FOLDER_ROOT.

Here are examples for a generic library (GENLIB1) that points to a PostgreSQL database and for the task job flow instance data that is written to **`.../data/123456789`**:

- If the library is used alone (for example, GENLIB1), the task creates a folder under the instance root. The name of the library (in lowercase) is used as the output URL file name.

  `.../data/123456789/genlib1/genlib1.url`

- If the task output data objects are specific generic library tables (for example, GENLIB1.TABLE_A and GENLIB1.TABLE_B), the folder is still named after the library (in lowercase), but the URL file name is named after the table (in lowercase).

  `.../data/123456789/genlib1/table_a.url`

  `.../data/123456789/genlib1/table_b.url`

Parameter Substitutions

The task developer can specify substitution parameters as part of the URL record. When SAS Infrastructure for Risk Management scans the URL record, it replaces any substitution parameters with the appropriate values based on the calling context. The final URL is generated with all the substitutions returned.

Table 8.3 *Supported Substitution Parameters*

Parameter	Replacement
%instanceid	the job flow instance key for which the call was made
%et	the entity for the job flow for which the call was made
%userid	the user ID for the job flow for which the call was made
%bd	the base date of the job flow for which the call was made (in the format **`mmddyyyy`**)
%cs	the configuration set value for the job flow for which the call was made
%etr	the entity role for the job flow for which the call was made

Here is an example:

http://localhost:7090/someapp/fetch?id=%*instanceid*&entity=%*et*

Here is an example of the resulting URL, where the value for the **`%instanceid`** parameter is **`1004817`** and the value for the **`%be`** parameter is **`be`**.

http://localhost:7090/someapp/fetch?id=1004817&entity=be

Configuring and Viewing an Output URL

Overview

Here is an overview of how to configure and view an output URL:

1. Create a simple task that produces an output data object with an associated output URL.
2. Create a job flow that uses the task that you created.
3. View the generated job flow instance with an output URL in the user interface on page 67.

Step 1: Create a Simple Task That Produces an Output Data Object with an Associated Output URL

In this example, the output URL uses substitution parameters to build the URL associated with the WebDAV URL directory for the files of our job flow instances.

Here is a code sample that creates a task (word_sort_with_url.sas). When executed, it produces an output URL:

```
/**
   \file
   \brief The task creates output data with an associated url.
   \param[in] %IN_WORDS     word_parts dataset from landing area
   \param[out] %OUT_WORDS   sorted word_parts dataset
*/

proc sort data=&IN_WORDS out=&OUT_WORDS;
   by word;
run;

/*-- get path and name of output data object --*/
data _null_;
   call symput('url_filename',
      trim(PATHNAME(scan("&OUT_WORDS",1,'.')))||'/'||
      trim(scan(lowcase("&OUT_WORDS"),2,'.'))||'.url');
run;

/*-- create the output url file with 2 lines, url and description --*/

/*-- create the output url file with 2 lines, url and description --*/
data _null_;
   file "&url_filename";
   url_line = "url=http://fsnlax04.unx.sas.com:7980/SASIRMServer/" ||
              "irmwebdav/%userid/jobflows/%bd/%cs/%et/%etr/sas_sc";
   put url_line;
   put "description=WebDAV URL for My Job Flow Instance";
run;
```

Note: The base name of the output URL file must match (case and value) the base name of the output data object. In this example, the associated output URL file base name is created in lowercase because its associated output data object is a SAS data set. SAS data set member names are always written in lowercase on all file systems on which SAS runs.

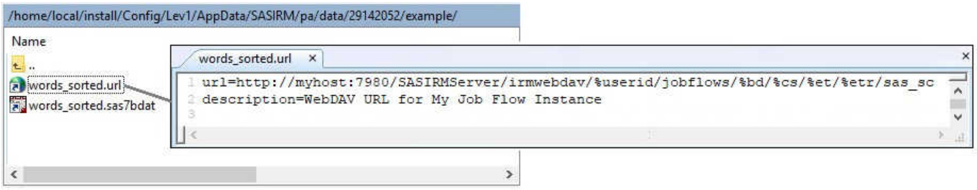
Step 2: Create a Job Flow That Uses the Task

Here is a scripting client job flow script example that creates a job flow (words_and_url_flow) that contains the task (word_sort_with_url):

```
/****************************************************************/
/* NAME:      word_sort_with_output_url_jobflow.sas            */
/* PURPOSE:   SAS Scripting Client program to create a job flow */
/*            that contains a task with an optional output url  */
/****************************************************************/
/* Scripting Client Initialization - Required setup */
%irm_sc_init();

/* Name the job flow and create a jobflow definition */
%irm_sc_build_jobflow(
          i_jf_name       =words_and_url_flow,
          o_jf_ref_name   =example_jf_ref );

/*Add Task that has the output url */
%word_sort_with_url(
          i_jf_ref        =&example_jf_ref,
          t_in_words      =staging.word_parts.sas7bdat,
          t_out_words     =example.words_sorted.sas7bdat);

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

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

When you run the job flow script, an instance of the job flow (words_and_url_flow) is created. The output of the instance — output data object (words_sorted.sas7bdat) and its associated output URL file (words_sorted.url) — is written to a persistent area instance folder (**`.../data/29142052/example`**).

Here is the content of the output URL file:



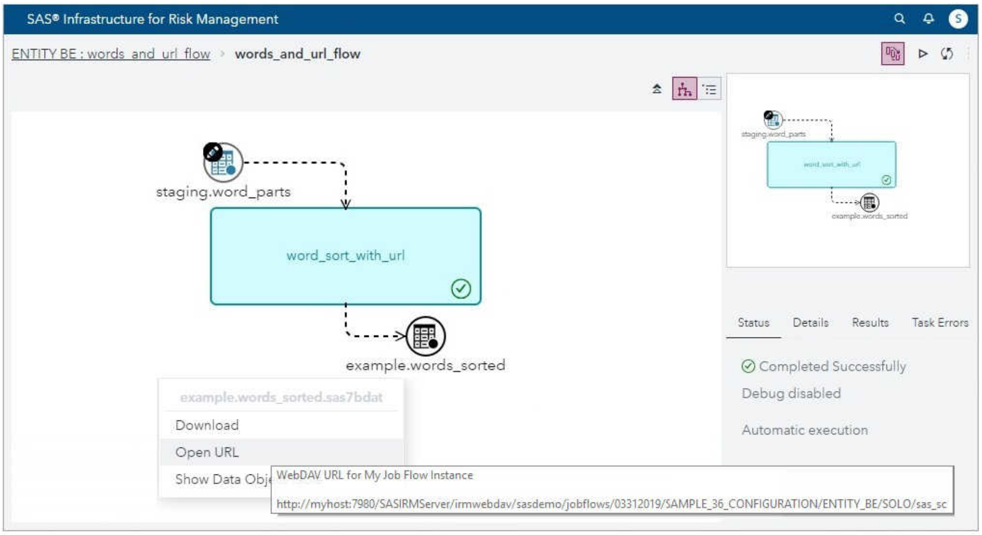
/home/local/install/Config/Lev1/AppData/SASIRM/pa/data/29142052/example/
Name
..
words_sorted.url
words_sorted.sas7bdat
words_sorted.url ×

```
url=http://myhost:7980/SASIRMServer/irmwebdav/%userid/jobflows/%bd/%cs/%et/%etr/sas_sc
description=WebDAV URL for My Job Flow Instance
```

Step 3: View the Generated Job Flow Instance with Output URL in the SAS Infrastructure for Risk Management Web Application

If an associated output URL exists for the output data object, the **Open URL** option is available in the pop-up menu for the output data object.

Here is the job flow instance created in this example:

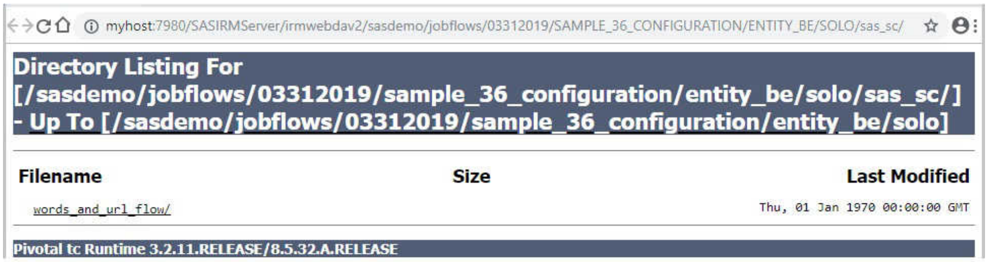


Completed Successfully
Debug disabled
Automatic execution

WebDAV URL for My Job Flow Instance
http://myhost:7980/SASIRMServer/irmwebdav/sasdemo/jobflows/03312019/SAMPLE_36_CONFIGURATION/ENTITY_BE/SOLO/sas_sc

Notice that the pop-up menu for the output data object (example.words_sorted) displays the **Open URL** option.

In this example, the associated URL is the WebDAV URL directory for the job flow instance files:



myhost:7980/SASIRMServer/irmwebdav2/sasdemo/jobflows/03312019/SAMPLE_36_CONFIGURATION/ENTITY_BE/SOLO/sas_sc/

Directory Listing For [/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo/sas_sc/] - Up To [/sasdemo/jobflows/03312019/sample_36_configuration/entity_be/solo]

Filename	Size	Last Modified
words_and_url_flow/		Thu, 01 Jan 1970 00:00:00 GMT

Pivotal tc Runtime 3.2.11.RELEASE/8.5.32.A.RELEASE

Appendix 1
Frequently Asked Questions

Can I include LIBNAME statements in task code?

In general, it is not acceptable to have LIBNAME statements in task code. If you have a LIBNAME statement in your task code, you are probably doing something wrong.

Fundamental to SAS Infrastructure for Risk Management design is the full disclosure of your immutable inputs and outputs for your tasks. A LIBNAME statement in your task code is a way to read input data or write output data without revealing information about the data to SAS Infrastructure for Risk Management and its users.

Issues that occur with the lack of full disclosure of the task data inputs and outputs include the following:

- It compromises the traceability and self-documentation of your job flows. For example, hidden data inputs and outputs might be considered a significant breach in regulatory solutions.
- It might compromise the correctness of the computations. SAS Infrastructure for Risk Management is dependent on the inputs and outputs that you declare in your tasks. Therefore, the wrong information can lead to incorrect computations.
- It might lead to unpredictable behavior that can break the transactional integrity of job flow instances. This is especially true if you declare references to SAS Infrastructure for Risk Management's reserved areas (landing area [%LA], input area [%IA], and the persistent area [%PA]). Declaring libraries that use the WebDAV engine to access the outputs of the job flow instances can also lead to unpredictable behavior.

SAS Infrastructure for Risk Management owns its LIBNAME assignments. These LIBNAME assignments are made in the libnames.txt files, which are located in the federated area config folder.

For more information about working with libraries, see *SAS Infrastructure for Risk Management: Administrator's Guide*.

For more information about the %irm_sc_init macro parameter options, see Appendix 2, "Scripting Client Macro Reference".

How can I share the content that I have developed in my personal federated area?

After developing and testing your content in your personal federated area, you can make you content available to others by creating a ZIP file of your personal federated area. After creating a ZIP file, you send a request to your system administrator to install your content as a stand-alone federated area.

For more information, see “Sharing the Content in Your Personal Federated Area”.

Error: Server is not in development mode

If your SAS Infrastructure for Risk Management server dev mode property is not set to true, you will get the following error message in SAS Studio when running the scripting client macros:

```
-------------------------------------------------------------
 ERROR: Server is not in development mode ! Property
 com.sas.solutions.risk.irm.server.devmode not set to true
-------------------------------------------------------------
```

If you receive this error, contact your system administrator and request to have the SAS Infrastructure for Risk Management server property com.sas.solutions.risk.irm.server.devmode set to true.

Error: An I/O error has occurred on file RESULTS

The %irm_sc_show_results macro creates the RESULTS libref using the SAS WebDAV access engine. This usage requires that you store your password in your user account properties metadata.

If the password is not stored in user account properties, you will see this error:

```
NOTE:  Credential could not be obtained from SAS metadata server.
 ERROR: An I/O error has occurred on file RESULTS.
 ERROR: Error in the LIBNAME statement.
```

If you receive this error, contact your system administrator and request to that your password be stored in your user account properties.

Error: An I/O error has occurred on file RESULTS

The error message is displayed under the following conditions:

- This message is displayed in the log when %irm_sc_show_jobflow_results macro encounters an error when assigning the RESULTS libref to the WebDAV RESULTS directory.
- This error can be caused when the job flow instance RESULTS directory does not exist.

  The message displayed in the log is:

  ```
  NOTE: Credential obtained from SAS metadata server.
       ERROR: Library RESULTS does not exist.
       ERROR: Error in the LIBNAME statement.
  ```

  The possible reasons that the library does not exist might be that the job flow does not produce results or that the job flow execution terminated prior to producing results.
- This error can also be caused when the LIBNAME statement cannot access the WebDAV server.

  The message displayed in the log is:

  ```
  NOTE: Credential obtained from SAS metadata server.
       ERROR: An I/O error has occurred on file RESULTS.
       ERROR: Error in the LIBNAME statement.
  ```

A possible reason that the LIBNAME statement cannot access the WebDAV server is that your user name and user ID are not the same for your SAS Metadata Server user account properties.

Why am I getting the warning “Apparent symbolic reference IRM_USER_ID not resolved” in the log when I run a task that connects to an external application like SAS Visual Analytics?

Your task needs the proper credentials to connect to an external application. When the task executes and tries to connect to an external application without the proper credentials, the following warning is generated in the task execution log:

```
WARNING: Apparent symbolic reference IRM_USER_ID not resolved.
WARNING: Apparent symbolic reference IRM_USER_PASSWORD not resolved.
```

If you receive this warning, your task needs a one-time password generated to connect to the external application. To do this, add the following parameter to the task header:

```
\param[in] %IRM_NUM_OF_ONE_TIME_PASSWORD=1
```

The parameter IRM_NUM_OF_ONE_TIME_PASSWORD tags your task to indicate that a one-time password is needed for task execution.

Appendix 2
Scripting Client Reference

About the SAS Scripting Client Reference

This reference provides a complete list of the SAS Infrastructure for Risk Management SAS scripting client macros that you can use in job flow scripts to implement features in your SAS programs.

In addition, this section provides instructions on how to access reference information for the SAS Infrastructure for Risk Management sample content. This documentation is provided in the SAS Infrastructure for Risk Management content.

SAS Infrastructure for Risk Management: Sample Basic Flows Reference Manual

For additional information about the SAS Infrastructure for Risk Management SAS scripting client, see the *SAS Infrastructure for Risk Management: Sample Basic Reference Manual* that included in the SAS Infrastructure for Risk Management content.

This content is automatically installed when SAS Infrastructure for Risk Management is installed.

This reference documentation provides information about the sample SAS flows and job flow scripts that are available, and how to use them.

To access this reference documentation, complete the following steps:

1. Log on to the SAS Infrastructure for Risk Management web application.
2. Open a job flow instance, right-click on a SAS task and select **Show Help**.

Details about that SAS task are displayed.

3. To view the reference documentation, click **Main** in the menu bar.

%irm_sc_add_flow_from_fa

Adds a reusable subflow to a job flow definition by its name and category ID. The reusable subflow must be created by theSAS Infrastructure for Risk Management scripting client.

Syntax

```
%irm_sc_add_flow_from_fa(
                i_jf_ref               =,
                i_sub_jf_name          =,
                i_sub_jf_category_id   =,
                i_is_in_subflow_folder  =);
```

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition, preceded by an ampersand (&).
i_sub_jf_name	name of the reusable subflow.
i_sub_jf_category_id	(Optional) category ID of the reusable subflow. If you do not specify a value, the scripting client uses the category ID from %irm_sc_init.
i_is_in_subflow_folder	(Optional) Boolean flag that indicates whether the reusable subflow is located in the subflow folder. The default is TRUE. The subflow is located in the subflow folder.

%irm_sc_add_partition_byn_task

Adds a By Number (ByN) method partition task to the job flow.

Syntax

```
%irm_sc_add_partition_byn_task(
                i_jf_ref              =,
                i_task_name           =,
                o_task_ref_name       =,
                i_card_tb_byn         =,
                i_part_ds_byn         =,
                i_out_part_tb_byn     =;)
```

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition to which the task is being added, preceded by an ampersand (&).
i_task_name	name of the task.
o_task_ref_name	(Optional) name of the reference to the task. The value of this parameter can be empty.
i_card_tb_byn	(Optional) name of the input cardinality table for the ByN method. If you do not specify a name, the default is used.
i_part_ds_byn	(Optional) name of the input data to be partitioned. If you do not specify a name, the default is used.
i_out_part_tb_byn	(Optional) name of the output partitioned table for ByN method. If you do not specify a name, the default is used.

%irm_sc_add_partition_byvar_task

Adds a By Variable (ByVar) method partition task to the job flow.

Syntax

```
%irm_sc_add_partition_byvar_task(
               i_jf_ref                =,
               i_task_name             =,
               o_task_ref_name         =,
               i_card_tb_byvar         =,
               i_part_ds_byvar         =,
               i_card_map_byvar        =,
               i_out_part_tb_byvar     =);
```

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition to which the task is to be added, preceded by an ampersand (&).
i_task_name	name of the task.

Parameter	Description
o_task_ref_name	(Optional) name of the reference to the task. The value of this parameter can be empty.
i_card_tb_byvar	(Optional) name of the input cardinality table for ByVar method. If you do not specify a name, the default is used.
i_part_ds_byvar	(Optional) name of the input data to be partitioned. If you do not specify a name, the default is used.
i_out_part_tb_byvar	(Optional) name of the output partitioned table for ByVar method. If you do not specify a name, the default is used.

%irm_sc_add_recombine_byn_task

Adds a task that recombines data that was partitioned using the ByN method.

Syntax

```
%irm_sc_add_recombine_byn_task(
               i_jf_ref              =,
               i_task_name           =,
               o_task_ref_name       =,
               i_rec_tb_in_byn       =,
               i_rec_out_tb_byn      =);
```

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition to which the task is to be added, preceded by an ampersand (&).
i_task_name	name of the task.
o_task_ref_name	(Optional) name of the reference to the task. The value of this parameter can be empty.
i_rec_tb_byn	(Optional) name of the partitioned input to be recombined. If you do not specify a name, the default is used.
i_rec_out_tb_byn	(Optional) name of the output for the recombined data. If you do not specify a name, the default is used.

%irm_sc_add_recombine_byvar_task

Creates a task that recombines data that was partitioned using the ByVar method.

Syntax

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

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition to which the task is to be added, preceded by an ampersand (&).
i_task_name	name of the task.
o_task_ref_name	(Optional) name of the reference to the task. The value of this parameter can be empty.
i_rec_tb_in_byvar	(Optional) name of the partitioned input to be recombined. If you do not specify a name, the default is used.
i_rec_card_map_byvar	(Optional) name of the cardinality mapping table for recombining the partitioned data. If you do not specify a name, the default is used.
i_rec_out_tb_byn	(Optional) name of the output for the recombined data. If you do not specify a name, the default is used.

%irm_sc_add_subflow

Creates and adds a subflow to a job flow.

Syntax

```
%irm_sc_add_subflow(
               i_jf_ref           =,
               i_sub_jf_ref       =);
```

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition to which the subflow is to be added, preceded by an ampersand (&).
i_sub_jf_ref	name of the reference to the subflow that is being added to a job flow, preceded by an ampersand (&).

%irm_sc_add_visual_data

Configures the data inputs and data outputs to be displayed in the SAS Infrastructure for Risk Management web application.

Syntax

```
%irm_sc_add_visual_data(
                i_jf_ref                     =,
                i_visual_input_data_list     =,
                i_visual_output_data_list    =);
```

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition, preceded by an ampersand (&).
i_visual_input_data_lis t	name of the visual input list. List entries are separated by a colon (:).
i_visual_output_data_l ist	name of the visual output data list. Its list entries are separated by a colon (:).

%irm_sc_build_jobflow

Names and builds a job flow definition.

Syntax

```
%irm_sc_build_jobflow(
                i_jf_name           =,
                i_jf_category_id    =,
                o_jf_ref_name       =);
```

Description

Parameter	Description
i_jf_name	name of the job flow (parallel program) that is created when you run the script. A valid value is a name that consists of up to 32 alphanumeric characters with no spaces.
i_jf_category_id	job flow category ID. If you do not specify a value, the scripting client uses the category ID defined in irm_sc_init.
o_jf_ref_name	name of the job flow definition as it appears in the SAS Infrastructure for Risk Management web application.

%irm_sc_create_jobflow_instance

Creates a job flow instance.

Syntax

```
%irm_sc_create_jobflow_instance(
               i_jf_ref                    =,
               i_jf_def_name               =,
               i_instance_name             =,
               i_jf_federated_area_id      =,
               i_jf_base_date              =,
               i_jf_category_id            =,
               i_jf_config_set_id          =,
               i_jf_entity_id              =,
               i_jf_entity_role_key        =,
               i_jf_entity_role            =,
               i_system_debug_on           =,
               i_auto_run_flag             =TRUE,
               i_debug_mode_flag           =FALSE,
               i_status_code_var_name      =_CREATE_JF_INST_STATUS,
               i_jf_instance_key_var_name  =INSTANCE_KEY);
```

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition, preceded by an ampersand (&). This variable is required if you do not specify a value for the i_jf_def_name.

Parameter	Description
i_jf_def_name	name of the job flow. This variable is required if you do not specify a value for i_jf_ref.
i_instance_name	(Optional) Name of the job flow instance. If you do not specify a value for this variable, the scripting client uses the name of the job flow instance’s job flow definition.
i_auto_run_flag	(Optional) Boolean flag that specifies whether to enable or disable automatic execution of the job flow instance. Valid values are TRUE (enable) and FALSE (disable). The default TRUE.
i_debug_mode_flag	(Optional) Boolean flag that specifies whether to enable or disable debug mode. Valid values are TRUE (enable) and FALSE (disable). The default FALSE.
i_jf_instance_key_var_name	(Optional) return job flow instance key variable name. If the call fails, the return value is null. By default, the value is INSTANCE_KEY.
i_jf_entity_role	(Optional) job flow entity role. Required only if entity role in irm_sc_init is set as both. Valid values are solo and group (case sensitive).
i_jf_entity_role_key	(Optional) job flow entity role key. If i_flow_entity_role is set, it overrides i_flow_entity_role_key.
i_jf_federated_area_id	(Optional) federated area ID in which the job flow definition is located. If you do not specify a value, the scripting client sets it as the personal federated area of the user that is logged in to SAS Studio.
i_jf_base_date	(Optional) job flow base date, entered in the format MMDDYYYY (for example, for a base date of March 31, 2018, you enter the value as 03312018). If you do not specify a value, the scripting client obtains a value from %irm_sc_init.
i_jf_category_id	(Optional) job flow category ID. If you do not specify a value, the scripting client obtains a value from %irm_sc_init.
i_jf_config_set_id	(Optional) job flow config set ID. If you do not specify a value, the scripting client obtains a value from %irm_sc_init.
i_jf_entity_id	(Optional) job flow entity ID. If you do not specify a value, the scripting client obtains a value from %irm_sc_init.
i_system_debug_on	(Optional) Boolean flag that enables or disables verbose debug messages. Valid values are TRUE (enable) or FALSE (disable). If you do not specify a value, the scripting client obtains a value from %irm_sc_init.
i_status_code_var_name	(Optional) returns the status code variable name. If executed successfully, 200 is returned. The default is _CREATE_JF_INST_STATUS.

%irm_sc_del_flow_def

SAS Scripting Client deletes the job flow definition in a personal federated area. The job flow definition must be created by a scripting client.

Syntax

```
%irm_sc_del_flow_def(
               i_jf_name              =,
               i_jf_category_id       =,
               i_status_code_var_name  = );
```

Description

Parameter	Description
i_jf_name	specifies the name of the job flow definition.
i_jf_category_id	(Optional) job flow category ID. If you do not specify a value, the scripting client obtains the category ID from %irm_sc_init.
i_status_code_var_name	(Optional) returns the status code variable name. If executed successfully, 200 is returned.

%irm_sc_execute_jobflow

Executes job flow instance by job flow reference.

Syntax

```
%irm_sc_execute_jobflow(
               i_jf_ref                    =,
               i_jf_instance_key           =,
               i_jf_federated_area_id      =,
               i_jf_base_date              =,
               i_jf_category_id            =,
               i_jf_config_set_id          =,
               i_jf_entity_id              =,
               i_jf_entity_role_key        =,
               i_jf_entity_role            =,
               i_system_debug_on           =,
               i_status_code_var_name      =_EXECUTE_JF_INST_STATUS,
               i_jf_instance_key_var_name  =INSTANCE_KEY);
```

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition, preceded by an ampersand (&).
i_jf_instance_key	instance key of the job flow. This value is required if you do not specify a value for i_jf_ref.
i_jf_instance_key_var_name	(Optional) return job flow instance key variable name. If the call fails, the return value is null. By default it is INSTANCE_KEY.
i_jf_entity_role	(Optional) job flow entity role. A value for this variable is required if the entity role in irm_sc_init is set as both. Valid values are solo and group (case sensitive).
i_jf_entity_role_key	(Optional) job flow entity role key. If i_flow_entity_role is set, it overrides i_flow_entity_role_key. However, if you provide a value for the job flow instance, any input is ignored.
i_jf_federated_area_id	(Optional) federated area ID in which job flow definition is located. If you do not specify a value for this variable, the scripting client sets it as the personal federated area of the user that is logged in to SAS Studio. However, if you do specify a value, any input is ignored.
i_jf_base_date	(Optional) job flow base date, entered in the format MMDDYYYY (for example, for a base date of March 31, 2018, you enter the value as 03312018). If you do not specify a value, the scripting client obtains a value from %irm_sc_init. However, if you provide a value for the i_jf_instance_key, any input is ignored.
i_jf_category_id	(Optional) job flow category. If you do not specify a value, the scripting client obtains a value from % irm_sc_init. However, if you provide a value for i_jf_instance_key, any input is ignored.
i_jf_config_set_id	(Optional) job flow config set ID. If you do not specify a value, the scripting client obtains a value from % irm_sc_init. However, if you provide a value for i_jf_instance_key, any input is ignored.
i_jf_entity_id	(Optional) Job flow entity ID. If you do not specify a value, the scripting client obtains a value from % irm_sc_init. However, if you provide a value for i_jf_instance_key, any input is ignored.
i_system_debug_on	(Optional) (Optional) Boolean flag that enables or disables verbose debug messages. Valid values are TRUE (enable) or FALSE (disable). If you do not specify a value, the scripting client obtains a value from %irm_sc_init.
i_status_code_var_name	(Optional) returns the status code variable name. If executed successfully, 200 is returned. The default is _EXECUTE_JF_INST_STATUS.

%irm_sc_gen_doc

SAS scripting client generates documentation for the personal federated area in the SAS Infrastructure for Risk Management.

Syntax

```
%irm_sc_gen_doc(i_status_code_var_name=);
```

Description

Parameter	Description
i_status_code_var_name	(Optional) returns the status code variable name. If executed successfully, 200 is returned.

%irm_sc_get_instance_property

Retrieves the properties of a job flow instance.

Syntax

```
%irm_sc_get_instance_property(
              i_jf_instance_key         =,
              i_inst_property_list      =,
              i_variable_name_list      =,
              i_status_code_var_name    =_GET_INST_INFO_STATUS);
```

Description

Parameter	Description
i_jf_instance_key	job flow instance key. If the instance key is invalid, the returned properties value for autoRun, debug, name, description are true, false, an empty string, and nil.
i_inst_property_list	list of instance properties (case sensitive) separated by colon (:). Supported properties: autoRun, debug, name, and description.
i_variable_name_list	list of returned variable names, separated by colon (:).
i_status_code_var_name	(Optional) returns the status code variable name. If executed successfully, 200 is returned. The default is _GET_INST_INFO_STATUS.

Example

```
%irm_sc_get_instance_property(
              i_jf_instance_key=1311334918,
              i_inst_property_list=name:autoRun:description,
              i_variable_name_list=instName:currentAutoRunSetting:InstDescription);

%put "Instance Name:" &instName;
%put "Auto Run Setting:" &currentAutoRunSetting;
%put "Instance Description:" &InstDescription;
```

%irm_sc_init

Initializes the SAS Infrastructure for Risk Management scripting client. When initialized, the scripting client queries the metadata server configuration. In addition, the scripting client retrieves the URL of the SAS Infrastructure for Risk Management server, port, and the path to the user's personal federate area.

Syntax

```
%irm_sc_init(
              i_jf_entity_role            =,
              i_jf_base_date              =03312018,
              i_jf_category_id            =sas_sc,
              i_jf_config_set_id          =SAMPLE_35_CONFIGURATION,
              i_jf_entity_id              =ENTITY_BE,
              i_jf_entity_role_key        =1,
              i_system_debug_on           =FALSE,
              i_system_flow_size_limit    =50,
              i_connection_type           =INTERNAL);
```

Description

Parameter	Description
i_jf_base_date	(Optional) job flow base date, entered in the format MMDDYYYY (for example, for a base date of March 31, 2018, you enter the value as 03312018).
i_jf_category_id	(Optional) job flow category
i_jf_config_set_id	(Optional) job flow config set ID
i_jf_entity_id	(Optional) job flow entity ID
i_jf_entity_role	(Optional) job flow entity role. Valid values are solo, group, or both (case sensitive).

Parameter	Description
i_jf_entity_role_key	(Optional) job flow entity role key. If you specify a value for i_flow_entity_role, that value overrides the value specified for i_flow_entity_role_key.
i_system_debug_on	(Optional) Boolean flag that enables or disables verbose debug messages. Valid values are TRUE (enable) or FALSE (disable). If you do not specify a value, the scripting client obtains a value from %irm_sc_init.
i_system_flow_size_li mit	(Optional) specifies the job flow JSON file size limit, in megabytes. The default is 50MB.
i_connection_type	(Optional) type of connection to IRM server. Available values are INTERNAL and EXTERNAL. The default value is INTERNAL.
i_system_flow_size_li mit	specifies the job flow JSON file size limit, in megabytes. The default is 50MB.

%irm_sc_save_jobflow

Saves the job flow definition to the SAS Infrastructure for Risk Management server. When overwriting an existing job flow definition, executing this macro deletes all job flow instances associated with the definition that you are overwriting.

Syntax

```
%irm_sc_save_jobflow(
              i_jf_ref                        =,
              i_status_code_var_name          =,
              i_should_use_subflow_folder     =TRUE);
```

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition, preceded by an ampersand (&).
i_should_use_subflow _folder	a Boolean flag to specify whether the job flow should be saved to the subflow folder. Valid values are TRUE and FALSE. The default is FALSE.
i_status_code_var_na me	(Optional) returns the status code variable name. If the job flow is executed successfully, 200 is returned.

%irm_sc_set_instance_property

Sets properties for a job flow instance.

Syntax

```
%irm_sc_set_instance_property(
              i_jf_instance_key         =,
              i_inst_property_list      =,
              i_value_list              =,
              i_status_code_var_name    =_SET_INST_INFO_STATUS);
```

Description

Parameter	Description
i_jf_instance_key	job flow instance key.
i_inst_property_list	list of instance properties (case sensitive) separated by colon (:). Supported properties: autoRun, debug, name, and description.
i_value_list	list of corresponding property values, separated by colon (:).
i_status_code_var_name	(Optional) returns the status code variable name. If the job flow is executed successfully, 200 is returned. The default is _SET_INST_INFO_STATUS.

%irm_sc_show_jobflow_results

This macro is run after the %irm_sc_execute_jobflow macro in a scripting client script. It references the job flow instance that was created by %irm_sc_execute_jobflow macro. This macro creates a libref to the WebDAV results location for the generated job flow instance. It also creates a file shortcut (IRM_FLOW) link to the job flow instance in SAS Infrastructure for Risk Management web application. You can execute this macro only as part of a scripting client job flow script. In addition, you must run it after the job flow instance is created by the %irm_sc_execute_jobflow macro.

Syntax

```
%irm_sc_show_jobflow_results(
              i_jf_ref   =  );
```

Description

Parameter	Description
i_jf_ref	a name that references the job flow definition, preceded by an ampersand (&).

Recommended Reading

- *SAS Infrastructure for Risk Management: Administrator's Guide*
- *SAS Infrastructure for Risk Management: User's Guide*
- *SAS Studio 3.7: User's Guide*

For a complete list of SAS publications, go to support.sas.com/en/books.html. If you have questions about which titles you need, please contact a SAS Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-0025
Fax: 1-919-677-4444
Email: sasbook@sas.com
Web address: support.sas.com/en/books.html