



THE  
POWER  
TO KNOW.

# SAS<sup>®</sup> Drug Development 3.5 Macros User's Guide

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**

Copyright © 2009, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America.

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

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

**U.S. Government Restricted Rights Notice:** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19, Commercial Computer Software-Restricted Rights (June 1987).

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](http://support.sas.com/publishing) or call 1-800-727-3228.

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

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

---

## Contents

### **Chapter 1 Introduction 1**

Audience 1

Typographic and Syntax Conventions Used in This Guide 1

### **Chapter 2 Installing the Macros 3**

Overview 3

Requirements 3

Installation Instructions for Microsoft Windows 3

Installation Instructions for UNIX 4

Verifying the Installation 6

### **Chapter 3 SAS Drug Development Macros 9**

Introduction 11

Macro Return Codes 11

Macros and System Policies 12

The Macros 12

Examples 32

### **Appendix 1 Information for Users of the Command Facility 37**

Correspondence of Command Facility Macros to SAS Drug Development Macros 37

### **Appendix 2 Reference 39**

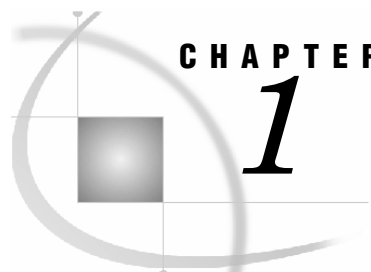
Name of an Object Type When Specified as a Macro Parameter 39

Editable Object Properties 40

Editable User Account Properties 44

System Policies 45





# CHAPTER 1 Introduction

---

<i>Audience</i> .....	1
<i>Typographic and Syntax Conventions Used in This Guide</i> .....	1

---

## 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 and Syntax Conventions Used in This Guide

Throughout this guide, you will see the following typographic conventions:

Convention	Description
monospace font	denotes code, such as a code example
<b>monospace bold</b> font	denotes text that you type, such as an object name
<b><i>monospace bold italics</i></b> font	denotes a value that you specify, such as your name

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

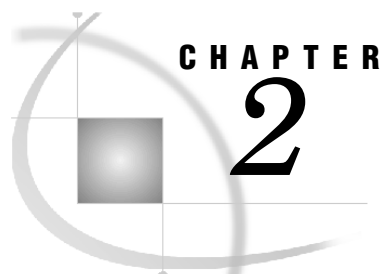
## Syntax Conventions

```

PROC DATASETS <LIBRARY=libref> <MEMTYPE=(mtype-list)>
      <DETAILS | NODETAILS> <other-options>;
RENAME variable-1=new-name-1 <... variable-n=new-name-n>;

```

- |  |  |
|--|--|
| <p><b>1</b> SAS keywords, such as statement or procedure names, appear in bold type.</p> <p><b>2</b> Values that you must spell as they are given in the syntax appear in uppercase type.</p> <p><b>3</b> Optional arguments appear inside angle brackets(&lt;&gt;).</p> | <p><b>4</b> Mutually exclusive choices are joined with a vertical bar( ).</p> <p><b>5</b> Values that you must supply appear in italic type.</p> <p><b>6</b> Argument groups that you can repeat are indicated by an ellipsis (...).</p> |
|--|--|



## Installing the Macros

<i>Overview</i> .....	3
<i>Requirements</i> .....	3
<i>Installation Instructions for Microsoft Windows</i> .....	3
<i>Installation Instructions for UNIX</i> .....	4
<i>Verifying the Installation</i> .....	6

### 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 Files\SAS\SASFoundation\9.2`.

- 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 .sas files.

- 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/SASDrugDe
vRemoteAPI_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:

**6 Verifying the Installation** *Chapter 2*

```
-JREOPTIONS=(
-Dsas.app.class.dirs=/apps/sas9.2/SASFoundation/9.2/RemoteAPI/SASDrugDe
vRemoteAPI_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://sdd-
instance/sddremote),sdduserid=%str(sdd-user-ID),sddpassword=%str(sdd-
password));

/* 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.

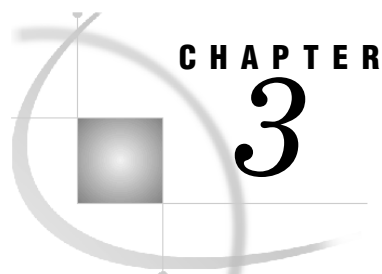
The SAS log will contain information that might be useful for debugging the installation of the SAS Drug Development macros.

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://sdd-  
instance/sddremote),sdduserid=%str(sdd-user-ID),sddpassword=%str(sdd-  
password),debuglog=absolute-path-to-log,debuglevel=DEBUG)
```

**8** **Verifying the Installation** *Chapter 2*



## CHAPTER

## 3

# SAS Drug Development Macros

---

<i>Introduction</i> .....	11
<i>Macro Return Codes</i> .....	11
<i>Macros and System Policies</i> .....	12
<i>The Macros</i> .....	12
<i>Using the Ampersand Character (&amp;) in URLs</i> .....	12
<i>The Proper Case for Parameter Values</i> .....	13
SASDRUGDEV_CONTAINEREXISTS .....	13
Description .....	13
Syntax .....	13
SASDRUGDEV_COPYCONTAINER .....	13
Description .....	13
Syntax .....	13
SASDRUGDEV_COPYFILE .....	13
Description .....	13
Syntax .....	14
SASDRUGDEV_CREATEGROUP .....	14
Description .....	14
Syntax .....	14
SASDRUGDEV_CREATELOCALFILE .....	14
Description .....	14
Syntax .....	14
SASDRUGDEV_CREATESDDCONTAINER .....	15
Description .....	15
Syntax .....	15
SASDRUGDEV_CREATESDDFILE .....	15
Description .....	15
Syntax .....	15
SASDRUGDEV_CREATEUSER .....	16
Syntax .....	16
SASDRUGDEV_DELETECONTAINER .....	16
Description .....	16
Syntax .....	16
SASDRUGDEV_DELETEFILE .....	16
Description .....	16
Syntax .....	16
SASDRUGDEV_DELETEGROUP .....	17
Description .....	17
Syntax .....	17
SASDRUGDEV_FILEEXISTS .....	17
Description .....	17
Syntax .....	17
SASDRUGDEV_GETCONTAINERPROPS .....	17
Description .....	17
Syntax .....	17
SASDRUGDEV_GETFILEPROPS .....	18
Description .....	18
Syntax .....	18
SASDRUGDEV_GETGROUPS .....	18
Description .....	18
Syntax .....	19
SASDRUGDEV_GETGROUPMEMBERS .....	19
Description .....	19
Syntax .....	19

<i>SASDRUGDEV_GETOBJECTS</i> .....	19
Description.....	19
Syntax .....	19
<i>SASDRUGDEV_GETPERMISSIONS</i> .....	20
Description.....	20
Syntax .....	20
<i>SASDRUGDEV_GETPASSWORDEXPIRES</i> .....	22
Description.....	22
Syntax .....	22
<i>SASDRUGDEV_GETSDDVALUES</i> .....	22
Description.....	22
Syntax .....	22
<i>SASDRUGDEV_GETUSERGROUPS</i> .....	23
Description.....	23
Syntax .....	23
<i>SASDRUGDEV_GETUSERPOLICIES</i> .....	23
Description.....	23
Syntax .....	23
<i>SASDRUGDEV_GETUSERPROPS</i> .....	24
Description.....	24
Syntax.....	24
<i>SASDRUGDEV_GETUSERS</i> .....	24
Description.....	24
Syntax .....	25
<i>SASDRUGDEV_GETUSERSWITHPOLICY</i> .....	25
Description.....	25
Syntax .....	25
<i>SASDRUGDEV_LOGIN</i> .....	26
Description.....	26
Syntax .....	26
<i>SASDRUGDEV_LOGOUT</i> .....	27
Description.....	27
Syntax .....	27
<i>SASDRUGDEV_SETCONTAINERPROPERTY</i> .....	27
Description.....	27
Syntax .....	27
<i>SASDRUGDEV_SETCONTAINERPROPS</i> .....	27
Description.....	27
Syntax .....	27
<i>SASDRUGDEV_SETFILEPROPERTY</i> .....	28
Description.....	28
Syntax.....	28
<i>SASDRUGDEV_SETFILEPROPS</i> .....	28
Description.....	28
Syntax .....	28
<i>SASDRUGDEV_SETGROUPMEMBERS</i> .....	29
Description.....	29
Syntax .....	29
<i>SASDRUGDEV_SETUSERAUTH</i> .....	29
Description.....	29
Syntax .....	29
<i>SASDRUGDEV_SETUSERPOLICIES</i> .....	30
Description.....	30
Syntax .....	30
<i>SASDRUGDEV_SETUSERPROPERTY</i> .....	30
Description.....	30
Syntax .....	30
<i>SASDRUGDEV_SETUSERPROPS</i> .....	31
Description.....	31
Syntax .....	31
<i>SASDRUGDEV_SETUSERSTATUS</i> .....	31
Description.....	31

<i>Syntax</i> .....	31
<i>SASDRUGDEV_USEREXISTS</i> .....	32
<i>Description</i> .....	32
<i>Syntax</i> .....	32
<i>Examples</i> .....	32
<i>Introduction</i> .....	32
<i>Create Files and Content Objects</i> .....	32
<i>Create a Container Object</i> .....	33
<i>Manipulate a SAS Data Set</i> .....	33
<i>Manipulate Metadata</i> .....	33
<i>Manipulate a User Group and Its Members</i> .....	34
<i>Get the User Accounts That Have a Specific System Policy</i> .....	34
<i>Create a User Account and Manipulate Its System Policies</i> .....	34
<i>Manipulate the Properties of a Content Object</i> .....	35

---

## 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:

<b><code>_SDDRC_</code> Value</b>	<b>Explanation</b>
-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
-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

---

<b>_SDDRC_ Value</b>	<b>Explanation</b>
-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=*destination-path*);

*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\_CREATE\_SDDCONTAINER

### Description

Creates a container object in the SAS Drug Development repository. All parent folders must already exist.

### Syntax

```
%SASDRUGDEV_CREATE_SDDCONTAINER(SDDPATH=sdd-path)  
<, TYPE=type>;
```

*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
- trashcan

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

---

## SASDRUGDEV\_CREATE\_SDDFILE

### Description

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

### Syntax

```
%SASDRUGDEV_CREATE_SDDFILE(LOCALPATH=local-path, SDDPATH=sdd-path,  
path<, 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=last-  
name, 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 the SAS 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 the SAS 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 to 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 to 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 to 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 to 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  
<RECURSIVE=boolean, 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.

<i>recursive</i>	is set to <code>true</code> when the metadata for all subcontainer objects is to be returned and <code>false</code> when only the metadata for the objects in the current container object is to be returned. The default value is <code>false</code> .
<i>typedef</i>	is the SDD file type to return, or <code>*</code> for all file types. The default value is <code>**</code>
<i>SAS-data-set</i>	is the name of a SAS data set to create to contain the metadata for the objects. Specify <i>SAS-data-set</i> as <i>libref.dataset</i> . The default value is <code>WORK._GETOBJECTS_</code> .

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
<RECURSIVE=boolean, TYPEDEF=object-type, DSNAME=SAS-data-set>);
```



<i>sdd-path</i>	is the path (starting at the root) and name of the container object in the SAS Drug Development repository.
<i>recursive</i>	is set to true when the permissions metadata for all subcontainer objects is to be returned and false when only the permissions metadata for objects in the current container object is to be returned. The default value is false.
<i>typedef</i>	is the string containing the SDD file type to return, or * for all file types. The default value is is “*”.
<i>SAS-data-set</i>	is the name of a SAS data set to create to contain the metadata for the objects. Specify <i>SAS-data-set</i> as <i>libref.dataset</i> . 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 to 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 to create to contain 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 to 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 to 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>
<, DSNNAME=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 to 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
<, DSNNAME=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 to 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\_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\_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-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._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\_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.

```

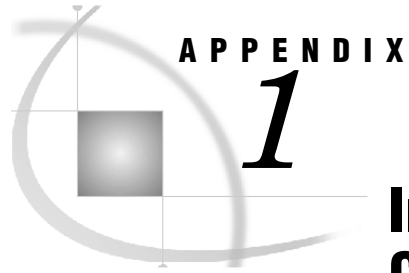
%let sddPath=/SDD/Test.doc;
%sasdrugdev_getfileprops(sddpath=&sddPath, dsname=work._initfileprops_);
** SETUP - alter all data to include uniqueID for this test session **;
data work.testprops;
  length path name value $50;
  set _initfileprops_;
  path="&sddPath";
  if name="keywords" then value=(trim(left(value)) || " more keywords");
  if name="description" then value="changing value of the description";
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_);

```

**36 Examples** *Chapter 3*

```
title "final properties for file &sddPath";  
proc print data=work._afterfileprops_;  
run;  
title;
```





# Information for Users of the Command Facility

---

*Correspondence of Command Facility Macros to SAS Drug Development Macros..... 37*

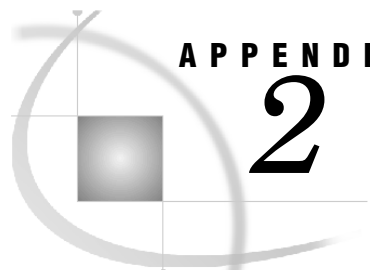
---

## **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.

**38 Correspondence of Command Facility Macros to SAS Drug Development Macros** *Appendix 1*

<b>Command Facility Macro Name</b>	<b>SAS Drug Development Macro Name</b>
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

---

<i>Name of an Object Type When Specified as a Macro Parameter</i> .....	39
<i>Container Objects</i> .....	39
<i>Content Objects</i> .....	40
<i>Editable Object Properties</i> .....	40
<i>Container Objects</i> .....	40
<i>Compound</i> .....	40
<i>Folder</i> .....	41
<i>Indication</i> .....	41
<i>Protocol</i> .....	41
<i>Trashcan</i> .....	41
<i>Content Objects</i> .....	41
<i>Archive</i> .....	41
<i>Catalog (sasCatalog)</i> .....	41
<i>Clinical Data View (clinicalDataView)</i> .....	42
<i>Data Table View (sasDataTable)</i> .....	42
<i>Document</i> .....	42
<i>Job</i> .....	42
<i>Job Log (jobLog)</i> .....	42
<i>Link (urlLink)</i> .....	43
<i>Note</i> .....	43
<i>Process</i> .....	43
<i>Search</i> .....	43
<i>Shortcut</i> .....	43
<i>Study Definition File (studyDef)</i> .....	43
<i>Table Definition File (tableDef)</i> .....	44
<i>Table View (tableView)</i> .....	44
<i>Editable User Account Properties</i> .....	44
<i>System Policies</i> .....	45

---

## Name of an Object Type When Specified as a Macro Parameter

When you call a macro that creates an object, such as SASDRUGDEV\_CREATE\$DDFILE, 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**.



# SAS® Publishing delivers!

Whether you are new to the workforce or an experienced professional, you need to distinguish yourself in this rapidly changing and competitive job market. SAS® Publishing provides you with a wide range of resources to help you set yourself apart.

## SAS® Press Series

Need to learn the basics? Struggling with a programming problem? You'll find the expert answers that you need in example-rich books from the SAS Press Series. Written by experienced SAS professionals from around the world, these books deliver real-world insights on a broad range of topics for all skill levels.

[support.sas.com/saspress](http://support.sas.com/saspress)

## SAS® Documentation

To successfully implement applications using SAS software, companies in every industry and on every continent all turn to the one source for accurate, timely, and reliable information—SAS documentation. We currently produce the following types of reference documentation: online help that is built into the software, tutorials that are integrated into the product, reference documentation delivered in HTML and PDF—free on the Web, and hard-copy books.

[support.sas.com/publishing](http://support.sas.com/publishing)

## SAS® Learning Edition 4.1

Get a workplace advantage, perform analytics in less time, and prepare for the SAS Base Programming exam and SAS Advanced Programming exam with SAS® Learning Edition 4.1. This inexpensive, intuitive personal learning version of SAS includes Base SAS® 9.1.3, SAS/STAT®, SAS/GRAPH®, SAS/QC®, SAS/ETS®, and SAS® Enterprise Guide® 4.1. Whether you are a professor, student, or business professional, this is a great way to learn SAS.

[support.sas.com/LE](http://support.sas.com/LE)



THE  
POWER  
TO KNOW®