

Novell Identity Manager Fan-Out Driver

3.5

www.novell.com

API DEVELOPER GUIDE

March 19, 2007



Novell®

Legal Notices

Novell, Inc. and Omnibond Systems LLC. make no representations or warranties with respect to the contents or use of this documentation, and specifically disclaim any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. and Omnibond Systems LLC. reserve the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. and Omnibond Systems LLC. make no representations or warranties with respect to any software, and specifically disclaim any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. and Omnibond Systems LLC. reserve the right to make changes to any and all parts of the software, at any time, without any obligation to notify any person or entity of such changes.

You may not export or re-export this product in violation of any applicable laws or regulations including, without limitation, U.S. export regulations or the laws of the country in which you reside.

Copyright © 2004 Omnibond Systems, LLC. All Rights Reserved. Licensed to Novell, Inc. Portions Copyright © 2004 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

The Solaris* standard IO library has kernel limitations that interfere with the operation of the Provisioning Manager. Therefore, components for Solaris use the AT&T* SFIO library. Use of this library requires the following notice:

The authors of this software are Glenn Fowler, David Korn and Kiem-Phong Vo.

Copyright (c) 1991, 1996, 1998, 2000, 2001, 2002 by AT&T Labs - Research.

Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software.

This software is being provided as is, without any express or implied warranty. In particular, neither the authors nor AT&T Labs make any representation or warranty of any kind concerning the merchantability of this software or its fitness for any particular purpose.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

eDirectory is a trademark of Novell, Inc.

NetWare is a registered trademark of Novell, Inc. in the United States and other countries.

Novell is a registered trademark of Novell, Inc. in the United States and other countries.

Novell Directory Services and NDS are registered trademarks of Novell, Inc. in the United States and other countries.

Nsure is a trademark of Novell, Inc.

Third-Party Materials

All third-party trademarks are the property of their respective owners.

Contents

About This Guide	7
1 About the API	9
1.1 Using the API in the MVS Environment	9
1.2 Using the API on a NetWare Server	9
1.3 Using the API in the OS/400 Environment.	10
1.4 Using the API in the UNIX Environment	10
1.5 API Function List	10
2 C Language API Reference	13
ASC_ADMINRSTPASSWD	14
ASC_CHGPASSWD.	17
ASC_CHKPASSWD.	19
ASC_DAYS	22
ASC_GETCONTEXT	23
ASC_GRPMEM	26
ASC_INIT	29
ASC_INIT_EXT	31
ASC_LISTSEQV.	33
ASC_READATTR.	36
ASC_RIGHTS.	39
ASC_SECEQUAL.	42
ASC_STRERROR	44
ASC_TERM	45
ASC_USER_INCLUDE_EXCLUDE	47
3 Java Language API Reference	49
Class com.novell.asam.JAscAuth.JAscAuth	50
Classes Used by checkPassword.	58
Exception Classes in com.novell.asam.JAscAuth	60
4 API Examples	63
4.1 Adding API Support to the Apache Web Server	63
4.2 Adding API Support to the QUALCOMM POP Server.	63
4.3 Adding API Support to SASL.	63
4.4 Adding API Support to SSH Secure Shell	64
4.5 Adding API Support to TACACS+	64
A Troubleshooting the API	65

About This Guide

This guide describes the AS Client API of the Novell® Identity Manager Fan-Out driver. It also describes simple modifications that you can make to several popular products to make use of the API.

This guide assumes that you have knowledge of eDirectory™ and program development for the operating system platform on which Platform Services is installed, and that you are familiar with the concepts and facilities of the Identity Manager Fan-Out driver.

This guide is divided into the following sections:

- ♦ Chapter 1, “About the API,” on page 9
- ♦ Chapter 2, “C Language API Reference,” on page 13
- ♦ Chapter 3, “Java Language API Reference,” on page 49
- ♦ Chapter 4, “API Examples,” on page 63
- ♦ Appendix A, “Troubleshooting the API,” on page 65

Additional Documentation

The following publications contain information about the Identity Manager Fan-Out driver. These publications are available at the [Identity Manager Driver Web site \(http://www.novell.com/documentation/dirxmldrivers\)](http://www.novell.com/documentation/dirxmldrivers).

Concepts and Facilities Guide

Core Driver Administration Guide

Platform Services Planning Guide and Reference

Platform Services Administration Guide for Linux and UNIX

Platform Services Administration Guide for MVS

Platform Services Administration Guide for OS/400

NetWare Intercept and API Administration Guide

API Developer Guide

Messages Reference

Core Driver Quick Start Guide for Linux and Solaris

Core Driver Quick Start Guide for NetWare

Core Driver Quick Start Guide for Windows

Platform Services Quick Start Guide for AIX

Platform Services Quick Start Guide for FreeBSD, HP-UX, Linux, and Solaris

Platform Services Quick Start Guide for MVS CA-ACF2

Platform Services Quick Start Guide for MVS CA-Top Secret

Platform Services Quick Start Guide for MVS RACF

Platform Services Quick Start Guide for OS/400

NetWare Intercept and API Quick Start Guide

Documentation for related products, such as Identity Manager and eDirectory, is available at the [Novell Documentation Web site \(http://www.novell.com/documentation\)](http://www.novell.com/documentation).

Documentation Updates

For the most recent versions of -Identity Manager Fan-Out driver documentation, see the [Identity Manager Driver Web site \(http://www.novell.com/documentation/dirxmldrivers\)](http://www.novell.com/documentation/dirxmldrivers).

Documentation Conventions

In this documentation, a greater-than symbol (>) is used to separate actions within a step and items within a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark. When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as UNIX*, should use forward slashes as required by your software.

User Comments

We want to hear your comments and suggestions about this manual and the other documentation included with the driver. To contact us, send e-mail to namdoc@novell.com.

About the API

1

Novell® Identity Manager Fan-Out driver platforms provide an Authentication Services application programming interface (API) that can be used by applications to access eDirectory™. This API is compatible with the AS Client API that was provided in the NDS® Authentication Services and the Account Management 3.0 products. In order to use this API, you must obtain and install the Identity Manager Fan-Out driver.

The platform configuration file provides the information necessary for establishing communications with a core driver. For details about the platform configuration file, see the *Platform Services Planning Guide and Reference*.

In the C language environment, a call must be made to `ASC_INIT()` or `ASC_INIT_EXT()` to process the platform configuration file and initialize the environment before API calls can be made to the core driver. The header file `ascauth.h` provides the function prototypes for the API calls and their return value definitions.

In the Java* environment, a call must be made to the `init()` method to process the platform configuration file and initialize the environment before API calls can be made to the core driver. Class `com.novell.asam.JAscAuth.JAscAuth` provides the methods used to call the API.

Topics in this section are

- ♦ [Section 1.1, “Using the API in the MVS Environment,” on page 9](#)
- ♦ [Section 1.2, “Using the API on a NetWare Server,” on page 9](#)
- ♦ [Section 1.3, “Using the API in the OS/400 Environment,” on page 10](#)
- ♦ [Section 1.4, “Using the API in the UNIX Environment,” on page 10](#)
- ♦ [Section 1.5, “API Function List,” on page 10](#)

1.1 Using the API in the MVS Environment

Access to the API in the MVS* (OS/390*, z/OS*) environment uses the ASCLIENT started task. The caller must be APF authorized. The C header file is located in the MVS Platform Services Samples Library member ASCAUTH. When link-editing applications that call the API, include the MVS Platform Services Load Library in your SYSLIB concatenation. Calls to `ASC_INIT()` and `ASC_INIT_EXT()` must be made with the NULL parameter in place of the configuration file name, because ASCLIENT processes the platform configuration file for the MVS environment.

For additional information about the MVS platform, see the *Platform Services Administration Guide for MVS*.

1.2 Using the API on a NetWare Server

Access to the C language AS Client API in the NetWare® environment is through calls to `ascauth.nlm`. Access to the API using Java is through calls to the methods of class `com.novell.asam.JAscAuth.JAscAuth`.

`Ascauth.nlm`, together with the files `ascauth.h` and `ascauth.imp` used for development in the C environment and `jascauth.jar` used in the Java environment, are distributed in the AS Client API

distribution package at bin\platformservices\platformclient in the ASAM directory created during Platform Services installation.

For additional information about the AS Client API on NetWare, see the *NetWare Intercept and API Administration Guide*.

1.3 Using the API in the OS/400 Environment

Access to the API using C in the OS/400* environment is provided by the libascauth *SRVPGM in the ASAM library. The C header file is located in /usr/local/ASAM/bin.

The caller must have read privileges to the /usr/local/ASAM/data/PlatformServices/certs directory.

For additional information about the OS/400 platform, see the *Platform Services Administration Guide for OS/400*.

1.4 Using the API in the UNIX Environment

Access to the API using C in the UNIX environment is through calls to the shared library. The shared library and the C header file ascauth.h are copied to system-specific directories during the UNIX Platform Services installation process.

Access to the API using Java is through calls to the methods of class com.novell.asam.JAscAuth.JAscAuth. The jascauth.jar file is copied to the ASAM/bin/PlatformServices/PlatformClient/Java directory during Platform Services installation.

The caller must have read access to the /usr/local/ASAM/data/PlatformServices/certs directory.

For additional information about the UNIX platform, see the *Platform Services Administration Guide for Linux and UNIX*.

1.5 API Function List

API routines are provided to perform the following functions:

- ♦ Initialize the environment.
 - ♦ **C:** “ASC_INIT” on page 29, “ASC_INIT_EXT” on page 31
 - ♦ **Java:** “init” on page 54
- ♦ Terminate the environment.
 - ♦ **C:** “ASC_TERM” on page 45
 - ♦ **Java:** “destroy” on page 52
- ♦ Validate a user ID and password combination .
 - ♦ **C:** “ASC_CHKPASSWD” on page 19
 - ♦ **Java:** “checkPassword” on page 52
- ♦ Change a user's password, given the current password .
 - ♦ **C:** “ASC_CHGPASSWD” on page 17
 - ♦ **Java:** “changePassword” on page 51

- ♦ Reset a user's password as an administrative user.
 - ♦ **C:** “ASC_ADMINRSTPASSWD” on page 14
 - ♦ **Java:** “adminResetPassword” on page 51
- ♦ Obtain the fully distinguished name for a user ID.
 - ♦ **C:** “ASC_GETCONTEXT” on page 23
 - ♦ **Java:** “getContext” on page 53
- ♦ Determine if a user has security equal to a given object.
 - ♦ **C:** “ASC_SECEQUAL” on page 42
 - ♦ **Java:** “securityEquals” on page 55
- ♦ Determine if an object has the specified effective rights to the specified attribute of another object.
 - ♦ **C:** “ASC_RIGHTS” on page 39
 - ♦ **Java:** “effectiveRights” on page 53
- ♦ Obtain a list of members of a group.
 - ♦ **C:** “ASC_GRPMEM” on page 26
 - ♦ **Java:** “groupMembers” on page 54
- ♦ Obtain a list of security equivalences for a user.
 - ♦ **C:** “ASC_LISTSEQV” on page 33
 - ♦ **Java:** “listSecurityEquivalences” on page 54
- ♦ Obtain attribute values for an object.
 - ♦ **C:** “ASC_READATTR” on page 36
 - ♦ **Java:** “readAttribute” on page 55
- ♦ Determine if a given user is in the Include/Exclude list.
 - ♦ **C:** “ASC_USER_INCLUDE_EXCLUDE” on page 47
 - ♦ **Java:** “userIncludeExclude” on page 56
- ♦ Decode API return values.
 - ♦ **C:** “ASC_STRERROR” on page 44
 - ♦ **Java:** “strError” on page 56
- ♦ Convert number of seconds to number of days.
 - ♦ **C:** “ASC_DAYS” on page 22
 - ♦ **Java:** “secondsToDays” on page 55

C Language API Reference

2

A description of each Novell® Identity Manager Fan-Out driver AS Client API function along with example C code follows.

- ♦ “ASC_ADMINRSTPASSWD” on page 14
- ♦ “ASC_CHGPASSWD” on page 17
- ♦ “ASC_CHKPASSWD” on page 19
- ♦ “ASC_DAYS” on page 22
- ♦ “ASC_GETCONTEXT” on page 23
- ♦ “ASC_GRPMEM” on page 26
- ♦ “ASC_INIT” on page 29
- ♦ “ASC_INIT_EXT” on page 31
- ♦ “ASC_LISTSEQV” on page 33
- ♦ “ASC_READATTR” on page 36
- ♦ “ASC_RIGHTS” on page 39
- ♦ “ASC_SECEQUAL” on page 42
- ♦ “ASC_STRERROR” on page 44
- ♦ “ASC_TERM” on page 45
- ♦ “ASC_USER_INCLUDE_EXCLUDE” on page 47

ASC_ADMINRSTPASSWD

Performs an administrative reset of a user's password. The new password is marked as being expired unless it is non-expiring.

Syntax

```
#include <ascauth.h>

int ASC_ADMINRSTPASSWD(ASCENV *asce, char *adminUser, char
*adminPassword,
                        char *user, char *newpass);
```

Parameters

asce	The environment item returned from the call to ASC_INIT() or ASC_INIT_EXT().
adminUser	The Enterprise User ID of an administrative user with rights to change the target user's password.
adminPassword	The password of the administrative user ID.
user	The Enterprise User ID whose password is to be changed.
newpass	The new password for the user.

Return Values

Returns one of the following integer values defined in ascauth.h:

AS_OK	Password changed
AS_NOUSER	User inactive or not found in the Census
AS_BADCLIENT	Local host not authorized to query the core driver
AS_NOAGENT	No core driver could be contacted
AS_NOAUTHENV	No environment has been established
AS_INVALIDREQ	Call rejected by the core driver as not valid or not supported
AS_INVALIDARGS	Invalid arguments supplied to the function
AS_KEYEXPIRED	Old key rejected by the core driver because the expiration date has passed
AS_INSUFFICIENTRIGHTS	Administrative user does not exist, administrative user does not have rights to change the password, or administrative user password not valid

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

main(int argc, char *argv[])
{
    ASCENV *asce;
    ASCUSER ascu;
    char *adminUser, *adminPass, *user, *newpass;
    int rc;

    if (argc != 5) {
        fprintf(stderr, "usage: %s <adminUser> <adminPass> <user>
<newpass>\n",
            argv[0]);
        exit(EXIT_FAILURE);
    }

    adminUser = argv[1];
    adminPass = argv[2];
    user      = argv[3];
    newpass   = argv[4];

    /* initialize the authentication environment */
    asce = ASC_INIT(NULL);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        exit(EXIT_FAILURE);
    }

    /* change the user's password */
    rc = ASC_ADMINRSTPASSWD(asce, adminUser, adminPass, user, newpass);
    if (rc == AS_OK)
        printf("password has been changed\n");
    else if (rc == AS_NO)
        printf("password has not been changed\n");
    else
        printf("RC=%d, %s", rc, ASC_STRERROR(rc));

    /* now terminate the authentication environment */
    ASC_TERM(asce);
    return 0;
}
```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_INIT_EXT” on page 31](#)

[“ASC_TERM” on page 45](#)

[“ASC_STRERROR” on page 44](#)

ASC_CHGPASSWD

Changes the password of a user.

Syntax

```
#include <ascauth.h>
int ASC_CHGPASSWD(ASCENV *asce, char *user, char *oldpass, char
*newpass);
```

Parameters

asce	The environment item returned from the call to ASC_INIT() or ASC_INIT_EXT().
user	The Enterprise User ID whose password is to be changed.
oldpass	The old password for the user.
newpass	The new password for the user.

Return Values

Returns one of the following integer values defined in ascauth.h:

AS_OK	Password changed
AS_NO	Old password is invalid
AS_NOUSER	User inactive or not found in the Census
AS_REVOKED	User's password is okay, but the user is disabled
AS_INTRUDER	Intruder lockout is enabled for this user
AS_PASSDUPLICATE	New password has been used previously
AS_PASSTOOSHORT	New password is too short
AS_BADCLIENT	Local host not authorized to query the core driver
AS_NOAGENT	No core driver could be contacted
AS_NOAUTHENV	No environment has been established
AS_INVALIDREQ	Call rejected by the core driver as not valid or not supported
AS_INVALIDARGS	Invalid arguments supplied to the function
AS_KEYEXPIRED	Old key rejected by the core driver because the expiration date has passed

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

main(int argc, char *argv[])
{
    ASCENV *asce;
    ASCUSER ascu;
    char *user, *oldpass, *newpass;
    int rc;

    if (argc != 4) {
        fprintf(stderr, "usage: %s <user> <oldpass> <newpass>\n",
            argv[0]);
        exit(EXIT_FAILURE);
    }

    user = argv[1];
    oldpass = argv[2];
    newpass = argv[3];

    /* initialize the authentication environment */
    asce = ASC_INIT(NULL);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        exit(EXIT_FAILURE);
    }

    /* change the user's password */
    rc = ASC_CHGPASSWD(asce, user, oldpass, newpass);
    if (rc == AS_OK)
        printf("password has been changed\n");
    else if (rc == AS_NO)
        printf("password has not been changed\n");
    else
        printf("RC=%d, %s", rc, ASC_STRERROR(rc));

    /* now terminate the authentication environment */
    ASC_TERM(asce);
    return 0;
}
```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_INIT_EXT” on page 31](#)

[“ASC_TERM” on page 45](#)

[“ASC_STRERROR” on page 44](#)

ASC_CHKPASSWD

Verifies the password of a user.

Syntax

```
#include <ascauth.h>
int ASC_CHKPASSWD(ASCENV *asce, char *user, char *pass, ASCUSER
*ascu);
```

Parameters

asce	The environment item returned from the call to ASC_INIT() or ASC_INIT_EXT().
user	The Enterprise User ID to be checked.
pass	The password to be checked for the user.
ascu	The ASCUSER structure (defined in ascauth.h) to be filled in by ASC_CHKPASSWD() if the password is valid.

Return Values

Returns one of the following integer values defined in ascauth.h:

AS_OK	Password is okay
AS_NO	User ID/Password combination is invalid
AS_NOUSER	User inactive or not found in the Census
AS_REVOKED	User's password is okay, but the user is disabled
AS_INTRUDER	Intruder lockout enabled for this user
AS_BADCLIENT	Local host is not authorized to query the core driver
AS_NOAGENT	No core driver could be contacted
AS_NOAUTHENV	No environment has been established
AS_INVALIDREQ	Call rejected by the core driver as not valid or not supported
AS_INVALIDARGS	Invalid arguments supplied to the function
AS_KEYEXPIRED	Old key rejected by the core driver because the expiration date has passed

If an AS_OK return code is returned, the following fields in the ASCUSER structure contain additional information about the authenticated user:

<ascu>.pass.expire	Number of seconds until the password expires (or -1 if the password does not expire)
--------------------	--

<ascu>.pass.interval	Password change interval in seconds (or -1 if the password does not expire)
----------------------	---

If an AS_REVOKED code is returned, the following field in the ASCUSER structure contains additional information about the user:

<ascu>.login.disabled	User disabled flag
-----------------------	--------------------

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

main(int argc, char *argv[])
{
    ASCENV *asce;
    ASCUSER ascu;
    char *user, *pass;
    int rc;

    if (argc != 3) {
        fprintf(stderr, "usage: %s <user> <password>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    user = argv[1];
    pass = argv[2];

    /* initialize the authentication environment */
    asce = ASC_INIT(NULL);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        exit(EXIT_FAILURE);
    }

    /* check the user's password */
    rc = ASC_CHKPASSWD(asce, user, pass, &ascu);
    if (rc == AS_OK)
        printf("password ok\n");
    else if (rc == AS_NO)
        printf("password invalid\n");
    else
        printf("RC=%d, %s", rc, ASC_STRERROR(rc));

    /* now terminate the authentication environment */
    ASC_TERM(asce);
    return 0;
}
```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_INIT_EXT” on page 31](#)

[“ASC_TERM” on page 45](#)

[“ASC_STRERROR” on page 44](#)

ASC_DAYS

Converts an integer number of seconds into an integer number of days.

Syntax

```
#include <ascauth.h>
long ASC_DAYS(long secs);
```

Parameters

secs	A number of seconds.
------	----------------------

Return Values

Returns the integer number of days corresponding to the given number of seconds.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

printf("*** CHKPASWD expire days=%ld, expire interval days=%ld\n",
        ASC_DAYS(ascu.pass.expire), ASC_DAYS(ascu.pass.interval));
```

ASC_GETCONTEXT

Obtains a user's fully distinguished object name from the Census and copies it into the buffer supplied by the caller.

Syntax

```
#include <ascauth.h>

int ASC_GETCONTEXT(ASCENV *asce, char *user, char *buffer, u_int
size);
```

Parameters

asce	The environment item returned from the call to ASC_INIT() or ASC_INIT_EXT().
user	The Enterprise User ID.
buffer	The buffer that is to receive the context. The result is truncated and the call returns AS_TOOSMALL if the buffer size cannot hold the entire result.
size	The length in bytes of the buffer.

Return Values

Returns one of the following integer values defined in ascauth.h:

AS_OK	Context was found
AS_NOUSER	User inactive or not found in the Census
AS_BADCLIENT	Local host not authorized to query the core driver
AS_NOAGENT	No core driver could be contacted
AS_NOAUTHENV	No environment has been established
AS_INVALIDREQ	Call rejected by the core driver as not valid or not supported
AS_INVALIDARGS	Invalid arguments supplied to the function
AS_TOOSMALL	Size of the pre-allocated buffer is too small-result truncated
AS_KEYEXPIRED	Old key rejected by the core driver because the expiration date has passed

Remarks

The buffer is padded with nulls if needed.

The format of the returned login context is the simple dot form. For example: .jondoe.j.myorg

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

#define MAX_CONTEXT 512

main(int argc, char *argv[])
{
    ASCENV *asce;
    char *user, *context;
    int rc;

    if (argc != 2) {
        fprintf(stderr, "usage: %s <user>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    user = argv[1];

    /* allocate buffer */
    context = (char *) malloc(MAX_CONTEXT);

    /* initialize the authentication environment */
    asce = ASC_INIT(NULL);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        exit(EXIT_FAILURE);
    }

    /* get the user's context */
    rc = ASC_GETCONTEXT(asce, user, context, MAX_CONTEXT);
    if (rc == AS_OK)
        printf("context is %s\n", context);
    else
        printf("RC=%d, %s", rc, ASC_STRERROR(rc));

    free(context);

    /* now terminate the authentication environment */
    ASC_TERM(asce);
    return 0;
}
```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_INIT_EXT” on page 31](#)

[“ASC_TERM” on page 45](#)

[“ASC_STRERROR” on page 44](#)

ASC_GRPMEM

Obtains a list of all members of the given group and places it in the buffer supplied by the caller.

Syntax

```
#include <ascauth.h>
int ASC_GRPMEM(ASCENV *asce, char *object, char *buf, u_int size);
```

Parameters

asce	The environment item returned from the call to ASC_INIT() or ASC_INIT_EXT().
object	The fully distinguished group name whose membership list is to be returned.
buf	The buffer in which the membership list is to be returned. Member names are separated by a newline '\n' character. The list is truncated and the call returns AS_TOOSMALL if the buffer size cannot hold the entire list.
size	The size of the buffer you provided.

Return Values

Returns one of the following integer values defined in ascauth.h:

AS_OK	Member list successfully returned
AS_BADCLIENT	Local host not authorized to query the core driver
AS_NOAGENT	No core driver could be contacted
AS_NOAUTHENV	No environment has been established
AS_INVALIDREQ	Call rejected by the core driver as not valid or not supported
AS_INVALIDARGS	Invalid arguments supplied to the function
AS_TOOSMALL	Size of the pre-allocated buffer is too small-list truncated
AS_INVALIDOBJ	Specified object does not exist
AS_KEYEXPIRED	Old key rejected by the core driver because the expiration date has passed

Remarks

The list is truncated, and the call returns AS_TOOSMALL if the buffer size cannot hold the entire list. You can retry with a larger buffer.

You can use ASC_SECEQUAL to see if an individual user is a member of a given group.

The groups a given user is a member of are included in the list returned by `ASC_LISTSEQV`.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

main(int argc, char *argv[])
{
    ASCENV *asce;
    char *group, buffer[2000];
    int rc;

    if (argc != 2) {
        fprintf(stderr, "usage: %s <group>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    group = argv[1];

    /* initialize the authentication environment */
    asce = ASC_INIT(NULL);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        exit(EXIT_FAILURE);
    }

    /* Get group membership info */
    rc = ASC_GRPMEM(asce, group, buffer, sizeof(buffer));
    if (rc == AS_OK)
        printf("Members of group %s:\n%s\n", group, buffer);
    else if (rc == AS_TOOSMALL) {
        printf("Members of group %s:\n%s\n", group, buffer);
        printf("*** list was truncated because of lack of buffer space
**\n");
    }
    else
        printf("RC=%d, %s", rc, ASC_STRERROR(rc));

    /* now terminate the authentication environment */
    ASC_TERM(asce);
    return 0;
}
```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_INIT_EXT” on page 31](#)

[“ASC_LISTSEQV” on page 33](#)

[“ASC_SECEQUAL” on page 42](#)

[“ASC_TERM” on page 45](#)

[“ASC_STRERROR” on page 44](#)

ASC_INIT

Reads the platform configuration file and initializes the environment so that calls can be made to a core driver. This function or ASC_INIT_EXT() must be called before any other API function.

Syntax

```
#include <ascauth.h>
ASCENV *ASC_INIT(char *filename);
```

Parameters

filename	The name of the platform configuration file. If you call ASC_INIT() with a NULL in place of the filename parameter as in ASC_INIT(NULL), the default is as follows: MVS: Always uses the ASCLIENT started task active configuration. NetWare: sys:asam\data\asamplat.conf OS/400: /usr/local/ASAM/data/asamplat.conf UNIX: /usr/local/ASAM/data/asamplat.conf
----------	--

Return Values

Returns a pointer to the API environment item created upon success. If an error has occurred, NULL is returned.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

main()
{
    ASCENV *asce;

    /* initialize the authentication environment */
    asce = ASC_INIT(NULL);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        exit(EXIT_FAILURE);
    }

    /* now you can make additional authentication calls here */

    /* now terminate the authentication environment */
    ASC_TERM(asce);
}
```

```
    return 0;  
}
```

See Also

[“ASC_INIT_EXT” on page 31](#)

[“ASC_TERM” on page 45](#)

ASC_INIT_EXT

Reads the platform configuration file and initializes the environment so that calls can be made to a core driver. This function or ASC_INIT() must be called before any other API function.

ASC_INIT_EXT() differs from ASC_INIT() in that you can provide a buffer into which the API places error messages if the API environment cannot be initialized.

Syntax

```
#include <ascauth.h>
ASCENV *ASC_INIT_EXT(char *filename, char *error_msg, size_t size);
```

Parameters

filename	<p>The name of the platform configuration file.</p> <p>If you call ASC_INIT_EXT() with a NULL in place of the filename parameter as in ASC_INIT_EXT(NULL, buffer, BUFSIZE), the default is as follows:</p> <p>MVS: Always uses the ASCLIENT started task active configuration.</p> <p>NetWare: sys:asam\data\asamplat.conf</p> <p>OS/400: /usr/local/ASAM/data/asamplat.conf</p> <p>UNIX: /usr/local/ASAM/data/asamplat.conf</p>
error_msg	<p>A buffer you provide into which an error message can be placed if the environment cannot be initialized.</p>
size	<p>The size of the error_msg buffer you have provided.</p>

Return Values

Returns a pointer to the environment item created upon success. If an error has occurred, NULL is returned, and a descriptive error message is placed into the error_msg buffer.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

#define BUFSIZE 256

main()
{
    ASCENV *asce;

    /* initialize the authentication environment */
    /* allocate buffer */
    buffer = (char *) malloc(BUFSIZE);
```

```

    asce = ASC_INIT_EXT(NULL, buffer, BUFSIZE);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        fprintf(stderr, "  %s \n", buffer);
        exit(EXIT_FAILURE);
    }

    /* now you can make additional authentication calls  here */

    /* now terminate the authentication environment */
    ASC_TERM(asce);
    return 0;
}

```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_TERM” on page 45](#)

ASC_LISTSEQV

Obtains a user's Security Equals attribute list and places it in the buffer supplied by the caller.

Syntax

```
#include <ascauth.h>
int ASC_LISTSEQV(ASCENV *asce, char *user, char *buf, u_int size);
```

Parameters

asce	The environment item returned from the call to ASC_INIT() or ASC_INIT_EXT().
user	The Enterprise User ID whose Security Equals list is to be returned.
buf	The buffer in which the security equivalence list is to be returned. Object names are separated by a newline '\n' character. The list is truncated and the call returns AS_TOOSMALL if the buffer size cannot hold the entire list.
size	The size of the buffer you provided.

Return Values

Returns one of the following integer values defined in ascauth.h:

AS_OK	Security equivalence list successfully returned
AS_NOUSER	User inactive or not found in the Census
AS_BADCLIENT	Local host is not authorized to query the core driver
AS_NOAGENT	No core driver could be contacted
AS_NOAUTHENV	No environment has been established
AS_INVALIDREQ	Call rejected by the core driver as not valid or not supported
AS_INVALIDARGS	Invalid arguments supplied to the function
AS_TOOSMALL	Size of pre-allocated buffer is too small-the list is truncated
AS_INVALIDOBJ	Specified object does not exist
AS_KEYEXPIRED	Old key rejected by the core driver because the expiration date has passed

Remarks

The list is truncated, and the call returns AS_TOOSMALL if the buffer size cannot hold the entire list. You can retry with a larger buffer.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

main(int argc, char *argv[])
{
    ASCENV *asce;
    char *object, buffer[2000];
    int rc;

    if (argc != 2) {
        fprintf(stderr, "usage: %s <object>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    object = argv[1];

    /* initialize the authentication environment */
    asce = ASC_INIT(NULL);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        exit(EXIT_FAILURE);
    }

    /* Get security equivalence info */
    rc = ASC_LISTSEQV(asce, object, buffer, sizeof(buffer));
    if (rc == AS_OK)
        printf("Security equivalences of object %s:\n%s\n", object,
buffer);
    else if (rc == AS_TOOSMALL) {
        printf("Security equivalences of object %s:\n%s\n", object,
buffer);
        printf("*** list was truncated because of lack of buffer space
**\n");
    }
    else
        printf("RC=%d, %s", rc, ASC_STRERROR(rc));

    /* now terminate the authentication environment */
    ASC_TERM(asce);
    return 0;
}
```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_INIT_EXT” on page 31](#)

[“ASC_TERM” on page 45](#)

“ASC_STRERROR” on page 44

ASC_READATTR

Returns the value of the specified single-valued attribute for the specified object.

Syntax

```
#include <ascauth.h>
int ASC_READATTR(ASCENV *asce, char *object, char *attribute,
                 char *buffer, u_int bufsize);
```

Parameters

asce	The environment item returned from the call to ASC_INIT() or ASC_INIT_EXT().
object	The Enterprise User ID or fully distinguished object name of the object whose attribute value is to be returned.
attribute	The single-valued attribute whose value is to be returned for the object. Only the Home Directory attribute of a User object is supported at this time.
buffer	The buffer in which the object's attribute value is to be returned. The results are truncated and the call returns AS_TOOSMALL if the buffer size cannot hold the entire attribute value.
bufsize	The size of the buffer you provided.

Return Values

Returns one of the following integer values defined in ascauth.h:

AS_OK	Attribute value has been placed in the buffer successfully
AS_BADCLIENT	Local host not authorized to query the core driver
AS_ATTRNOTFOUND	Attribute does not exist for the specified object
AS_NOAGENT	No core driver could be contacted
AS_NOAUTHENV	No environment has been established
AS_INVALIDREQ	Call rejected by the core driver as not valid or not supported
AS_INVALIDARGS	Invalid arguments supplied to the function
AS_TOOSMALL	Size of the pre-allocated buffer is too small-results are truncated
AS_INVALIDOBJ	Specified object does not exist
AS_KEYEXPIRED	Old key rejected by the core driver because the expiration date has passed

Remarks

The results are truncated, and the call returns `AS_TOOSMALL` if the buffer size cannot hold the entire attribute value. You can retry with a larger buffer.

Limitations

Only the Home Directory attribute of a User object is supported at this time.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

main(int argc, char *argv[])
{
    ASCENV *asce;
    char *user, buffer[2000];
    int rc;

    if (argc != 2) {
        fprintf(stderr, "usage: %s <UserObjectFDN>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    user = argv[1];

    /* initialize the authentication environment */
    asce = ASC_INIT(NULL);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        exit(EXIT_FAILURE);
    }

    /* Get User object's home directory info */
    rc = ASC_READATTR(asce, user, "HOME DIRECTORY", buffer,
sizeof(buffer));
    if (rc == AS_OK)
        printf("Home Directory for User object %s:\n%s\n", user, buffer);
    else
        printf("RC=%d, %s", rc, ASC_STRERROR(rc));

    /* now terminate the authentication environment */
    ASC_TERM(asce);
    return 0;
}
```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_INIT_EXT” on page 31](#)

[“ASC_TERM” on page 45](#)

[“ASC_STRERROR” on page 44](#)

ASC_RIGHTS

Checks the specified effective rights of one object over another for a specific attribute.

Syntax

```
#include <ascauth.h>
int ASC_RIGHTS(ASCENV *asce, char *obj1, char *obj2,
               char *attribute, char *rights);
```

Parameters

asce	The environment item returned from the call to ASC_INIT() or ASC_INIT_EXT().
obj1	The Enterprise User ID or fully distinguished object name whose effective rights are to be tested.
obj2	The Enterprise User ID or fully distinguished object name for which access by obj1 is to be tested.
attribute	The name of an attribute of obj2 for which the effective rights of obj1 are requested. The special attribute names All Attributes Rights, Entry Rights, and SMS Rights can also be specified.
rights	The rights to test. The characters specified must be in the following set: [S,C,R,W,A]. These correspond to Supervisor, Compare, Read, Write, and Add Self.

Return Values

Returns one of the following integer values defined in ascauth.h:

AS_OK	User or object has the specified rights to the specified object attribute
AS_NO	User or object does not have the specified rights to the specified object attribute
AS_ATTRNOTFOUND	Specified attribute could not be found
AS_INVALIDOBJ	Specified user not found in the Census or the specified object does not exist
AS_INVALIDOBJLEN	Specified object exceeds maximum length
AS_BADCLIENT	Local host not authorized to query the core driver
AS_NOAGENT	No core driver could be contacted
AS_NOAUTHENV	No environment has been established
AS_INVALIDREQ	Call rejected by the core driver as not valid or not supported
AS_INVALIDARGS	Invalid arguments supplied to the function

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

main(int argc, char *argv[])
{
    ASCENV *asce;
    char *obj1, *obj2, *attr, *rights;
    int rc;

    if (argc != 5) {
        fprintf(stderr, "usage: %s <obj1> <obj2> \
            <attribute> <rights>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    obj1   = argv[1];
    obj2   = argv[2];
    attr   = argv[3];
    rights = argv[4];

    /* initialize the authentication environment */
    asce = ASC_INIT(NULL);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        exit(EXIT_FAILURE);
    }

    /* check for rights */
    rc = ASC_RIGHTS(asce, obj1, obj2, attr, rights);
    if (rc == AS_OK)
        printf("User has rights\n");
    else
        printf("RC=%d, %s", rc, ASC_STRERROR(rc));

    /* now terminate the authentication environment */
    ASC_TERM(asce);
    return 0;
}
```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_INIT_EXT” on page 31](#)

[“ASC_TERM” on page 45](#)

[“ASC_STRERROR” on page 44](#)

ASC_SECEQUAL

Checks to see if a user has security equivalence to the specified object.

Syntax

```
#include <ascauth.h>
int ASC_SECEQUAL(ASCENV *asce, char *user, char *object);
```

Parameters

asce	The environment item returned from the call to ASC_INIT() or ASC_INIT_EXT().
user	The Enterprise User ID to be tested.
Object	The fully distinguished object name to test the user for security equivalence.

Return Values

Returns one of the following integer values defined in ascauth.h:

AS_OK	User has security equivalence to the specified object
AS_NO	User does not have security equivalence to the object
AS_NOUSER	User inactive or not found in the Census
AS_BADCLIENT	Local host not authorized to query the core driver
AS_NOAGENT	No core driver could be contacted
AS_NOAUTHENV	No environment has been established
AS_INVALIDREQ	Call rejected by the core driver as not valid or not supported
AS_INVALIDARGS	Invalid arguments supplied to the function
AS_INVALIDOBJ	Specified object does not exist
AS_KEYEXPIRED	Old key rejected by the core driver because the expiration date has passed

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

main(int argc, char *argv[])
{
    ASCENV *asce;
    char *user, *object;
```

```

int rc;

if (argc != 3) {
    fprintf(stderr, "usage: %s <user> <object>\n", argv[0]);
    exit(EXIT_FAILURE);
}
user = argv[1];
object = argv[2];

/* initialize the authentication environment */
asce = ASC_INIT(NULL);
if (asce == NULL) {
    fprintf(stderr, "Error: cannot initialize authentication
environment\n");
    exit(EXIT_FAILURE);
}

/* check for security equivalence */
rc = ASC_SECEQUAL(asce, user, object);
if (rc == AS_OK)
    printf("User has security equivalence\n");
else
    printf("RC=%d, %s", rc, ASC_STRERROR(rc));

/* now terminate the authentication environment */
ASC_TERM(asce);
return 0;
}

```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_INIT_EXT” on page 31](#)

[“ASC_TERM” on page 45](#)

[“ASC_STRERROR” on page 44](#)

ASC_STRERROR

Returns the error string for the specified ASC function error code.

Syntax

```
#include <ascauth.h>
const char *ASC_STRERROR(int errnum);
```

Parameters

errnum	The error return value from a call to an ASC_ function.
--------	---

Return Values

Returns a static character string corresponding to the integer errnum value as defined in ascauth.h for ASC function error codes.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

rc = ASC_CHKPASSWD(asce, userid, password, &ascu);
strcpy(status, ASC_STRERROR(rc));
printf("\n*** CHKPASSWD return code = %d (%s)\n", rc, status);
```

ASC_TERM

Terminates and frees the environment that was created by a call to `ASC_INIT()` or `ASC_INIT_EXT()`. After the environment is terminated, no more calls to the core driver can be made without first issuing another `ASC_INIT()` or `ASC_INIT_EXT()` call.

Syntax

```
#include <ascauth.h>

void ASC_TERM(ASCENV *asce);
```

Parameters

<code>asce</code>	The environment item returned from the call to <code>ASC_INIT()</code> or <code>ASC_INIT_EXT()</code> .
-------------------	---

Return Values

No value is returned from this function.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

main()
{
    ASCENV *asce;

    /* initialize the authentication environment */
    asce = ASC_INIT(NULL);
    if (asce == NULL) {
        fprintf(stderr, "Error: cannot initialize authentication
environment\n");
        exit(EXIT_FAILURE);
    }

    /* now you can make additional authentication calls here */

    /* now terminate the authentication environment */
    ASC_TERM(asce);
    return 0;
}
```

See Also

[“ASC_INIT” on page 29](#)

“ASC_INIT_EXT” on page 31

ASC_USER_INCLUDE_EXCLUDE

Determines if a given user matches an AS.USER.INCLUDE or AS.USER.EXCLUDE statement in the platform configuration file.

Syntax

```
#include <ascauth.h>
int ASC_USER_INCLUDE_EXCLUDE(ASCENV *asce, char *user);
```

Parameters

asce	The environment item returned from the call to ASC_INIT() or ASC_INIT_EXT().
user	The Enterprise User ID of the user to be checked.

Return Values

Returns one of the following integer values defined in ascauth.h:

AS_NOMATCH	The user does not match any INCLUDE/EXCLUDE statement. Because "AS.USER.INCLUDE *" is implicit in the absence of AS.USER.EXCLUDE *, the user is included.
AS_INCLUDED	User matches an AS.USER.INCLUDE statement.
AS_EXCLUDED	User matches an AS.USER.EXCLUDE statement or an entry in the built-in standard exclude list.
AS_NOAUTHENV	No environment has been established.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <ascauth.h>

rc = ASC_USER_INCLUDE_EXCLUDE(asce, userid);
if (rc == AS_NOMATCH)
    printf("%s does not match an Include or Exclude statement\n",
userid);
else if (rc == AS_INCLUDED)
    printf("%s matches an Include statement\n", userid);
else if (rc == AS_EXCLUDED)
    printf("%s matches an Exclude statement\n", userid);
else
    printf("RC=%d, %s", rc, ASC_STRERROR(rc));
```

See Also

[“ASC_INIT” on page 29](#)

[“ASC_INIT_EXT” on page 31](#)

A description of the Novell® Identity Manager Fan-Out driver AS Client API Java implementation follows.

- ♦ “Class `com.novell.asam.JAscAuth.JAscAuth`” on page 50
 - ♦ “`adminResetPassword`” on page 51
 - ♦ “`changePassword`” on page 51
 - ♦ “`checkPassword`” on page 52
 - ♦ “`destroy`” on page 52
 - ♦ “`effectiveRights`” on page 53
 - ♦ “`getContext`” on page 53
 - ♦ “`getLastReturnCode`” on page 54
 - ♦ “`groupMembers`” on page 54
 - ♦ “`init`” on page 54
 - ♦ “`listSecurityEquivalences`” on page 54
 - ♦ “`readAttribute`” on page 55
 - ♦ “`secondsToDays`” on page 55
 - ♦ “`securityEquals`” on page 55
 - ♦ “`strError`” on page 56
 - ♦ “`userIncludeExclude`” on page 56
- ♦ “Classes Used by `checkPassword`” on page 58
- ♦ “Exception Classes in `com.novell.asam.JAscAuth`” on page 60

To view the reference documentation in JavaDoc format, see the `asam\bin\platformservices\platformclient\java\javadoc` directory on the platform system.

For code examples, see the `asam\bin\platformservices\platformclient\java` directory on the platform system.

Class com.novell.asam.JAscAuth.JAscAuth

Provides the methods you use to access the AS Client API.

Constructor

```
public JAscAuth()
```

Fields

The following fields map the AS Client API return codes. For more information about return codes from the AS Client API, see [Appendix A, “Troubleshooting the API,” on page 65](#).

```
public static int AS_OK = 0
public static int AS_NO = 1
public static int AS_NOUSER = 2
public static int AS_NOAGENT= 3
public static int AS_NOSERVER = 3
public static int AS_BADCLIENT= 4
public static int AS_REVOKED= 5
public static int AS_INTRUDER= 6
public static int AS_INVALIDARGS= 7
public static int AS_INVALIDOBJ= 8
public static int AS_INVALIDOBJLEN= 9
public static int AS_PASSDUPLICATE= 10
public static int AS_PASSTOOSHORT= 11
public static int AS_TOOSMALL= 12
public static int AS_ATTRNOTFOUND= 13
public static int AS_WSOCKUP= 14
public static int AS_WSOCKDOWN= 15
public static int AS_NOAUTHENV= 16
public static int AS_PRODUCTEXPIRED = 17
public static int AS_INCLUDED= 18
public static int AS_EXCLUDED= 19
public static int AS_NOMATCH= 20
public static int AS_NOLICENSE= 21
public static int AS_INVALIDREQ= 22
public static int AS_KEYEXPIRED= 23
```

Methods

The following methods invoke the API functions:

- ♦ [“adminResetPassword” on page 51](#)
- ♦ [“changePassword” on page 51](#)
- ♦ [“checkPassword” on page 52](#)
- ♦ [“destroy” on page 52](#)

- ♦ [“effectiveRights” on page 53](#)
- ♦ [“getContext” on page 53](#)
- ♦ [“getLastReturnCode” on page 54](#)
- ♦ [“groupMembers” on page 54](#)
- ♦ [“init” on page 54](#)
- ♦ [“listSecurityEquivalences” on page 54](#)
- ♦ [“readAttribute” on page 55](#)
- ♦ [“secondsToDays” on page 55](#)
- ♦ [“securityEquals” on page 55](#)
- ♦ [“strError” on page 56](#)
- ♦ [“userIncludeExclude” on page 56](#)

adminResetPassword

Performs an administrative reset of a user's password. The new password is marked as being expired unless it is non-expiring.

You must call the `init` method to initialize the `JAscAuth` environment before calling `adminResetPassword`. For more information about `init`, see [“init” on page 54](#).

For details about the exceptions that can be thrown, see [“Exception Classes in com.novell.asam.JAscAuth” on page 60](#).

Syntax

```
public void adminResetPassword(
    java.lang.String adminUser,
    java.lang.String adminPass,
    java.lang.String user,
    java.lang.String pass)
```

Parameters

<code>adminUser</code>	The Enterprise User ID of an administrative user with rights to change the target user's password
<code>adminPass</code>	The password of the administrative user
<code>user</code>	The Enterprise User ID whose password is to be changed
<code>pass</code>	The new password for the user

changePassword

Changes the password of a user.

You must call the `init` method to initialize the `JAscAuth` environment before calling `changePassword`. For more information about `init`, see [“init” on page 54](#).

For details about the exceptions that can be thrown, see [“Exception Classes in com.novell.asam.JAscAuth” on page 60](#).

Syntax

```
public void changePassword(  
    String user,  
    String oldPass,  
    String newPass)
```

Parameters

user	The Enterprise User ID whose password is to be changed
oldPass	The old password for the user
newPass	The new password for the user

checkPassword

Verifies the password of a user.

You must call the `init` method to initialize the JAscAuth environment before calling `checkPassword`. For more information about `init`, see [“init” on page 54](#).

The `checkPassword` method can optionally return information about the user and password in a JAscUser object. For details about the contents of JAscUser, see [“Classes Used by checkPassword” on page 58](#).

For details about the exceptions that can be thrown, see [“Exception Classes in com.novell.asam.JAscAuth” on page 60](#).

Syntax

```
public void checkPassword(  
    String user,  
    String pass)  
public void checkPassword(  
    String user,  
    String pass,  
    JAscUser ascuser)
```

Parameters

user	The Enterprise User ID whose password is to be verified
pass	The password to be verified for the user
ascuser	A JAscUser object to be filled with information about the user and password

destroy

Destroys the JAscAuth environment and frees its underlying resources.

Syntax

```
public void destroy()
```

See Also

[“init” on page 54](#)

effectiveRights

Checks the effective rights of one object over another for a specific attribute.

You must call the `init` method to initialize the `JAscAuth` environment before calling `effectiveRights`. For more information about `init`, see [“init” on page 54](#).

For details about the exceptions that can be thrown, see [“Exception Classes in com.novell.asam.JAscAuth” on page 60](#).

Syntax

```
public void effectiveRights(  
    String user,  
    String object,  
    String attribute,  
    String rights)
```

Parameters

user	The Enterprise User ID or fully distinguished object name whose effective rights are to be tested
object	The Enterprise User ID or fully distinguished object name for which access by user is to be tested
attribute	The name of an attribute of object for which the effective rights of user are tested. The special attribute names All Attributes Rights, Entry Rights, and SMS Rights can also be specified.
rights	The rights to test. The characters specified must be in the following set: [S,C,R,W,A]. These correspond to Supervisor, Compare, Read, Write, and Add Self.

getContext

Returns the fully distinguished object name from the Census for a given user.

You must call the `init` method to initialize the `JAscAuth` environment before calling `getContext`. For more information about `init`, see [“init” on page 54](#).

For details about the exceptions that can be thrown, see [“Exception Classes in com.novell.asam.JAscAuth” on page 60](#).

Syntax

```
public String getContext(String user)
```

Parameters

user	The Enterprise User ID whose context is to be returned
------	--

getLastReturnCode

Returns the return code from the last call to the AS Client API.

For details about return codes from the AS Client API, see [Appendix A, “Troubleshooting the API,” on page 65](#).

Syntax

```
public int getLastReturnCode()
```

See Also

[“strError” on page 56](#)

groupMembers

Returns an enumeration of all members of a given Group.

You must call the init method to initialize the JAscAuth environment before calling groupMembers. For more information about init, see [“init” on page 54](#).

For details about the exceptions that can be thrown, see [“Exception Classes in com.novell.asam.JAscAuth” on page 60](#).

Syntax

```
public Enumeration groupMembers(String group)
```

Parameters

group	The Enterprise Group or fully distinguished Group object name whose members are to be returned
-------	--

init

Initializes the JAscAuth environment using the platform configuration file.

You can optionally specify the location of the platform configuration file to be used. If you do not specify the location of the platform configuration file, the default platform configuration file is used.

Call the destroy method to free the JAscAuth environment and its underlying resources when you are finished. For more information about destroy, see [“destroy” on page 52](#).

Syntax

```
public void init()
public void init(java.lang.String filename)
```

Parameters

filename	The path name of the platform configuration file to use
----------	---

listSecurityEquivalences

Returns an enumeration of a given user's security equivalences.

You must call the `init` method to initialize the JAscAuth environment before calling `listSecurityEquivalences`. For more information about `init`, see [“init” on page 54](#).

For details about the exceptions that can be thrown, see [“Exception Classes in com.novell.asam.JAscAuth” on page 60](#).

Syntax

```
public Enumeration listSecurityEquivalences(String user)
```

Parameters

user	The Enterprise User ID whose Security Equals attribute values are to be returned
------	--

readAttribute

Returns an enumeration of the values of a specified attribute for a given object.

You must call the `init` method to initialize the JAscAuth environment before calling `readAttribute`. For more information about `init`, see [“init” on page 54](#).

For details about the exceptions that can be thrown, see [“Exception Classes in com.novell.asam.JAscAuth” on page 60](#).

Syntax

```
public Enumeration readAttribute(  
    String object,  
    String attribute)
```

Parameters

object	The Enterprise User ID or fully distinguished object name of the object whose attribute values are to be returned
attribute	The single-valued attribute whose value is to be returned for the object. Only the Home Directory attribute of a User object is supported at this time.

secondsToDays

Returns the integer number of days for the given number of seconds.

Syntax

```
public long secondsToDays(long secs)
```

securityEquals

Checks to see if a user has security equivalence to the specified object.

You must call the `init` method to initialize the JAscAuth environment before calling `securityEquals`. For more information about `init`, see [“init” on page 54](#).

For details about the exceptions that can be thrown, see [“Exception Classes in com.novell.asam.JAscAuth” on page 60](#).

Syntax

```
public void securityEquals(  
    String user,  
    String object)
```

Parameters

user	The Enterprise User ID to be tested
object	The fully distinguished object name for which the security equivalence of user is to be tested

strError

Returns the string representation of the given AS Client API return code.

Syntax

```
public String strError(int rc)
```

Parameters

rc	The AS Client API return code value whose string representation is to be returned
----	---

See Also

[“getLastReturnCode” on page 54](#)

userIncludeExclude

Determines if a given user matches an AS.USER.INCLUDE or AS.USER.EXCLUDE statement in the platform configuration file.

Syntax

```
public int userIncludeExclude(String user)
```

Parameters

user	The Enterprise User ID of the user to be checked
------	--

Return Values

AS_NOMATCH	The user does not match any INCLUDE/EXCLUDE statement. Because AS.USER.INCLUDE * is implicit in the absence of AS.USER.EXCLUDE *, the user is included.
AS_INCLUDED	User matches an AS.USER.INCLUDE statement.

AS_EXCLUDED	User matches an AS.USER.EXCLUDE statement or an entry in the built-in standard exclude list.
-------------	--

Classes Used by checkPassword

The following topics describe classes used by the checkPassword method of JAscAuth to return information.

Class com.novell.asam.JAscAuth.JAscUser

The checkPassword method of JAscAuth optionally returns a JAscUser object with information about the user being authenticated.

Constructor

```
public JAscUser()
```

Fields

public JAscLoginRestrict login	Contains the user's login disabled flag
public JAscPassRestrict pass	Contains the user's password expiration information

Class com.novell.asam.JAscAuth.JAscLoginRestrict

The checkPassword method of JAscAuth optionally returns a JAscUser object with information about the user being authenticated. One of the fields in JAscUser is a JAscLoginRestrict object, which contains the user's login disabled flag.

Constructor

```
public JAscLoginRestrict()
```

Fields

public int disabled	The user's login disabled flag
---------------------	--------------------------------

Methods

public int getDisabled()	Returns the user's login disabled flag
--------------------------	--

Class com.novell.asam.JAscAuth.JAscPassRestrict

The checkPassword method of JAscAuth optionally returns a JAscUser object with information about the user being authenticated. One of the fields in JAscUser is a JAscPassRestrict object, which contains password expiration information.

Constructor

```
public JAscPassRestrict()
```

Fields

<code>public long interval</code>	The password change interval in seconds (or -1 if the password does not expire)
<code>public long expire</code>	The number of seconds until the password expires (or -1 if the password does not expire)

Methods

<code>public long getInterval()</code>	Returns the password change interval in seconds (or -1 if the password does not expire)
<code>public long getExpire()</code>	Returns the number of seconds until the password expires (or -1 if the password does not expire)

Exception Classes in com.novell.asam.JAscAuth

The following exceptions, along with java/lang/NullPointerException, are the exceptions that are thrown by the methods of JAscAuth.

InvalidJAscException

Thrown when a method requires an authentication environment, but a valid authentication environment does not exist.

Most methods of com.novell.asam.JAscAuth.JAscAuth require that you call the init method before you call them. InvalidJAscException is thrown if you do not do so.

Corresponds to a return code of 16, AS_NOAUTHENV, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscAttrNotFoundException

Thrown when the attribute specified to the readAttr method was not found for the specified object.

Corresponds to a return code of 13, AS_ATTRNOTFOUND, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscAuthenticationException

Thrown when the password specified to the checkPassword method is not valid.

Corresponds to a return code of 1, AS_NO, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscBadClientException

Thrown when the network address used by the platform to contact a core driver for a method call does not match the network address listed in the Platform Configuration object in the ASAM System container.

Corresponds to a return code of 4, AS_BADCLIENT, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscChangePasswordException

Thrown by changePassword when the password cannot be changed.

Also thrown by changePassword if the old password given is not valid.

Corresponds to a return code of 1, AS_NO, and a return code of 4, AS_BADCLIENT, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscException

Thrown by most method calls when an unexpected or indeterminate error condition occurs.

JAscInsufficientRightsException

Thrown by `adminResetPassword` if the administrative user does not exist, if the administrative user password specified is not valid, or if the administrative user does not have rights to change the password.

Also thrown by `adminResetPassword` if the network address used by the platform to contact a core driver does not match the network address listed in the Platform Configuration object in the ASAM System container.

Corresponds to a return code of 24, `AS_INSUFFICIENTRIGHTS` from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscIntruderException

Thrown by `checkPassword` and `changePassword` when the specified user is locked because of intruder detection.

Corresponds to a return code of 6, `AS_INTRUDER`, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscInvalidArgsException

Thrown when a parameter passed to a method is null or not valid.

Corresponds to a return code of 7, `AS_INVALIDARGS`, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscInvalidObjException

Thrown when an object passed to a method is not found or is not of the correct type.

Corresponds to a return code of 8, `AS_INVALIDOBJ`, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscInvalidObjLenException

Thrown when an object name passed to a method is longer than the maximum allowable name.

Corresponds to a return code of 9, `AS_INVALIDOBJLEN`, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscInvalidReqException

Thrown when a method call is not known by the core driver.

Corresponds to a return code of 22, `AS_INVALIDREQ`, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscKeyExpiredException

Thrown when the DES encryption key used by a non-SSL platform has expired.

Corresponds to a return code of 23, AS_KEYEXPIRED, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscNoAgentException

Thrown when no core driver could be contacted to process a method call.

Corresponds to a return code of 3, AS_NOAGENT, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscNoUserException

Thrown when the user specified to a method call is inactive or not in the Census.

Corresponds to a return code of 2, AS_NOUSER, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscPassDuplicateException

Thrown by changePassword when the new password has been previously used for the user object, and the user is required to use unique passwords.

Corresponds to a return code of 10, AS_PASSDUPLICATE, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscPassTooShortException

Thrown by changePassword when the new password is shorter than the minimum password length set for the user.

Corresponds to a return code of 11, AS_PASSTOOSHORT, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscProductExpiredException

Thrown when the expiration date for the platform has passed.

Corresponds to a return code of 17, AS_PRODUCTEXPIRED, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

JAscRevokedException

Thrown by checkPassword and changePassword when the specified user is disabled.

Corresponds to a return code of 5, AS_REVOKED, from the AS Client API. For more information, see [Appendix A, “Troubleshooting the API,” on page 65](#).

The following topics describe simple modifications to several popular products to enable them for use with the Novell® Identity Manager Fan-Out driver.

- ♦ [Section 4.1, “Adding API Support to the Apache Web Server,” on page 63](#)
- ♦ [Section 4.2, “Adding API Support to the QUALCOMM POP Server,” on page 63](#)
- ♦ [Section 4.3, “Adding API Support to SASL,” on page 63](#)
- ♦ [Section 4.4, “Adding API Support to SSH Secure Shell,” on page 64](#)
- ♦ [Section 4.5, “Adding API Support to TACACS+,” on page 64](#)

4.1 Adding API Support to the Apache Web Server

The Apache HTTP Web Server software is one of the most popular Web servers in use today. It is developed by the Apache Group and can be downloaded free from the [Apache Software Foundation Web site \(http://www.apache.org\)](http://www.apache.org). Apache provides the facility to configure additional modules to handle specific functions, such as user authentication and locating a user's home directory.

You can install Platform Services on your Apache server and configure Apache to authenticate users using the AS Client API. You can also configure Apache to use the AS Client API to find a user's home directory on a NetWare® file system that is mounted on a Linux* Apache server. Example modules are provided in the ASAM/bin/PlatformServices/PlatformClient/Apache directory created by the Platform Services installation process.

4.2 Adding API Support to the QUALCOMM POP Server

QUALCOMM* Incorporated distributes freeware UNIX POP3 software known as Qpopper* in C source form. With no modifications, Qpopper can use Authentication Services for authentication through PAM. Qpopper can also be modified to use Authentication Services for authentication through the AS Client API.

You can obtain Qpopper from the [QUALCOMM Web site \(http://www.eudora.com\)](http://www.eudora.com).

You can install Platform Services on your POP server and use Qpopper to authenticate users using Authentication Services through PAM or through the API.

Directions for modifying Qpopper to use the AS Client API can be found in the ASAM/bin/PlatformServices/PlatformClient/POP directory created by the Platform Services installation process.

4.3 Adding API Support to SASL

SASL, the Simple Authentication and Security Layer, is a generic authentication protocol. Many connection-based protocols, such as SMTP, LDAP, IMAP, and POP3, support SASL. New authentication mechanisms that support SASL are automatically supported by those protocols.

Furthermore, protocols that use SASL for authentication support Kerberos* authentication through the Generic Security Services Application Programming Interface (GSSAPI).

A common open-source SASL library is Cyrus SASL, which is available at <ftp://ftp.andrew.cmu.edu/pub/cyrus-mail>.

Directions for modifying Cyrus SASL to use the AS Client API for authentication can be found in the ASAM/bin/PlatformServices/PlatformClient/sasl directory created by the Platform Services installation process.

4.4 Adding API Support to SSH Secure Shell

SSH* Communications Security produces a secure login application known as SSH Secure Shell. Source is available at <ftp://ftp.ssh.com/pub/ssh>. SSH Secure Shell is a commercial product. The rules governing the commercial and non-commercial use of SSH Secure Shell can be found at the [SSH Communications Security Web Site \(http://www.ssh.com\)](http://www.ssh.com).

You can install Platform Services on your server and modify the Secure Shell daemon, sshd, to use the AS Client API to authenticate users.

The sshd using the -Identity Manager Fan-Out driver allows users to skip setting up passphrases, because the authentication stage of setting up the Secure Shell connection is done with the driver instead of public-private key cryptography. After you have authenticated, your Secure Shell session is securely encrypted, as normal.

The Identity Manager Fan-Out driver provides sample instructions for modifying the Secure Shell sshunixuser.c module. These instructions are distributed in the ASAM/bin/PlatformServices/PlatformClient/SSH directory created by the Platform Services installation process.

4.5 Adding API Support to TACACS+

TACACS+ is a security protocol designed by Cisco Systems*, Inc. that is used to control dial-up access into networks. An unsupported but freely available implementation of TACACS+ is available in <ftp://ftpeng.cisco.com> in pub/tacacs.

You can install Platform Services on your server and modify TACACS+ to use the AS Client API to authenticate users.

Directions for modifying TACACS+ to use the AS Client API for authentication can be found in the ASAM/bin/PlatformServices/PlatformClient/TACACS directory created by the Platform Services installation process.

Troubleshooting the API

A

Most calls to the Novell® Identity Manager Fan-Out driver AS Client API return a value that describes the outcome of the call. These return code values are defined in the C language `ascauth.h` header file and are provided as fields in the `JAscAuth` class in the Java interface. The C language API function `ASC_STRERROR()` and the Java interface method `strError()` can be used to return a text string that corresponds to the return code. This text string is included in many of the messages that are written to the platform log file for errors involving API calls.

The Java interface uses exceptions for most non-affirmative API call outcomes.

The following table lists the return codes and their corresponding text string, and suggests actions to take for them.

Return Code	Symbol Text String	Explanation and Suggested Action
0	AS_OK Action successful	The operation returned a positive response. For calls such as check password, this corresponds to an answer of "Yes."
1	AS_NO Action not successful	The operation returned a negative response. For calls such as check password, this corresponds to an answer of "No."
2	AS_NOUSER Unknown user	<p>The Enterprise User ID specified on the call is inactive or is not in the Census.</p> <p>Action: If you expect this user to be active in the Census, see "Managing the Census" in the <i>Core Driver Administration Guide</i> for additional information.</p>
3	AS_NOAGENT No core drivers are available for authentication	<p>No core drivers could be contacted to process the request.</p> <p>Action: This is generally caused by a configuration problem.</p> <ul style="list-style-type: none">◆ Ensure that the platform configuration file specifies the correct network addresses for the core drivers.◆ Ensure that the core driver is running on the specified servers and listening on the port specified in the platform configuration file.◆ Ensure that the Platform Services Process is running or that you have specified the <code>DIRECTTOAUTHENTICATION</code> statement in your platform configuration file.◆ Ensure that you have network connectivity to a core driver server.◆ Ensure that driver has been activated or that the evaluation period has not expired. <p>For more information, see the <i>Platform Services Planning Guide and Reference</i> and the <i>Core Driver Administration Guide</i>.</p>

Return Code	Symbol Text String	Explanation and Suggested Action
4	AS_BADCLIENT Local host is not authorized to query the core driver	<p>The network address used by the platform to contact a core driver did not match the network address listed in the Platform Configuration object in the ASAM System container.</p> <p>Action: For information about managing Platform objects with the Web interface, see the <i>Core Driver Administration Guide</i>.</p> <p>For an administrative password reset, this can indicate that the administrator user ID/password is not valid or that the administrator does not have rights to change the password.</p>
5	AS_REVOKED User is disabled/revoked	The specified Enterprise User ID corresponds to a User object that has been disabled.
6	AS_INTRUDER Intruder detection is active	The specified Enterprise User ID corresponds to a User object that has been locked because of intruder detection.
7	AS_INVALIDARGS Invalid arguments	<p>The arguments specified on the call are not valid.</p> <p>Action: Make certain that the arguments passed to the call are of the correct type and value. For example, an argument that specifies the name of an object cannot be blank or null, and an argument that specifies a buffer size to hold a result cannot be zero.</p>
8	AS_INVALIDOBJ Invalid object	<p>An object specified as an argument was not of the correct type or was not found.</p> <p>Action: Verify that the objects specified on arguments to the call are of the proper type. Handle the not-found condition as appropriate for your application.</p>
9	AS_INVALIDOBJLEN Invalid object length	<p>An object name specified as an argument was longer than the maximum allowable eDirectory™ object name.</p> <p>Action: Check object names that are specified as arguments to be sure that they do not exceed the maximum length for an eDirectory object name.</p>
10	AS_PASSDUPLICATE Password has been previously used	<p>The new password that was specified to the change password API function has been previously used for this User object and the User object is required to specify unique passwords.</p> <p>Action: Specify a password that has not been previously used.</p>
11	AS_PASSTOOSHORT Password does not meet password rules	<p>The new password that was specified to the change password API function is shorter than the minimum password length set for the User object.</p> <p>Action: Specify a password that meets the password rules for the User object.</p>
12	AS_TOOSMALL Buffer is too small	<p>The size specified for a buffer argument is too small to hold the result. The result is truncated.</p> <p>Action: Allocate a larger buffer, and issue the request again.</p>

Return Code	Symbol Text String	Explanation and Suggested Action
13	AS_ATTRNOTFOUND Attribute not found	The attribute specified was not found for the specified object. Action: Process this response accordingly, or specify the name of an attribute that exists for the specified object.
14	AS_WSOCKUP WINSOCK not initialized	Not used in the Identity Manager Fan-Out driver.
15	AS_WSOCKDOWN WINSOCK not terminated	Not used in the Identity Manager Fan-Out driver.
16	AS_NOAUTHENV No authentication environment established	The asce argument did not specify a valid environment item. C Action: Verify that a successful call to ASC_INIT() or ASC_INIT_EXT() has been made. Successful calls return a pointer to a valid environment item. Unsuccessful calls return NULL. Verify that the pointer to the environment item is correctly specified as an argument to this call. Java Action: Verify that a successful call to init() has been made.
17	AS_PRODUCTEXPIRED Ascauth client has expired	The expiration date for the platform has passed. Action: Install a current version of Platform Services.
18	AS_INCLUDED User matched an INCLUDE statement	The Enterprise User ID specified on a call to ASC_USER_INCLUDE_EXCLUDE() matched an AS.USER.INCLUDE statement in the platform configuration file.
19	AS_EXCLUDED User matched an EXCLUDE statement	The Enterprise User ID specified on a call to ASC_USER_INCLUDE_EXCLUDE() matched an AS.USER.EXCLUDE statement in the platform configuration file.
20	AS_NOMATCH User did not match any INCLUDE/EXCLUDE statement	The Enterprise User ID specified on a call to ASC_USER_INCLUDE_EXCLUDE() did not match any AS.USER.INCLUDE or AS.USER.EXCLUDE statement in the platform configuration file. The user is included because AS.USER.INCLUDE * is implicit if AS.USER.EXCLUDE * is not specified.
21	AS_NOLICENSE Client is not licensed to use the driver	Not used in the Identity Manager Fan-Out driver.
22	AS_INVALIDREQ API request is not valid or unsupported	The AS Client API call was not recognized by the core driver. Action: Ensure that the version of Platform Services and the version of the core driver are compatible.

Return Code	Symbol Text String	Explanation and Suggested Action
23	AS_KEYEXPIRED Client is using an expired DES key	The DES encryption key used by a non-SSL version of Platform Services has expired. Action: Update the KEY statement in the platform configuration file with the same encryption key that is specified for the Platform in the Platform object in the ASAM System container. For information about managing Platform objects with the Web interface, see the <i>Core Driver Administration Guide</i> .
24	AS_INSUFFICIENTRIGHTS Client is using an expired DES key	An administrative password reset was rejected. The administrative user does not exist, the password given for the administrative user is not valid, or the administrative user does not have rights to change the target user's password.