SAS® Drug Development 3.5
Macros
User's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2008. *SAS® Drug Development 3.5: Macros User's Guide*. (Second printing) Cary, NC: SAS Institute Inc.

SAS® Drug Development 3.5: Macros User's Guide

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, April 2008

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Contents

CHAPTER 1

Introduction

Audience

This guide is intended for users who want to develop applications with the SAS Drug Development macros.

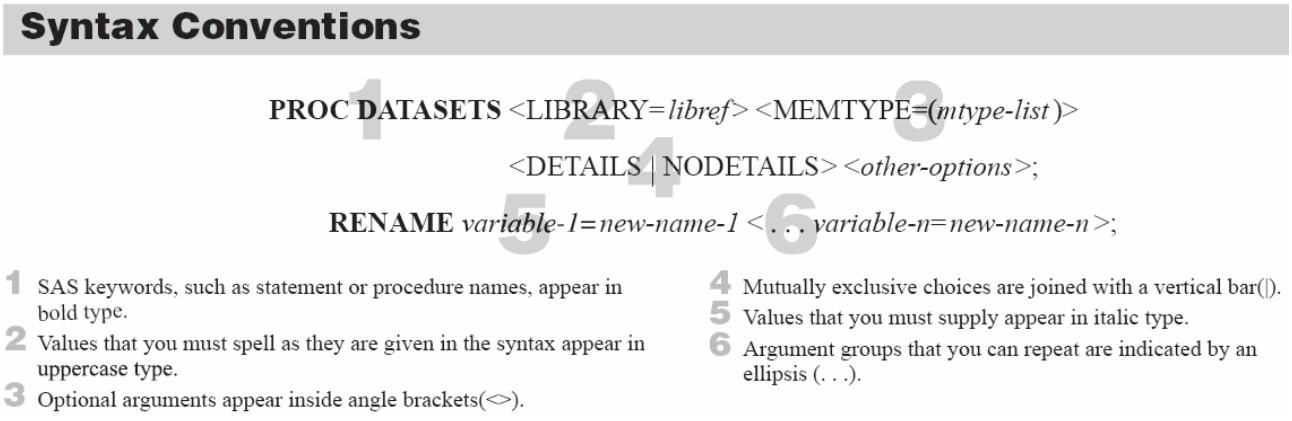
You must be familiar with SAS Drug Development functionality, such as type definitions, containers, files, and access permissions. For reference information on SAS Drug Development functionality, see the SAS Drug Development online Help and user's guide.

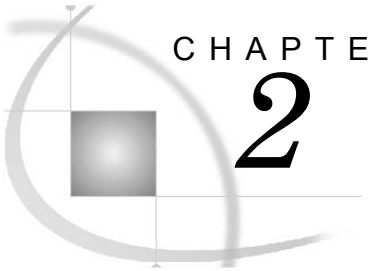
Typographic Conventions Used in This Guide

Throughout this document you'll see the following typographic conventions:

Convention	Description
`monospace` font	denotes code, such as a code example
`monospace bold` font	denotes text that you type, such as an object name
`monospace bold italics` font	denotes a value that you specify, such as your name

The following graphic explains the syntax for SAS code used in this document:





CHAPTER 2

Installing the Macros

Overview

This document describes how to install the SAS Drug Development macros, which are distributed in the file SASDrugDevRemoteAPI_Macros.zip.

Requirements

The following software is required by the SAS Drug Development macros:

- SAS 9.2
- Java Runtime Environment Version 1.5.0*
- SAS Drug Development remote API client

Installation Instructions for Microsoft Windows

1 Unzip the contents of SASDrugDevRemoteAPI_Macros.zip to C:\.

This creates the following files and folder in C:\:

- SASDrugDevRemoteAPI_Macros\sddmacros.cfg
- SASDrugDevRemoteAPI_Macros\sasv9_local.cfg
- SASDrugDevRemoteAPI_Macros\version.txt
- SASDrugDevRemoteAPI_Macros\lib\sas-drugdev-sasmacros.jar
- SASDrugDevRemoteAPI_Macros\lib\log4j.properties
- SASDrugDevRemoteAPI_Macros\sasmacro

  This is the folder that contains the SAS Drug Development macros as .sas files.

2 Review the properties of the shortcut that starts SAS to ensure that the shortcut includes the option that points to the configuration file

```
!sasroot\sasv9.cfg
```

For example:

```
-CONFIG "C:\Program Files\SAS\SASFoundation\9.2\sasv9.cfg"
```

Note: The default location for **SASROOT** is `C:\Program`

3 Edit the file `!sasroot\sasv9.cfg` by adding the following line:

```
-CONFIG "C:\SASDrugDevRemoteAPI_Macros\sddmacros.cfg"
```

4 If the SAS Drug Development remote API client is installed in a location other than `C:\SASDrugDevRemoteAPI`, you must edit the file sddmacros.cfg as follows:

Caution: Back up this file before you edit it. Use extreme care when editing this file. If you have any questions, concerns, or problems, contact your on-site SAS support personnel.

a. Modify the -JREOPTIONS statement by editing the option -Dsas.app.class.dirs to include the absolute path to the `lib` folder in the SAS Drug Development remote API client installation and

   `C:\SASDrugDevRemoteAPI_Macros\lib.`

b. If you are connecting to an instance of SAS Drug Development that is hosted by SAS, and you are behind a firewall and proxy servers, you might need to address proxy requirements or restrictions.

   If HTTPS traffic is proxied, specify the following Java system properties in the -JREOPTIONS statement to configure the Java runtime environment:

   `-Dhttps.proxyHost=<`*`proxy-host-name`*`> -Dhttps.proxyPort=<`*`port-number`*`>`

Caution: Modify only these options. Ensure that you do not insert any carriage returns in the `sas.app.class.dirs` option.

The -JREOPTIONS statement (when using a proxy server) will look similar to the following when you are finished:

```
-JREOPTIONS=(
-Dsas.app.class.dirs=C:\RemoteAPI\lib;C:\SASDrugDevRemoteAPI_Macros\lib
-Dsas.javaobj.experimental=no
-Dhttps.proxyHost=yourProxyServer.sas.com
-Dhttps.proxyPort=8080)
```

5 Edit the file `!sasroot\nls\language\sasv9.cfg` (where language is the language used by SAS, such as en for English) to change the -SET SASAUTOS statement to include the folder `C:\SASDrugDevRemoteAPI_Macros\sasmacro`.

For example:

```
-SET SASAUTOS ("C:\SASDrugDevRemoteAPI_Macros\sasmacro")
```

Installation Instructions for UNIX

1. Create a directory under **!SASROOT** called **RemoteAPI**. For the purposes of these instructions, !SASROOT will be referred to as

   `/apps/sas9.2/SASFoundation/9.2.`

2. Unzip the contents of SASDrugDevRemoteAPI_Macros.zip to **`/apps/sas9.2/SASFoundation/9.2/RemoteAPI/.`**

   This creates the following files and folder in

   `/apps/sas9.2/SASFoundation/9.2/RemoteAPI/`:

   - **`SASDrugDevRemoteAPI_Macros/sddmacros.cfg`**
   - **`SASDrugDevRemoteAPI_Macros/sasv9_local.cfg`**
   - **`SASDrugDevRemoteAPI_Macros/version.txt`**
   - **`SASDrugDevRemoteAPI_Macros/lib/sas-drugdev-sasmacros.jar`**
   - **`SASDrugDevRemoteAPI_Macros/lib/log4j.properties`**
   - **`SASDrugDevRemoteAPI_Macros/sasmacro`**

     This is the folder that contains the SAS Drug Development macros as .sasfiles.

3. To install the SAS Drug Development remote API client, unzip the contents of SASDrugDevRemoteAPI.zip to a temp area on the UNIX server. This will create a SASDrugDevRemoteAPI directory. Copy all the jars in SASDrugDevRemoteAPI/lib to

   `/apps/sas9.2/SASFoundation/9.2/RemoteAPI/SASDrugDevRemoteAPI_Macros/lib.`

4. Modify the access permissions of the! SASROOT/RemoteAPI directory recursively to match the access permissions of the user ID and group that installed and runs SAS.

   For example: chown –R sasadmin:sasadmin

   `/apps/sas9.2/SASFoundation/9.2/RemoteAPI`

5. Edit the file **`!SASROOT/sasv9_local.cfg`**::

   Caution: Back up this file before you edit it. Use extreme care when editing this file. If you have any questions, concerns, or problems, contact your on-site SAS support personnel.

   Add in the JREOPTIONS statement so that it looks like the following example:

   ```
   -JREOPTIONS=(
   -Dsas.app.class.dirs=/apps/sas9.2/SASFoundation/9.2/RemoteAPI/SASDrugD
   evRemoteAPI_Macros/lib
   -Dsas.javaobj.experimental=no
   )
   ```

   If you are connecting to an instance of SAS Drug Development that is hosted by SAS, and you are behind a firewall and proxy servers, you might need to address proxy requirements or restrictions.

   If HTTPS traffic is proxied, specify the following Java system properties in the -JREOPTIONS statement to configure the Java runtime environment:

   `-Dhttps.proxyHost=<`*`proxy-host-name`*`> -Dhttps.proxyPort=<`*`port-number`*`>`

Caution: Modify only these options. Ensure that you do not insert any carriage returns in the sas.app.class.dirs option.

The -JREOPTIONS statement will look similar to the following when you are finished:

```
-JREOPTIONS=(
-Dsas.app.class.dirs=/apps/sas9.2/SASFoundation/9.2/RemoteAPI/SASDrugD
evRemoteAPI_Macros/lib
-Dsas.javaobj.experimental=no
-Dhttps.proxyHost=yourProxyServer.sas.com
-Dhttps.proxyPort=8080)
```

6 Edit the file! SASROOT**/sasv9.cfg** to change or add the -SASAUTOS statement to include the path to the SAS Drug Development macros.

For example:

```
-sasautos ('!SASROOT/sasautos'
'/apps/sas9.2/SASFoundation/9.2/RemoteAPI/SASDrugDevRemoteAPI_Macros/s
asmacro')
```

Note: You might need to enclose existing SASAUTOS settings in quotation marks.

Verifying the Installation

The following SAS code displays the settings for the -JREOPTIONS and verifies that the JRE is configured properly. It does not verify that the SAS Drug Development macros are installed properly.

```
proc javainfo; run;
```

The following SAS code verifies that the SAS Drug Development macros are installed. Replace ***sdd-instance***, ***sdd-user-ID***, and ***sdd-password*** with values that reflect your instance of SAS Drug Development.

```
options mprint;
proc javainfo;
run;

%sasdrugdev_login(url=%str(https://sddinstance/
sddremote),sdduserid=%str(sdd-user-ID),sddpassword=%str(sddpassword));

/* List the contents of the root folder in SDD */
%sasdrugdev_getobjects(sddpath=/SDD);

proc print;
  title "List of Objects in /SDD";
run;

%sasdrugdev_logout;
```

The code should generate a list of the contents of the root folder in the SAS Drug Development repository.

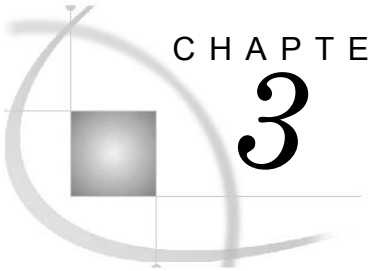
Note: By default, the root folder in the SAS Drug Development repository is **/SDD**. If the root of your SAS Drug Development repository is different, change **/SDD** in the code above.

The SAS log will contain information that might be useful for debugging the installation of the SAS Drug Development macros.

If you have problems connecting to SAS Drug Development, add the debugging parameters to **%sasdrugdev_login()**: DEBUGLOG and DEBUGLEVEL.

For example:

```
%sasdrugdev_login(url=%str(https://sddinstance/
sddremote),sdduserid=%str(sdd-user-ID),sddpassword=%str(sddpassword),
debuglog=absolute-path-to-log,debuglevel=DEBUG)
```



CHAPTER 3

SAS Drug Development Macros

Introduction

The SAS Drug Development macros enable you to use familiar SAS syntax to make calls to SAS Drug Development. Your SAS programming skills and these macros enable you to perform certain operations.

From a programming perspective, the folders and files in the SAS Drug Development repository are containers and files. The names of the macros use this naming convention. However, in the SAS Drug Development application, “containers” are “container objects” and “files” are “content objects.” Therefore, this chapter uses the terms “container object” and “content object.”

Macro Return Codes

After you execute a macro, the global variable _SDDRC_ contains the return code, and the global variable _SDDMSG_ contains any additional information.

Here are the specific values that can be returned by a macro:

SDDRC Value	Explanation
-99	The result is uninitialized
-23	No user group members were specified to be updated
-22	An invalid parameter was passed
-21	No system policies were specified to be updated
-20	No value was specified for the property

SDDRC Value	Explanation
-19	The property format is invalid
-18	The login credentials are invalid
-17	The user account is retired
-16	The user account is inactive
-15	The password has expired
-14	The user account could not be authenticated
-13	The URL is malformed
-12	The property does not exist for the object
-11	The system policy does not exist
-10	The user account does not exist
-9	The user group does not exist
-8	An unexpected error was encountered
-7	The property does not exist
-6	A protected system property
-5	Invalid access permissions were encountered
-4	The object does not exist
-3	The object type is invalid for this macro
-2	No valid session exists
-1	The session is no longer valid
0	The macro executed without error
1	The object exists
2	No properties were updated

Note: A macro always returns a code, but a macro does not necessarily return every one of these codes.

Macros and System Policies

The SAS Drug Development system policies control the macros that are functional for you. For example, to change a user account with a macro, you must have the system policy **User can manage user accounts**.

If you call a macro without the required system policy, the macro fails. The macro returns a code and an error message.

For more information about the system policies, see the online Help.

The Macros

Using the Ampersand Character (&) in URLs

For a macro with a parameter that specifies a URL, such as a macro that sets properties, you cannot embed the ampersand character (&) in the URL. The ampersand character is a special character in SAS. If you embed an ampersand character, SAS will attempt to resolve the subsequent text as a macro variable.

The Proper Case for Parameter Values

Although SAS is case insensitive, the parameter values passed by the SAS Drug Development macros are case sensitive. This chapter presents the parameter values in the case that they must be passed to the macros.

SASDRUGDEV_CONTAINEREXISTS

Description

Determines whether a container object exists.

Syntax

%SASDRUGDEV_CONTAINEREXISTS(SDDPATH=*sdd-path*);

sdd-path is the path (starting at the root) and name of the container object in the SAS Drug Development repository.

SASDRUGDEV_COPYCONTAINER

Description

Copies a container object (and the objects it contains) from one location in the SAS Drug Development repository to another location in the repository.

Syntax

%SASDRUGDEV_COPYCONTAINER(SRCPATH=*source-path*, DESTPATH=*destination-path*);

source-path is the path (starting at the root) and name of the container object in the SAS Drug Development repository to copy.

destination-path is the path (starting at the root) and name of the new container object in the SAS Drug Development repository to which to copy the container object.

SASDRUGDEV_COPYFILE

Description

Copies a content object from one location in the SAS Drug Development repository to another location in the repository.

Syntax

%SASDRUGDEV_COPYFILE(SRCPATH=source-path, DESTPATH=destinationpath);

source-path is the path (starting at the root) and name of the content object in the SAS Drug Development repository to copy.

destination-path is the path (starting at the root) and name of the new content object in the SAS Drug Development repository to which to copy the content object.

SASDRUGDEV_CREATEGROUP

Description

Creates a user group in SAS Drug Development.

Syntax

%SASDRUGDEV_CREATEGROUP(SDDGROUPNAME=*name*, SDDGROUPDESC=*description*);

name is the name of the user group.

description is the description of the user group.

SASDRUGDEV_CREATELOCALFILE

Description

Copies a content object from the SAS Drug Development repository to the computer on which SAS is executing.

Syntax

%SASDRUGDEV_CREATELOCALFILE(LOCALPATH=*local-path*, SDDPATH=*sdd-path*);

local-path is the absolute path and name of the file to create on the local computer on which SAS is executing.

sdd-path is the path (starting at the root) and name of the content object in the SAS Drug Development repository to copy.

SASDRUGDEV_CREATESDDCONTAINER

Description

Creates a container object in the SAS Drug Development repository. All parent folders in the path that do not exist will be created.

Syntax

%SASDRUGDEV_CREATESDDCONTAINER(SDDPATH=*sdd-path*) <, TYPE=*type*, VERBOSE=*verbose-level* >);

sdd-path is the path (starting at the root) and name of the container object to create in the SAS Drug Development repository.

type is the type of container object to create. These are the valid values:

- compound
- folder (default)
- indication
- protocol

verbose-level is the level of message detail to include in the SAS log file. These are the valid values:

- NONE
- INFO (default)
- ERROR

For information about the valid values for the type of container object, see Appendix 2, "Reference."

SASDRUGDEV_CREATESDDFILE

Description

Copies a file from the computer on which SAS is executing to the SAS Drug Development repository.

Syntax

%SASDRUGDEV_CREATESDDFILE(LOCALPATH=*local-path*, SDDPATH=*sddpath*<, TYPE=*type*>);

local-path is the absolute path and name of the file.

sdd-path is the path (starting at the root) and name of the content object in the SAS Drug Development repository to which to copy the file.

type is the type of content object to create.

For information about the valid values for the type of content object, see Appendix 2, "Reference."

SASDRUGDEV_CREATEUSER

Creates a user account in SAS Drug Development.

Syntax

%SASDRUGDEV_CREATEUSER(SDDUSERID=*user-ID*, SDDPASSWORD=*password*, SDDFIRSTNAME=*first-name*, SDDLASTNAME=*lastname*, SDDEMAIL=*email-address*);

user-ID is the ID for the user account.

Password is the password for the user account.

first-name is the first name of the user.

last-name is the last name of the user.

email-address is the e-mail address of the user.

SASDRUGDEV_DELETECONTAINER

Description

Deletes a container object from the SAS Drug Development repository.

Syntax

%SASDRUGDEV_DELETECONTAINER(SDDPATH=*sdd-path*);

sdd-path is the path (starting at the root) and name of the container object in theSAS Drug Development repository to delete.

SASDRUGDEV_DELETEFILE

Description

Deletes a content object from the SAS Drug Development repository.

Syntax

%SASDRUGDEV_DELETEFILE(SDDPATH=*sdd-path*);

sdd-path is the path (starting at the root) and name of the content object in the SAS Drug Development repository to delete.

SASDRUGDEV_DELETEGROUP

Description

Deletes a user group in SAS Drug Development.

Syntax

%SASDRUGDEV_DELETEGROUP(SDDGROUPNAME=*name*);

name is the name of the user group to delete.

SASDRUGDEV_FILEEXISTS

Description

Determines whether a content object exists.

Syntax

%SASDRUGDEV_FILEEXISTS(SDDPATH =*sdd-path*);

sdd-path is the path (starting at the root) and name of the content object in theSAS Drug Development repository.

SASDRUGDEV_GETCONTAINERPROPS

Description

Returns a SAS data set that contains the properties of a container object.

Syntax

%SASDRUGDEV_GETCONTAINERPROPS(SDDPATH=*sdd-path*
<, DSNAME=*SAS-data-set*>);

sdd-path is the path (starting at the root) and name of the container object in the SAS Drug Development repository.

SAS-data-set is the name of a SAS data set to create that will contain the properties of the container object. Specify *SAS-data-set* as *libref.dataset*. The default value is WORK._CONTAINERPROPS_.

The data set contains columns with these names:

- path, which is the path (starting at the root) and name of
- the container object in the SAS Drug Development
- repository
- name, which is the name of the property

  For information about the valid values for the name of the container object property, see Appendix 2, "Reference."
- value, which is the value to assign to the property

SASDRUGDEV_GETFILEPROPS

Description

Returns a SAS data set that contains the properties of a content object.

Syntax

%SASDRUGDEV_GETFILEPROPS(SDDPATH=*sdd-path*
<, DSNAME=*SAS-data-set*>);

sdd-path is the path (starting at the root) and name of the content object in the SAS Drug Development repository.

SAS-data-set is the name of a SAS data set to create that will contain the properties of the content object. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._FILEPROPS_**.

The data set contains columns with these names:

- path, which is the path (starting at the root) and name of the
- content object in the SAS Drug Development repository
- name, which is the name of the property

For information about the valid values for the name of the

- content object property, see Appendix 2, "Reference."
- value, which is the value to assign to the property

SASDRUGDEV_GETGROUPS

Description

Returns a SAS data set that contains the names and descriptions of the SAS Drug Development user groups.

Syntax

%SASDRUGDEV_GETGROUPS(<, DSNAME=*SAS-data-set*>);

SAS-data-set is the name of a SAS data set to create that will contain the groups. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._GETGROUPS_**.

The data set contains columns with these names:

- groupname, which is the name of the user group
- description, which is the description of the group

SASDRUGDEV_GETGROUPMEMBERS

Description

Returns a SAS data set that contains the members of a SAS Drug Development user group.

Syntax

%SASDRUGDEV_GETGROUPMEMBERS(SDDGROUPNAME=*name* <, DSNAME=*SAS-data-set*>);

name is the name of the user group.

SAS-data-set is the name of a SAS data set to create that will contain the members. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._GMEMBERS_**.

The data set contains columns with these names:

- groupname, which is the name of the user group
- userid, which is the ID for the user account
- action, which is blank

SASDRUGDEV_GETOBJECTS

Description

Returns a SAS data set that contains the metadata for all of the objects within a container object in the SAS Drug Development repository.

Syntax

%SASDRUGDEV_GETOBJECTS(SDDPATH=*sdd-path* <RECURSIVELEVEL=*recursion-level*, TYPEDEF=*object-type*, DSNAME=*SAS-data-set*>);

sdd-path is the path (starting at the root) and name of the container object in the SAS Drug Development repository.

recursion-level is the level of recursion, and valid values are 0, 1, or 2. Setting the parameter to 0 returns permission records for the input container only. Setting the parameter to 1 returns permission records for the container

and all objects within the container, but not sub containers. Setting the parameter to 2 returns permission records for the input container, objects within the container, and all sub containers and objects. The default value is 0.

typedef is the SDD file type to return, or * for all file types. The default value is '*'

SAS-data-set is the name of a SAS data set to create that will contain the metadata for the objects. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._GETOBJECTS_**.

NOTE: The previous version contained the parameter recursive=true/false and is available only for backward compatibility. The new parameter recursiveLevel should be used instead.

The data set contains a row for each object in the container object and columns with these names:

- name, which is the name of the object
- fullPathName, which is the fully qualified name of the object
- sizeString, which is the size of the object represented as a string
- type, which is the type of the SAS Drug Development object
- owner, which is the ID for the user account of the owner of the object
- dateTimeString, which is the date and time when the object was last modified The time is in GMT format and is represented as a string.
- modifiedBy, which is the ID for the user account of the user who last modified the object
- containingFolderName, which is the name of the folder containing the object. Same as fullPathName without the object name.
- isContainer, which is a numeric flag that indicates whether the object is a container object (0=no, 1=yes)
- dateModified, which is a SAS date field of the date the object was last modified.
- timeModified, which is a SAS time field of the time of day the object was modified.

SASDRUGDEV_GETPERMISSIONS

Description

Returns a SAS data set that contains the permissions metadata for all of the objects within a container object in the SAS Drug Development repository.

Syntax

%SASDRUGDEV_GETPERMISSIONS(SDDPATH=*sdd-path* <RECURSIVELEVEL=*recursion-level*, TYPEDEF=*object-type*, DSNAME=*SAS-data-set*>);

sdd-path is the path (starting at the root) and name of the container object in the SAS Drug Development repository.

recursion-level is the level of recursion, and valid values are 0, 1, or 2. Setting the parameter to 0 returns permission records for the input container only. Setting the parameter to 1 returns permission records for the container and all objects within the container, but not sub containers. Setting the parameter to 2 returns permission records for the input container, objects within the container, and all sub containers and objects. The default value is 0.

typedef is the string containing the SDD file type to return, or * for all file types. The default value is is '*'.

SAS-data-set is the name of a SAS data set to create that will contain the metadata for the objects. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._GETNODEPERMISSIONS_**.

The data set contains a row for each object in the container object and columns with these names:

- name, which is the name of the object
- fullPathName, which is the fully qualified name of the object
- sizeString, which is the size of the object represented as a string
- type, which is the type of the SAS Drug Development object
- owner, which is the ID for the user account of the owner of the object
- folder, which is the name of the folder containing the object. Same as fullPathName without the object name.
- isContainer, which is a numeric flag that indicates whether the object is a container object (0=no, 1=yes)
- ACEName, which is the name of the access control entry represented
- canRead --- which is a numeric flag that indicates whether the ACEName user or group has read access(0=no, 1=yes)
- canWrite --- which is a numeric flag that indicates whether the ACEName user or group has write access(0=no, 1=yes)
- canDelete --- which is a numeric flag that indicates whether the ACEName user or group has delete access(0=no, 1=yes)
- canManage -- which is a numeric flag that indicates whether the ACEName user or group has manage access(0=no, 1=yes)
- canInheritRead -- which is a numeric flag that indicates whether the ACEName user or group has inherit read access(0=no, 1=yes)
- canInheritWrite -- which is a numeric flag that indicates whether the ACEName user or group has inherit write access(0=no, 1=yes)
- canInheritDelete --- which is a numeric flag that indicates whether the ACEName user or group has inherit delete access(0=no, 1=yes)
- canInheritManage --- which is a numeric flag that indicates whether the ACEName user or group has inherit manage access (0=no, 1=yes)

SASDRUGDEV_GETPASSWORDEXPIRES

Description

Reports the expiration date of the user account currently logged on to SAS Drug Development. The value is contained in the global variable _SDDPROPVALUE_.

Note: The macro variable _SDDPROPVALUE_ can be overwritten by other programming statements, so the value might not be maintained throughout the session.

Syntax

%SASDRUGDEV_GETPASSWORDEXPIRES();

SASDRUGDEV_GETSDDVALUES

Description

Returns a SAS data set that contains all of the valid values which can be passed to a SDD macro for the specified list type. The values also appear in the SAS log.

You can use these values in other macro calls such as SETUSERPOLICIES() and CREATESDDCONTAINER().

Syntax

%SASDRUGDEV_GETSDDVALUES(LISTTYPE=*type*,<, OBJTYPE=*object-type*> <, DSNAME=*SAS-data-set*>);

type is the type of list to be returned. The valid values are POLICIES, STATUS, CONTAINERTYPES, FILETYPES, PROPERTYNAMES.

object-type is only necessary when type=PROPERTYNAMES. It specifies the type of system object whose properties are to be returned. The valid values for *object-type* are USER and any value returned when calling this macro with the *type* set to CONTAINERTYPES or FILETYPES.

SAS-data-set is the name of a SAS data set to create that will contain the values. Specify *SAS-data-set* as *libref.dataset*. The default value is WORK._SDDVALUES_.

The data set contains columns with these names:

- type, which is the type of list
- value, which is the valid value
- objectType, which is the type of object used to retrieve property names

For information about the valid values returned by this macro, see Appendix 2, “Reference.”

SASDRUGDEV_GETUSERGROUPS

Description

Returns a SAS data set that contains the SAS Drug Development user groups that include a specific user account.

Syntax

%SASDRUGDEV_GETUSERGROUPS(SDDUSERID=*user-ID*
<, DSNAME=*SAS-data-set*>);

user-ID is the ID for the user account.

SAS-data-set is the name of a SAS data set the user groups. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._USERGROUPS_**.

The data set contains columns with these names:

- userid, which is the ID for the user account
- groupname, which is the name of the user group

SASDRUGDEV_GETUSERPOLICIES

Description

Returns a SAS data set that contains the system policies for a SAS Drug Development user account.

Syntax

%SASDRUGDEV_GETUSERPOLICIES(SDDUSERID=*user-ID*
<, DSNAME=*SAS-data-set*>);

user-ID is the ID for the user account.

SAS-data-set is the name of a SAS data set to create that will contain the system policies. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._UPOLICIES_**.

The data set contains columns with these names:

- userid, which is the ID for the user account
- policy, which is the system policy

  For information about the valid values for the system policy, see Appendix 2, "Reference."
- action, which is blank

SASDRUGDEV_GETUSERPROPS

Description

Returns a SAS data set that contains the properties of a SAS Drug Development user account.

Syntax

%SASDRUGDEV_GETUSERPROPS(SDDUSERID=*user-ID*
<, DSNAME=*SAS-data-set*>);

user-ID is the ID for the user account.

SAS-data-set is the name of a SAS data set to create that will contain the properties. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._UPROPERTIES_**.

The data set contains columns with these names:

- userid, which is the ID for the user account
- name, which is the name of the property

  For information about the valid values for the name of the user account property, see Appendix 2, "Reference."
- value, which is the value to assign to the property

SASDRUGDEV_GETUSERS

Description

Returns a SAS data set that contains the SAS Drug Development user accounts with a specific status.

Syntax

%SASDRUGDEV_GETUSERS(<STATUS=*status*>
<, DSNAME=*SAS-data-set*>);

Status is the status of the user accounts. These are the valid values:

- active
- inactive
- retired

If you do not specify a value for STATUS, or if you specify an invalid value, all user accounts are returned.

SAS-data-set is the name of a SAS data set to create that will contain the user accounts. Specify *SAS-data-set* as *libref.dataset*. The default value is WORK._GETUSERS_.

The data set contains columns with these names:

- userid, which is the ID for the user account
- firstname, which is the first name of the user
- middlename, which is the middle name of the user
- lastname, which is the last name of the user
- email, which is the e-mail address of the user
- status, which is the status of the user account
- created, which is the date on which the user account was created
- lastModified, which is the date on which the user account was last modified

SASDRUGDEV_GETUSERSWITHPOLICY

Description

Returns a SAS data set that contains the SAS Drug Development user accounts that have a specific system policy.

Syntax

%SASDRUGDEV_GETUSERSWITHPOLICY(SDDPOLICY=*system-policy* <, DSNAME=*SAS-data-set*>);

system-policy is the system policy.

For information about the valid values for the system policy, see Appendix 2, "Reference."

SAS-data-set is the name of a SAS data set to create that will contain the user accounts. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._USERSWITHPOLICY_**.

The data set contains columns with these names:

- policy, which is the system policy

  For information about the valid values for the system policy, see Appendix 2, "Reference."
- userid, which is the ID for the user account

SASDRUGDEV_GROUPEXISTS

Description

Returns a SAS macro variable _SDDGROUPEXISTS_ that contains a value of 1 when the specified group name exists within SAS Drug Development.

Syntax

%SASDRUGDEV_GROUPEXISTS(GROUPNAME=*groupname* <, VERBOSE=*verbose-level*>);

groupname is the group name to check for existance.

verbose-level is the level of message detail to include in the SAS log file. These are the valid values:

- NONE
- INFO (default)
- ERROR

SASDRUGDEV_ISCONTAINER

Description

Returns a SAS macro variable _SDDISCONTAINER_ that contains a value of 1 when the specified path exists as a container object within SAS Drug Development.

Syntax

%SASDRUGDEV_ISCONTAINER(SDDPATH=*sdd-path* <, VERBOSE=*verbose-level*>);

sdd-path is the path (starting at the root) and name of the container object in the SAS Drug Development repository

verbose-level is the level of message detail to include in the SAS log file. These are the valid values:

- NONE
- INFO (default)
- ERROR

SASDRUGDEV_LOGIN

Description

Creates a connection to the SAS Drug Development instance.

Syntax

%SASDRUGDEV_LOGIN(URL=*url*, SDDUSERID=*user-ID*, SDDPASSWORD=*password*<, DEBUGLOG=*debug-log*, DEBUGLEVEL=*debug-level*>);

url is the URL of the SAS Drug Development instance to use in the form **`https://sddname.sas.com/sddremote`**.

user-ID is the SAS Drug Development ID for the user account.

Password is the SAS Drug Development password for the user account.

debug-log is the debugging log file to create. These are the valid values:

- A blank value does not create the debugging log file (default).
- A fully qualified filename creates the debugging log file in the specified location.

debug-level is the level of detail to include in the debugging log file. These are the valid values:

- ALL
- DEBUG
- INFO
- ERROR
- OFF (default)

Note: The DEBUGLOG and DEBUGLEVEL that you specify are in effect for the entire SAS session, not just the SAS Drug Development session. Therefore, to change these values, you must end the SAS session, and then log on again to SAS Drug Development.

SASDRUGDEV_LOGOUT

Description

Closes all open connections to SAS Drug Development.

Syntax

%SASDRUGDEV_LOGOUT();

SASDRUGDEV_SETCONTAINERPERMS

Description

Modifies the permissions on SAS Drug Development repository objects based upon the contents of a SAS data set that contains the permissions metadata for all of the objects.

Syntax

%SASDRUGDEV_SETCONTAINERPERMS(DSNAME=*SAS-data-set* <RECURSE=sdd-*recursion*, KEEPGROUP=*group-name*, VERBOSE=*verbose-level*>);

SAS-data-set is the name of a SAS data set to create that contains the metadata for the objects. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._CONTAINERPERMS_**. This dataset can be constructed using the macro SASDRUGDEV_GETPERMISSIONS.

The data set contains a row for each object in the container object and columns with these names:

- fullPathName, which is the fully qualified name of the object
- ACEName, which is the name of the access control entry represented. This must be an existing group or userid in SAS Drug Development
- canRead --- which is a numeric flag that indicates whether the ACEName user or group has read access(0=no, 1=yes)
- canWrite --- which is a numeric flag that indicates whether the ACEName user or group has write access(0=no, 1=yes)
- canDelete --- which is a numeric flag that indicates whether the ACEName user or group has delete access(0=no, 1=yes)
- canManage -- which is a numeric flag that indicates whether the ACEName user or group has manage access(0=no, 1=yes)
- canInheritRead -- which is a numeric flag that indicates whether the ACEName user or group has inherit read access(0=no, 1=yes)
- canInheritWrite -- which is a numeric flag that indicates whether the ACEName user or group has inherit write access(0=no, 1=yes)
- canInheritDelete --- which is a numeric flag that indicates whether the ACEName user or group has inherit delete access(0=no, 1=yes)
- canInheritManage --- which is a numeric flag that indicates whether the ACEName user or group has inherit manage access (0=no, 1=yes)

sdd-recursion is the SAS Drug Development recursion scope. These are the valid values and the matching SAS Drug Development recursion scopes:

- CHANGED - Apply changed settings to sub-objects
- ALL - Overwrite all permissions on all sub-objects
- NONE (default) - Only apply to this object

group-name is the name of the group that should remain unchanged on any object Access Control List being modified.

verbose-level is the level of message detail to include in the SAS log file. These are the valid values:

- NONE
- INFO (default)
- ERROR

SASDRUGDEV_SETCONTAINERPROPERTY

Description

Sets the value of a single property of a container object.

Syntax

%SASDRUGDEV_SETCONTAINERPROPERTY(SDDPATH=(*sdd-path*, PROPNAME=*property-name*, PROPVALUE=*property-value*);

sdd-path is the path (starting at the root) and name of the container object in the SAS Drug Development repository.

property-name is the name of the property.

For information about the valid values for the name of the container object property, see Appendix 2, "Reference."

property-value is the value to assign to the property.

SASDRUGDEV_SETCONTAINERPROPS

Description

Sets the values of multiple properties of one or more container objects.

Syntax

%SASDRUGDEV_SETCONTAINERPROPS(SDDPATH=(<DSNAME=*SAS-dataset*>);

SAS-data-set is the name of a SAS data set that contains the path, properties, and property values. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._CONTAINERPROPS_**.

The data set must contain columns with these names:

- path, which is the path (starting at the root) and name of the container object in the SAS Drug Development repository
- name, which is the name of the property

  For information about the valid values for the name of the container object property, see Appendix 2, "Reference."
- value, which is the value to assign to the property

All of the data values are strings of any length.

SASDRUGDEV_SETFILEPROPERTY

Description

Sets the value of a single property of a content object.

Syntax

%SASDRUGDEV_SETFILEPROPERTY(SDDPATH=(*sdd-path*, PROPNAME=*property-name*, PROPVALUE=*property-value*);

sdd-path is the path (starting at the root) and name of the content object in the SAS Drug Development repository.

property-name is the name of the property.

For information about the valid values for the name of the content object property, see Appendix 2, "Reference."

property-value is the value to assign to the property.

SASDRUGDEV_SETFILEPROPS

Description

Sets the values of multiple properties of one or more content objects.

Syntax

%SASDRUGDEV_SETFILEPROPS(SDDPATH=(<DSNAME=*SAS-data-set*>);

SAS-data-set is the name of a SAS data set that contains the path, properties, and property values. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._FILEPROPS_.**

The data set must contain columns with these names:

- path, which is the path (starting at the root) and name of the content object in the SAS Drug Development repository
- name, which is the name of the property

  For information about the valid values for the name of the content object property, see Appendix 2, "Reference."
- value, which is the value to assign to the property

All of the data values are strings of any length.

SASDRUGDEV_SETGROUPMEMBERS

Description

Sets or removes the members of a SAS Drug Development user group.

Syntax

%SASDRUGDEV_SETGROUPMEMBERS(<DSNAME=*SAS-data-set*>);

SAS-data-set is the name of a SAS data set that contains the members. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._GMEMBERS_**.

The data set must contain columns with these names:

- groupname, which is the name of the user group
- userid, which is the ID for the user account
- action, which is ADD or REMOVE

SASDRUGDEV_SETUSERAUTH

Description

Sets the authentication provider for a SAS Drug Development user account.

Syntax

%SASDRUGDEV_SETUSERAUTH(SDDUSERID=*user-ID*, *PROVIDER=provider*, *ID=ID*);

user-ID is the ID for the user account.

provider is the remote system to use to authenticate the user account logon.

ID is the ID for the user account as known by the remote system.

SASDRUGDEV_SETUSERPOLICIES

Description

Sets the system policies for a SAS Drug Development user account.

Syntax

%SASDRUGDEV_SETUSERPOLICIES(<DSNAME=*SAS-data-set*>);

SAS-data-set is the name of a SAS data set that contains the system policies. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._UPOLICIES_**.

The data set must contain columns with these names:

- userid, which is the ID for the user account
- policy, which is the system policy to add

  For information about the valid values for the system policy, see Appendix 2, "Reference."
- action, which is ADD or REMOVE

SASDRUGDEV_SETUSERPROPERTY

Description

Sets the value of a single property of a SAS Drug Development user account.

Syntax

%SASDRUGDEV_SETUSERPROPERTY(SDDUSERID=(*user-ID*, PROPNAME=*property-name*, PROPVALUE=*property-value*);

user-ID is the ID for the user account.

property-name is the name of the property.

For information about the valid values for the name of the user account property, see Appendix 2, "Reference."

property-value is the value to assign to the property.

SASDRUGDEV_SETUSERPROPS

Description

Sets the values of multiple properties of one or more SAS Drug Development user accounts.

Syntax

%SASDRUGDEV_SETUSERPROPS(DSNAME=*SAS-data-set*);

SAS-data-set is the name of a SAS data set that contains the user properties and their values. Specify *SAS-data-set* as *libref.dataset*. The default value is **WORK._USERPROPS_**.

The data set must contain columns with these names:

- userid, which is the ID for the user account
- name, which is the name of the property

  For information about the valid values for the name of the user account property, see Appendix 2, "Reference."
- value, which is the value of the property

All of the data values are strings of any length.

SASDRUGDEV_SETUSERSTATUS

Description

Sets the status of a SAS Drug Development user account.

Syntax

%SASDRUGDEV_SETUSERSTATUS(SDDUSERID=*user-ID*, STATUS=*status*, REASON=*reason*));

user-ID is the ID for the user account.

status is the status of the user account. These are the valid values:

- active
- inactive

- retired

reason is the reason for the status change.

SASDRUGDEV_UPDATEUSER

Description

Modifies the password for a SAS Drug Development user account. The user running this macro must the Manage User policy assigned within SAS Drug Development.

Syntax

%SASDRUGDEV_UPDATEUSER(SDDUSERID=*user-ID, SDDPASSWORD=password*<,*VERBOSE=verbose-level*>);

user-ID is the ID for the user account.

password is the new password for the user account.

verbose-level is the level of message detail to include in the SAS log file. These are the valid values:

- NONE
- INFO (default)
- ERROR

SASDRUGDEV_USEREXISTS

Description

Determines whether a SAS Drug Development user account exists.

Syntax

%SASDRUGDEV_USEREXISTS(SDDUSERID=*user-ID*);

user-ID is the ID for the user account.

Examples

Introduction

These examples are intended to illustrate certain techniques only. They do not include all of the required code, such as logging on and logging off, to run successfully.

Create Files and Content Objects

This example code creates a local file and a content object in the SAS Drug Development repository. Next, it creates a second local file from the content object in the SAS Drug Development repository, and then deletes the content object.

```
%let sddPath=/SDD/RAPI_testing/Test.doc;
%let localfile1=C:\temp\RemoteAPI\Test.doc;
%let localfile2=C:\temp\RemoteAPI\Test2.doc;
filename testfile "&localfile1";
data _null_;
file testfile;
put "This is a small test file.";
run;
filename testfile clear;

/* confirm that the file does not yet exist in SAS DD- _sddRC_ should be
-4 */
%sasdrugdev_fileexists(sddpath=&sddPath)
/* create the file */
%sasdrugdev_createsddfile(localpath=&localfile1, sddpath=&sddPath,
type=document)
/* confirm that the file was created - _sddRC_ should be 1 */
%sasdrugdev_fileexists(sddpath=&sddPath)
/* create the file on the local system from the SAS DD file - _sddRC_
should be 1 */
%sasdrugdev_createlocalfile(localpath=&localfile2, sddpath=&sddPath)
/* delete the SAS DD file */
%sasdrugdev_deletefile(sddpath=&sddPath)
```

Create a Container Object

This example code creates a container object in the SAS Drug Development repository, verifies that the container object exists, and then deletes the container object.

```
%let sddPath=/SDD/ANewFolder;
/* confirm that the container does not yet exist in SAS DD- _sddRC_
should be -4 */
%sasdrugdev_containerexists(sddpath=&sddPath)
/* create the new container */
%sasdrugdev_createsddcontainer(sddpath=&sddPath, type=folder)
/* confirm that the container was created - _sddRC_ should be 1 */
%sasdrugdev_containerexists(sddpath=&sddPath)
/* delete the container */
%sasdrugdev_deletecontainer(sddpath=&sddPath)
```

Manipulate a SAS Data Set

This example code loads a SAS data set into the SAS Drug Development repository, gets the data set's properties, and then deletes the data set.

```
%let sddPath=/SDD/RAPI_testing/test.sas7bdat;
%let localtestdir=C:\temp\RemoteAPI\;
%let localfullpath=&localtestdir\test.sas7bdat;
libname testlib "&localtestdir";
data testlib.test;
    x=1;y=1;z=1;
    output;
    x=2;y=2;z=2;
    output;
    x=3;y=3;z=3;
```

```
    output;
    x=4;y=4;z=4;
    output;
run;
/* confirm that the data set does not yet exist in SAS DD- _sddRC_ should
be -4 */
%sasdrugdev_fileexists(sddpath=&sddPath)
/* create the data set */
%sasdrugdev_createsddfile(localpath=&localfullpath, sddpath=&sddPath,
type=sasDataTable)
/* confirm that the data set was created - _sddRC_ should be 1 */
%sasdrugdev_fileexists(sddpath=&sddPath)
/* get the data set properties */
%sasdrugdev_getfileprops(sddpath=&sddPath, dsname=work.myDSprops)
title "Properties of &sddpath";
proc print data=myDSprops;
run;
title;
/* delete the datasets */
%sasdrugdev_deletefile(sddpath=&sddPath)
```

Manipulate Metadata

This example code gets the metadata for all objects in the SAS Drug Development repository, and then prints the metadata.

```
%sasdrugdev_getobjects(sddpath=%bquote(/SDD), dsname=work.getObjects)
title "Metadata for all objects in SAS Drug Dev";
proc print data=work.getObjects;
run;
title;
```

Manipulate a User Group and Its Members

This example code creates a user group, adds two members to it (admin and goodUserID), gets a list of the members in the user group, prints the list, and then deletes the user group.

```
%let myGroup=NewGroup;
%let goodUserID=myUser;
%sasdrugdev_creategroup(sddgroupname=&myGroup,sddgroupdesc=%str(a test
group that contains &goodUserID and admin))
data groupdata;
    length groupname userid action $50;
    groupname="&myGroup";
    action="add";
    userid="admin";
    output;
    userid="&goodUserID";
    output;
run;
%sasdrugdev_setgroupmembers(dsname=groupdata);
%sasdrugdev_getgroupmembers(sddgroupname=&myGroup)
title "List of users in group &myGroup";
proc print data=work._gmembers_;
run;
title;
%sasdrugdev_deletegroup(sddgroupname=&myGroup)
```

Get the User Accounts That Have a Specific System Policy

This example code gets the user accounts that have the system policy **User can view the global audit trail**.

```
%let policy=Auditor;
%sasdrugdev_getuserswithpolicy(sddpolicy=&policy)
title "List of users with policy &policy";
proc print;
run;
title;
```

Create a User Account and Manipulate Its System Policies

This example code creates a user account, prints the default system policies, adds and removes some system policies, and then prints the system policies (with the new system policy changes).

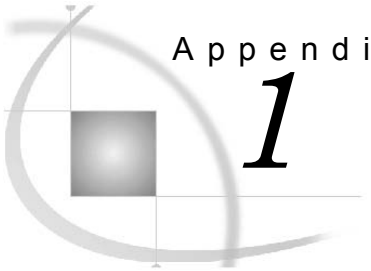
```
%let sdduserid=aTestUser5;
%let sddpassword=1Temp_Pw!;
%let sddfirstname=firstname;
%let sddlastname=lastname;
%let sddemail=somebody@somewhere.com;
/* confirm that the userid does not yet exist in SAS DD- _sddRC_ should
be -10 */
%sasdrugdev_userexists(sdduserid=&sdduserid)
/* create the userid */
%sasdrugdev_createuser(sdduserid=&sdduserid, sddpassword=&sddpassword,
sddfirstname=&sddfirstname, sddlastname=&sddlastname, sddemail=&sddemail)
/* confirm that the user was created- _sddRC_ should be 0 */
%sasdrugdev_userexists(sdduserid=&sdduserid)
/* get the user's policies (should be empty) */
%sasdrugdev_getuserpolicies(sdduserid=&sdduserid, dsname=work.initpols)
title "initial policies returned for &sdduserid";
proc print data=initpols;
run;
/* add user policies */
data work._UPOLICIES_;
    length userid policy action $100;
    userid=symget("sdduserid");
    action="add";
    policy="Archiver";
    output;
    policy="SystemManager";
    output;
    policy="AdvancedLoader";
    output;
run;
title "data being sent back into setuserpolicies";
proc print data=work._UPOLICIES_;
run;
%sasdrugdev_setuserpolicies(dsname=work._UPOLICIES_)
/* remove user policies */
data work._UPOLICIES_;
    length userid policy action $100;
    userid=symget("sdduserid");
    action="remove";
    policy="SystemManager";
    output;
    policy="AdvancedLoader";
    output;
run;
title "data being sent back into setuserpolicies";
proc print data=work._UPOLICIES_;
run;
```

```
%sasdrugdev_setuserpolicies(dsname=work._UPOLICIES_)
/* Confirm updates. Should only have policy Archiver */
%sasdrugdev_getuserpolicies(sdduserid=&sdduserid,
dsname=work.updatedpols)
title "List of policies for &sdduserid after setuserpolicies calls
";
proc print data=work.updatedpols;
run;
title;
/* Set the user to inactive (Retiring the user is an option, but cannot
be undone) */
%sasdrugdev_setuserstatus(sdduserid=&sdduserid, status=inactive,
reason=%str(done with testing policies))
```

Manipulate the Properties of a Content Object

This example code gets the properties of a content object, modifies the properties, and prints the new properties.

```
%let sddPath=/SDD/Test.doc;
%sasdrugdev_getfileprops(sddpath=&sddPath, dsname=work._initfileprops_);
** SETUP - alter all data to include uniqueID for this test session **;
data work.testprops;
    length path name value $50;
    set _initfileprops_;
    path="&sddPath";
    if name="keywords" then value=(trim(left(value)) || " more
    keywords");
    if name="description" then value="changing value of the desciption";
run;
title "data being sent back to setfileproperties";
proc print data=work.testprops;
run;
%sasdrugdev_setfileprops(dsname=testprops);
%sasdrugdev_getfileprops(sddpath=&sddPath, dsname=work._afterfileprops_);
title "final properties for file &sddPath";
proc print data=work._afterfileprops_;
run;
title;
```



Appendix

1 Information for Users of the Command Facility

Correspondence of Command Facility Macros to SAS Drug Development Macros

The Command Facility macros in earlier versions of SAS Drug Development are the predecessors of the SAS Drug Development macros. If you have used Command Facility macros in the past, this summary of the correspondence of the Command Facility macros to the SAS Drug Development macros will be helpful.

Command Facility Macro Name	SAS Drug Development Macro Name
SWD_CHECKIN	Not implemented
SWD_CHECKOUT	Not implemented
SWD_CLEARSAVEDSETTINGS	Not implemented
SWD_COPYFILE	SASDRUGDEV_COPYCONTAINER
	SASDRUGDEV_COPYFILE
SWD_DELETE	SASDRUGDEV_DELETECONTAINER
	SASDRUGDEV_DELETEFILE
SWD_FILENAME	Not implemented
SWD_GETALLOBJECTPROPERTIES	SASDRUGDEV_GETCONTAINERPROPS
	SASDRUGDEV_GETFILEPROPS
SWD_GETFILE	SASDRUGDEV_CREATELOCALFILE
SWD_GETOBJECTPROPERTY	Not implemented
SWD_GETSAVEDSETTING	Not implemented
SWD_GETVERSIONINFO	Not implemented
SWD_GETVERSIONLIST	Not implemented
SWD_LIBNAME	Not implemented
SWD_LISTOBJECTS	SASDRUGDEV_GETOBJECTS
SWD_MAKEDIR	SASDRUGDEV_CREATESDDCONTAINER
SWD_MAKEDIRS	Not implemented
SWD_OBJECTEXISTS	SASDRUGDEV_CONTAINEEXISTS
	SASDRUGDEV_FILEEXISTS
SWD_PUTFILE	SASDRUGDEV_CREATESDDFILE
SWD_SAVESETTINGS	Not implemented
SWD_SETCHECKOUTSTATE	Not implemented
SWD_SETOBJECTPROPERTY	SASDRUGDEV_SETCONTAINERPROPERTY
	SASDRUGDEV_SETCONTAINERPROPS
	SASDRUGDEV_SETFILEPROPERTY
	SASDRUGDEV_SETFILEPROPS
SWD_START	SASDRUGDEV_LOGIN
SWD_STOP	SASDRUGDEV_LOGOUT
SWD_TURNONVERSIONING	Not implemented
SWD_UNDOCHECKOUT	Not implemented

APPENDIX 2

Reference

Name of an Object Type When Specified as a Macro Parameter

When you call a macro that creates an object, such as SASDRUGDEV_CREATESDDFILE, you must specify the name of the object type. This section lists the names of the object types that are provided with SAS Drug Development.

Note: Your organization might have custom object types that are not included in this section.

Container Objects

Here are the names of each container object when specified as a macro parameter:

- compound
- folder
- indication

- protocol
- trashcan

Content Objects

Here are the names of each content object when specified as a macro parameter:

- archive
- clinicalDataView
- document
- job
- jobLog
- note
- process
- sasCatalog
- sasDataTable
- search
- shortcut
- studyDef
- tableDef
- tableView
- urlLink

Editable Object Properties

SAS Drug Development provides a set of object types. This section lists the names of the editable properties for each object type and the editable properties for user accounts. You need these names when you call a macro that sets the properties of an object or a user account.

Note: The name of each object type that is used in a macro is the same as the name in the SAS Drug Development application. Any exceptions are noted.

For more information about each object type, see the online Help.

Container Objects

Compound

A Compound object has these editable properties:

- contactName
- contactPhone
- description
- keywords
- scientificName

- tradeName

Folder

A Folder object has these editable properties:

- description
- keywords

Indication

An Indication object has these editable properties:

- contactName
- contactPhone
- description
- keywords

Protocol

A Protocol object has these editable properties:

- description
- keywords
- number
- phase
- title

Trashcan

A Trashcan object has these editable properties:

- description
- keywords

Content Objects

Archive

An Archive object has these editable properties:

- comment
- description
- keywords

Catalog (sasCatalog)

A Catalog object has these editable properties:

- allowStructureChanges
- comment

- description
- formats
- keywords
- numberOfCatalogEntries

Clinical Data View (clinicalDataView)

A Clinical Data View object has these editable properties:

- comment
- description
- keywords

Data Table View (sasDataTable)

A Data Table View object has these editable properties:

- allowStructureChanges
- columns
- comment
- description
- keywords
- numberOfColumns
- numberOfRows
- tableLabel

Document

A Document object has these editable properties:

- comment
- description
- keywords

Job

A Job object has these editable properties:

- comment
- description
- keywords

Job Log (jobLog)

A Job Log object has these editable properties:

- comment
- description
- keywords

Link (urlLink)

A Link object has these editable properties:

- description
- keywords
- text

Note

A Note object has these editable properties:

- description
- keywords
- text

Process

A Process object has these editable properties:

- comment
- description
- keywords

Search

A Search object has these editable properties:

- criteria
- description
- keywords

Shortcut

A Shortcut object has these editable properties:

- description
- keywords
- sddPath
- sddPathSelector

Study Definition File (studyDef)

A Study Definition File object has these editable properties:

- comment
- description
- keywords

Table Definition File (tableDef)

A Table Definition File object has these editable properties:

- comment
- description
- keywords

Table View (tableView)

A Table View object has these editable properties:

- comment
- description
- keywords

Editable User Account Properties

A user account has these editable properties:

- certificate
- city
- company
- country
- department
- email
- employeeid
- fax
- firstname
- imageurl
- lastname
- middlename
- mobile
- office
- pager
- phone
- postalcode
- salutation
- state
- street
- title

System Policies

For any macro that can set or get the system policies for a user account, here are the values for the system policies:

- AdvancedLoader
- Archiver

- Auditor
- Console
- DataDefinitionExplorer
- DataExplorer
- Documenter
- GroupManager
- JobEditor
- JobResultsViewer
- OwnerManager
- PermissionsReport
- PolicyManager
- ProcessEditor
- SASManager
- Scheduler
- Signer
- SignerDesignator
- StudyDefinitionEditor
- SystemManager
- TrashcanManager
- UndoCheckout
- UserManager

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **`yourturn@sas.com`**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **`suggest@sas.com`**.