RSA ACE/Server v 4.1
External Authorization
API Guide

RSA Security Inc.
20 Crosby Drive
Bedford, MA 01730-1402
Main number: **781.687.7000**
Customer Services: **800.995.5095**
International calls: **Refer to Web site for specific countries.**
Email: **info@rsasecurity.com**
Web Site: **http://www.rsasecurity.com**

See our Web Site for regional Customer Service telephone and fax numbers.

Distribution
Limit distribution of this document to trusted personnel.

First printing: March 2000

Contents

Preface

This guide is intended only for programmers, security administrators and other trusted personnel. Do not make it available to the general user population.

This guide explains how to use a generic Application Program Interface (API) for External Authorization to create and set up customized External Authorization criteria. These customized criteria work in addition to, not in place of, RSA ACE/Server® authentication. When External Authorization is enabled, users attempting to gain access to your system must meet an extra set of criteria that you create based on the needs of your organization.

It is assumed that users of this guide are programmers who are experienced with C programming on Windows NT™ or UNIX® systems, and are familiar with RSA ACE/Server authentication procedures. Because this guide is for experienced programmers, it gives detailed procedures only where necessary. After reading this guide, you should be able to set up and test External Authorization on an RSA ACE/Server system with the customized External Authorization criteria. Once you have finished these setup tasks, an RSA ACE/Server administrator must enable External Authorization as explained in the *RSA ACE/Server v 4.1 Administration Manual* (**admin.pdf** in the ***ACEDOC*** directory).

Supported Platforms and Operating Systems

Refer to the *RSA ACE/Server v 4.1 for UNIX Installation Guide* or the *RSA ACE/Server v 4.1 for Windows NT Installation Guide* (**unix_install.pdf** or **nt_install.pdf** in the ***ACEDOC*** directory) for information on supported platforms and operating systems.

Getting Support and Service

If you are a customer with a maintenance agreement and you have registered to use SecurCare Online, visit our RSA Security SecurCare Online web site at **http://www.rsasecurity.com/securcare**

Note: Because it takes two weeks to process your registration request to use SecurCare Online, we recommend that you register when you first install your software from RSA Security.

SecurCare Online can answer many of your technical questions and help you solve problems without making a service call.

If you need technical assistance, call your service provider. If this is RSA Security, the Technical Support Center phone numbers are as follows:

Customer Support	Telephone
U.S. office	800.995.5095
International calls to U.S.	781.687.7700
U.K. office	44.118.936 2699
Toll–free within the U.K.	0800.072.5095
Singapore office	+65.7335400

For the telephone numbers of other field offices, please consult the RSA Security Web site: **http://www.rsasecurity.com**

For international requests or low-priority domestic requests, the Technical Support Center fax numbers are

Help Desk	781.687.7016
Technical Support	781.687.7017

Before You Call for Technical Support

Note: There is no provision for technical support during the warranty period unless a valid Software Service Contract is in force.

Make sure that you have direct access to the computer running the RSA ACE/Server Master Server software.

Please have the following information available when you call:

☐ Your RSA Security Customer/License ID. You can find this number on the license distribution medium or by running the Configuration Management application on the NT platform, or by typing ‘sdinfo’ on any UNIX platform.

_____

☐ RSA ACE/Server software version number.

_____

☐ The name and version of the operating system under which the problem occurs.

_____

☐ Whether your RSA ACE/Server system includes a Slave Server.

_____

☐ Whether you are running an information service (for example, DNS).

_____

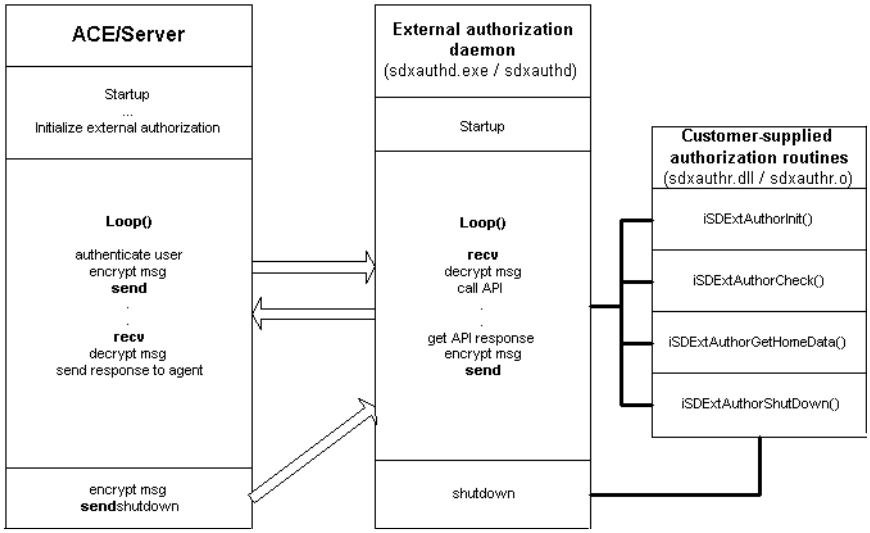
RSA ACE/Server v 4.1 External Authorization API

About External Authorization

External Authorization lets you use a generic Application Program Interface (API) for External Authorization to create customized criteria for authorizing users within your organization. This customized authorization is an addition to the RSA ACE/Server authentication, not a replacement for it. If the External Authorization option is enabled, users attempting access must meet this extra set of criteria before they are permitted access to your system. If the External Authorization option for authorization of remote login requests is also enabled, users are permitted access from a remote realm.

The RSA ACE/Server first applies the default authentication criteria to determine a user's identity. If the RSA ACE/Server authenticates the user's identity, a call is made to External Authorization. This call initiates a process that applies your customized criteria to determine whether or not a user is permitted access.

The following figure shows how External Authorization works in conjunction with RSA ACE/Server authentication to determine user identity and access.



If External Authorization fails, the user receives a timeout error message and the RSA ACE/Server denies subsequent authentication requests until either External Authorization is disabled or the RSA ACE/Server is restarted. Once the RSA ACE/Server is restarted, External Authorization resumes working in conjunction with the default RSA ACE/Server authentication to determine user identity and access.

External Authorization Components

RSA Security provides the **sdxauthr.dll** dynamic linked library for Windows NT-based systems and the **sdxauthr.o** static library for UNIX-based systems. These libraries are loaded on your Server when you install RSA ACE/Server v 4.1 software. They contain the following four API routines:

- **iSDExtAuthorInit()** – Performs any required initialization for External Authorization
- **SDExtAuthorCheck()** – Determines whether or not the user is authorized to log in to the client
- **iSDExtAuthorGetHomeData()** – Gets authorization data from the home realm as part of the enterprise authentication
- **iSDExtAuthorShutdown()** – Shuts down External Authorization

RSA Security also provides the following files that are on the RSA ACE/Server v 4.1 CD-ROM. Unless otherwise noted, these files are in the **/acesupp/sdxauthr** directory.

- **sdxauthd.exe** – For Windows NT-based systems, an External Authorization daemon to which the **sdxauthr.dll** dynamic linked library links. This file is in the **aceserv/nt_i386** directory that is loaded when you install the RSA ACE/Server v 4.1 software.
- **sdxauthd.a** – For UNIX-based systems, an External Authorization daemon to which the **sdxauthr.o** static library links
- **sdxatest.exe** – A Windows NT test program to exercise and debug the daemon and the customized routines that you create
- **sdxatest** – A UNIX test program to exercise and debug the daemon and the customized routines that you create
- **sdxatest.inp** – A sample script for the test program
- **sdxauthr_test.c** – A sample file that shows how to modify **sdxauthr.c** by adding authorization criteria
- Stubbed routines that you use to create your own External Authorization routines:

  sdxauthr.c – The template source file used to build the default **sdxauthr.dll** or **sdxauthr.o** shared library

  sdxauthr.h – The header file containing definitions to use when you create your customized External Authorization routines

  sdxauthr.mak – A file used to compile and link your C programs

RSA Security does *not* provide specific External Authorization code. Instead, you must create your own External Authorization routines based on the template source file. You substitute your customized routines for the stubbed routines supplied by RSA Security. Your customized routines must

- conform to this External Authorization API
- define a customized authorization scheme that integrates tightly with the RSA ACE/Server user-authentication process

A Task Overview

To set up External Authorization, you must perform the following tasks:

1. Review the information in "Before You Begin" and "Sample Calling Sequences."
2. Ask your RSA ACE/Server administrator to verify the authorization service port settings.
3. Create customized External Authorization routines that conform to this API.
4. Substitute your customized External Authorization routines for the stubbed routines supplied by RSA Security.
5. Optionally, create error messages for your customized External Authorization routines.
6. Compile your customized routines and link them, as a dynamic linked library for Windows NT or a static library for UNIX, to the External Authorization daemon.
7. Test your customized routines.

Once you have finished the above tasks, an RSA ACE/Server administrator must enable External Authorization as explained in the *RSA ACE/Server v 4.1 Administration Manual* (**admin.pdf** in the ***ACEDOC*** directory). The sections that follow provide additional information on these tasks.

Before You Begin

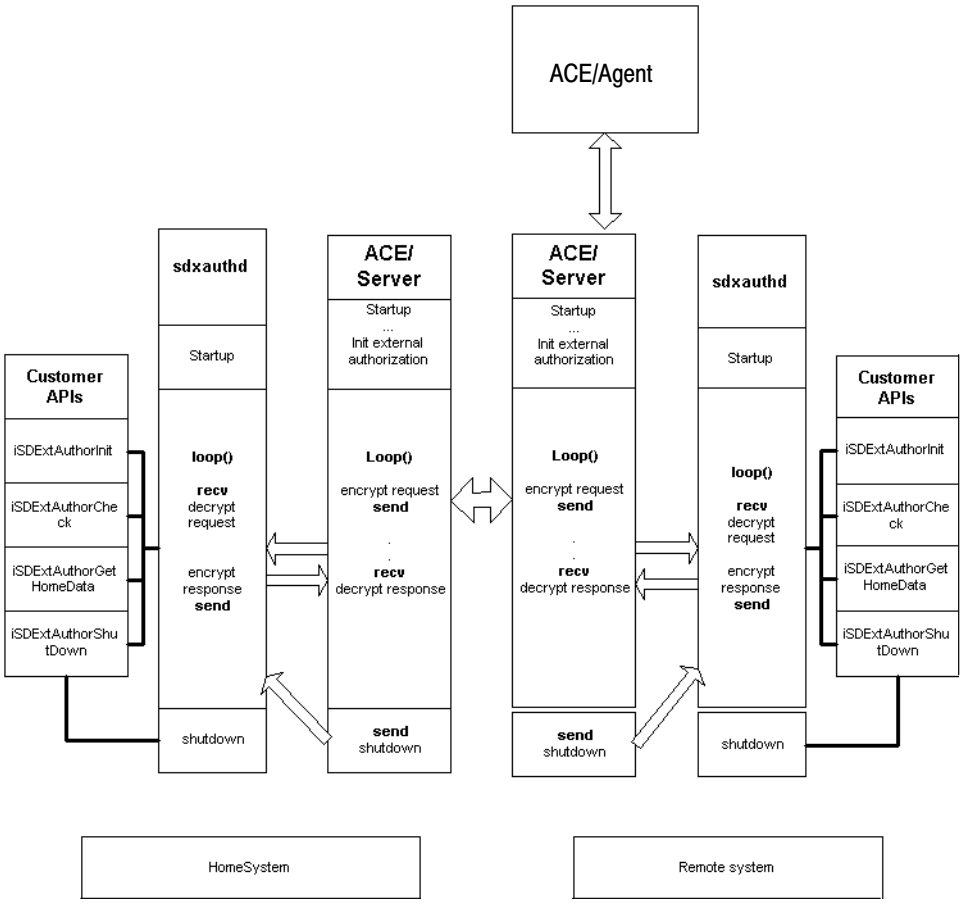
Review the information in this section before you begin setting up External Authorization. This review will help ensure that the setup goes smoothly and produces the results you want.

- You *must* include all four routines in the dynamic linked library (**sdxauthr.dll** for Windows NT-based systems) or static library (**sdxauthr.o** for UNIX-based systems). Do *not* exclude a stubbed routine if you do not plan to use it. Instead, stub the routine in the library to return 0 status. Maximum lengths for null-terminated strings include the byte for a trailing zero.
- Note the location of the source **sdxatest.exe**, **sdxatest**, and **sdxauthr.dll** files that shipped with the RSA ACE/Server v 4.1 software. You must substitute your customized versions of **sdxatest.exe** and **sdxauthr.dll** (for Windows NT) or **sdxatest** (for UNIX) files in the *same location* as the source files.
- You must provide your RSA ACE/Server system administrator with a list explaining any error messages that you create for your customized External Authorization routines. For messages with return values, you must explain the meaning you assigned to these values.
- After completing the tasks in this guide, ask your RSA ACE/Server system administrator to enable the External Authorization options that the administrator wants. Then restart your RSA ACE/Server so that your options take effect.

Sample Calling Sequences

Review the following sample sequences before you decide how to set up External Authorization. These sample sequences illustrate how an RSA ACE/Server with External Authorization enabled processes a user's request for access. The processing sequence depends on both your network configuration and which External Authorization options are enabled.

The sample sequences in this section apply to the following network configuration.



Single Server Configuration

The following sample sequence shows External Authorization of a user on an RSA ACE/Server.

External Authorization enabled
→ Receive an authentication request from the client. ↓
At the first request to authenticate a user, call **iSDExtAuthorInit()**. If **iSDExtAuthorInit()** fails, all subsequent user authentication attempts will fail. ↓
If RSA ACE/Server authentication is successful, call **iSDExtAuthorCheck()**. Log a message on return. ↓
Send an authentication response to the client. ↓
At server shutdown, call **iSDExtAuthorShutdown**.

Multi-Realm Configuration with External Remote Authorization Enabled on the Home Server

The following sample sequence shows External Authorization of a user from a remote realm. The enterprise authentication request goes to the user's home realm, which accepts the remote request.

Home Server External Authorization enabled External remote authorization enabled		**Remote Server** External Authorization enabled
		At the first request from the client to authenticate a user, call **iSDExtAuthorInit()**. ↓
At the first request to authenticate a user, call **iSDExtAuthorInit()**. ↓	←	Send an enterprise authentication request with the External Authorization flag set.
Process the enterprise authentication request. ↓		
Call **iExtAuthorGetHomeData()** to check authorization on the user's home realm and return data for authorization.	→	If authentication and home authorization are successful, call **iSDExtAuthorCheck()** with data from the home realm. ↓
		Send authentication response to the client. ↓
At Server shutdown, call **ExtAuthorShutdown()**.		At Server shutdown, call **ExtAuthorShutdown()**.

Multi-Realm Configuration with External Remote Authorization Disabled on the Home Server

The following sample sequence shows External Authorization of a user from a remote realm. The enterprise authentication request goes to the user's home realm, which has remote authorization disabled.

Home Server External Authorization enabled External remote authorization disabled		**Remote Server** External Authorization enabled
		→ At the first request to authenticate a user, call **iSDExtAuthorInit()**. Process the enterprise authentication request. ↓
At the first request to authenticate a user, call **iSDExtAuthorInit()**. ↓	←	Send an enterprise authentication request with the External Authorization flag set.
Process the enterprise authentication request. ↓		
Return disabled status indicating that external remote authorization is not enabled on the home realm. ↓	→	Send an authentication response to the client. ↓
At Server shutdown, call **ExtAuthorShutdown()**.		At Server shutdown, call **iSDExtAuthorShutdown()**.

Multi-Realm Configuration with External Authorization Disabled on the Remote Server

The following sample sequence shows authentication of a user from a remote realm that has disabled external remote authorization. The enterprise authentication request goes only to the user's home realm for authentication.

Home Server External Authorization enabled External remote authorization disabled		**Remote Server** External Authorization disabled
		→ Receive an authentication request from the client. ↓
At the first request to authenticate a user, call **iSDExtAuthorInit()**. Process the enterprise authentication request. ↓	←	Send an enterprise authentication request with *no* External Authorization flag set.
Process the enterprise authentication request only, *not* remote authorization. ↓	→	Send an authentication response to the client.
At Server shutdown, call **ExtAuthorShutdown()**		

Multi-Realm Configuration with External Authorization Disabled on the Home Server

The following sample sequence shows External Authorization of a user from a remote realm. The enterprise authentication request goes to the user's home realm, which has disabled External Authorization.

Home Server External Authorization disabled		**Remote Server** External Authorization enabled
		→ At the first request to authenticate a user, call **iSDExtAuthorInit()**. ↓
		Authentication request received from the client. ↓
Process the enterprise authentication request. ↓	←	Send an enterprise authentication request for authentication with an External Authorization flag set.
Return an error indicating that External Authorization is not enabled on the home realm.	→	Send an authentication failed response to the client. ↓
		At Server shutdown, call **iSDExtAuthorShutdown()**.

Setting Up External Authorization

This section explains the External Authorization setup tasks.

Substituting Customized Routines for the Stubbed Routines

Review the **sdauthor.mak** file and the source code in the stubbed routines **iSDExtAuthorInit()**, **DExtAuthorCheck()**, **iSDExtAuthorGet HomeData()**, and **iSDExtAuthorShutdown()**. Then create your customized External Authorization routines **sdxauthr.c**, **sdxauthr.h**, **sdxauthr.mak**, and substitute them for the stubbed routines.

Compiling and Linking Routines

For Windows NT-based systems, compile your customized routines **sdxauthr.c**, **sdxauthr.h**, **sdxauthr.mak**. Then, link your customized routines to the **sdxauth.exe** daemon as a dynamic linked library named **sdxauthr.dll**.

For UNIX-based systems, link your customized routines to the **sdxauthd.a** daemon as a static library named **sdxauthr.o**.

Creating Customized Status and Error Messages

Templates for creating messages for your customized External Authorization criteria are in the section for each stubbed routine in “API Routines” on page 15.

Note: For error messages, be sure to provide your RSA ACE/Server system administrator with a list of the return values and the meaning that you assigned to each of these values.

Testing Customized Routines

The test program lets you verify that your customized routines and the External Authorization daemon are working properly without your having to run your RSA ACE/Server.

To test your routines using the test program:

Note: The test program and other files listed in this procedure are on the RSA ACE/Server v 4.1 CD-ROM in the **acesupp/sdxauthr** directory.

---

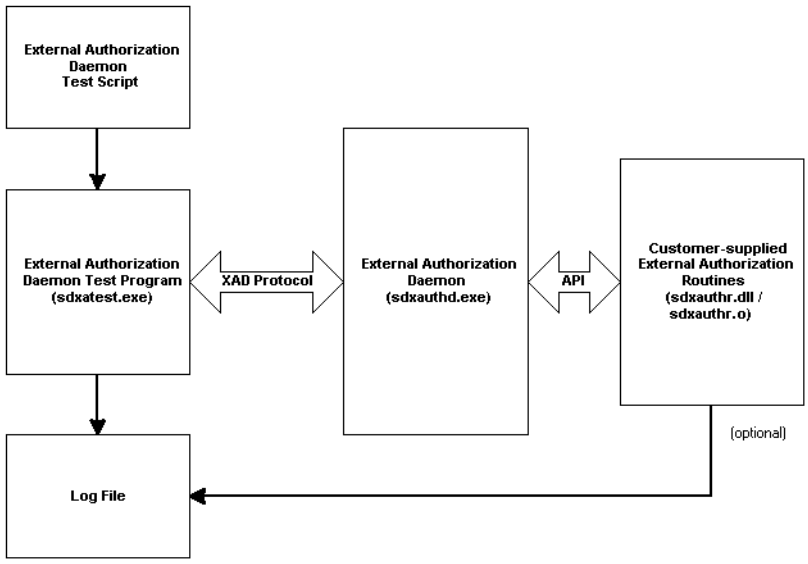
1. Review instructions on how to use the test program in the **sdxatest_readme_nt.txt** file (for Windows NT) or the **sdxatest_readme_unix.txt** (for UNIX) file.
2. Optionally, to use the log that runs with the test program, use the **sdxauthr_test.c** stubbed file to create a version of either the **sdxauthr.dll** dynamic linked library (for Windows NT) or the **sdxauthr.o** static library (for UNIX). This library simulates the module you create and the log file output.
3. Test your customized routines using the following files:

   The test program – **sdxatest.exe** for Windows NT or **sdxatest** for UNIX

   sdxatest.inp – A sample script for input to the test program

4. Start the daemon with the **test** command-line argument. This disables cross checks between the daemon and the RSA ACE/Server while you debug your customized routines.
5. Start the test program to generate protocol messages to the daemon and the External Authorization API.

The following figure illustrates what happens when you initiate an External Authorization request using the test program.



API Routines

This section describes the API routines that you use to create customized External Authorization. It also includes a sample audit log. The API routines

- set up and initialize the External Authorization API
- determine whether or not an authenticated user is permitted access
- if the **Enable Authorization of Remote Login Requests** option is enabled, pass additional authorization data from the home server to a remote server
- shut down the External Authorization API

You substitute the routines that you create for the stubbed routines in the library file. The library file (**sdxauthr.dll** for Windows NT or **sdxauthr.o** for UNIX) is linked at runtime to the External Authorization daemon spawned by the RSA ACE/Server.

Your External Authorization routines must execute as quickly as possible. If either **iSDExtAuthorCheck()** or **iSDExtAuthorGetHomeData()** takes a long time to process, the performance of the RSA ACE/Server severely degrades. If the API takes too long to process, the RSA ACE/Server times out the authentication request and returns a timeout error.

iSDExtAuthorInit()

Performs any required initialization in the External Authorization Module.

Synopsis

```
#include <sdxauthr.h>

int iSDExtAuthorInit(
   SD_U32             uExtAuthorFlags,  /* IN */
   P_SD_ERRSTRING_S   pErrorString            /* OUT */
);
```

Parameters

Parameter	Explanation
uExtAuthorFlags	Flags indicating the following External Authorization options for this RSA ACE/Server: SD_AUTHORIZE_ON – Currently not implemented SD_EXTAUTH_CHECK_ON – Enables External Authorization SD_EXTAUTH_HOMEDATA_ON – Enables external remote authorization; allows users to retrieve data from the home realm
pErrorString–>iStringType	0=ASCII
pErrorString–>cErrString	Pointer to location to receive a null-terminated user error string to be appended to syslog or event log.

Description

This routine lets the External Authorization Module initialize itself prior to the first authorization request. If this routine returns an error, **iSDExtAuthorCheck()**, **iExtAuthorGetHomeData()**, and **iSDExtAuthorShutdown()** are not called. All subsequent authentication requests will fail and return a timeout error until External Authorization is disabled or the RSA ACE/Server is restarted.

Return Values

0	Initialization succeeded.
nnnnn	Initialization error; the non-zero error value is stored in the audit log record.

Sample Customized Log Messages

Syslog or Event Log

External Authorization Initialized
External Authorization Initialization error, *errstring*

iSDExtAuthorCheck()

Authorizes a user to log in to the client.

Synopsis

```
#include <sdxauthr.h>

int iSDExtAuthorCheck (
   char  *           pszLoginName,     /* IN */
   char  *           pszClientName,    /* IN */
   unsigned int            InetAddr,         /* IN */
    P_SD_HOMEDATA_S  pHomeData,        /* IN */
   P_SD_ERRSTRING_S  pErrorString            /* OUT */
);
```

Parameters

Parameter	Explanation
pszLogin	Pointer to null-terminated string buffer containing the login name of the user authenticating on the RSA ACE/Server.
pszClientName	Pointer to null-terminated string buffer containing the name of the client machine the user is accessing.
InetAddr	32-bit unsigned integer containing the IP address of the client in network byte order. For example, address 12.1.68.4 is stored as 0x0C014404.
pHomeData	Pointer to the data structure retrieved from the home server during enterprise authentication. If pHomeData is null, the remote **iExtAuthorGetHomeData()** routine completed successfully but did not supply additional data.
pHomeData–>iLen	The number of bytes returned in cData
pHomeData–>cData	Data to be returned
pErrorString–>iStringType	0=ASCII
pErrorString–>cErrString	Pointer to a location to receive a null-terminated user error string to be appended to syslog or event log.

Description

If External Authorization is enabled on the RSA ACE/Server, this routine is called after a user successfully authenticates.

Your External Authorization routines must execute as quickly as possible. If either **iSDExtAuthorCheck ()** or **iSDExtAuthorGetHomeData()** takes a long time to process, the performance of the RSA ACE/Server severely degrades. If the API takes too long to process, the RSA ACE/Server times out the authentication request and returns a timeout error.

If **iSDExtAuthorInit()** or **iExtAuthorGetHomeData()** fails during an enterprise authentication, a user authorization failure is logged and this routine is not called.

Return Values

0	Authorization succeeded.
nnnnn	Authorization error; the non-zero error value is stored in the audit log record.

Sample Customized Log Messages

Audit Log

date time ExtAuthor Check *login user client server token*
date time ExtAuthor Check error *–nnnnn login user client server token*

iSDExtAuthorGetHomeData()

Gets local information to be returned as part of an enterprise authentication.

Synopsis

```
#include <sdxauthr.h>

int iSDExtAuthorGetHomeData (
   char  *           pszLogin,          /* IN */
   char  *           pszClientName,     /* IN */
   unsigned int      InetAddr,          /* IN */
   P_SD_HOMEDATA_S   pHomeData          /* OUT */
   P_SD_ERRSTRING_S  pErrorString            /* OUT */
);
```

Parameters

Parameter	Explanation
pszLogin	Pointer to a null-terminated string buffer containing the login name of the user authenticating on the RSA ACE/Server.
pszClientName	Pointer to a null terminated string buffer containing the name of the client machine the user is accessing.
InetAddr	32-bit unsigned integer containing the IP address of the client in network byte order. For example, address 12.1.68.4 is stored as 0x0C014404.
pHomeData	Upon successful authentication, a pointer to the location where data is copied to be returned to remote realm's **iSDExtAuthorCheck()** routine.
pHomeData–>iLen	Length of data returned in bytes, 0=no data returned.
pHomeData–>cData	Data to be returned
pErrorString–>iStringType	0=ASCII

Description

This routine is called if the **Enable Authorization of Remote Login Requests** option is enabled on the RSA ACE/Server, and the RSA ACE/Server then receives an enterprise authentication request with external remote authorization flagged.

Your External Authorization routines must execute as quickly as possible. If either **iSDExtAuthorCheck ()** or **iSDExtAuthorGetHomeData()** takes a long time to process, the performance of the RSA ACE/Server severely degrades. If the API takes too long to process, the RSA ACE/Server times out the authentication request and returns a timeout error.

If **iSDExtAuthorInit()** fails:

- An error is returned to the home realm.
- This routine is not called.
- A status of "authorization failed" is returned to the remote realm.

If external remote authorization is disabled in the home realm:

- A user message is logged.
- This routine is not called.
- A status of "authorization disabled" is returned to the remote realm.
- The remote realm skips further authentication.

Return Values

0	External remote authorization request succeeded.
nnnnn	External remote authorization error; the non-zero error value is stored in the audit log record.

Sample Customized Log Messages

Audit Log

date time ExtAuthor HomeData *login user client server token*
date time ExtAuthor Homedata error *–nnnnn login user client server token*

iSDExtAuthorShutdown()

Shuts down the External Authorization Module.

Synopsis

```
#include <sdxauthr.h>

int  iSDExtAuthorShutdown(
   P_SD_ERRSTRING_S  pErrorString            /* OUT */
   );
```

Parameters

Parameter	Explanation
pErrorString–>iStringType	0=ASCII
pErrorString–>cErrString	Pointer to location to receive a null-terminated user error string to be appended to syslog or event log.

Description

This routine is called just before the RSA ACE/Server is shut down or terminated. It allows the External Authorization Module to perform any required shutdown or cleanup procedures before it exits. If **iSDExtAuthorInit()** failed, a user authorization failure is logged and this routine is not called.

Disabling External Authorization while it is running on the RSA ACE/Server generates a call to this routine.

Return Values

0	Shutdown succeeded.
nnnnn	Shutdown error; the non-zero error value is stored in the audit log record.

Sample Customized Log Messages

Syslog or Event Log

External Authorization Shutdown
External Authorization Shutdown error, *errstring*

External Authorization Log Messages

External Authorization uses two forms of log messages:

- **For initialization and shutdown messages:** syslog (UNIX system log) or Windows NT event log messages

  Each API routine returns a null-terminated error string of up to 48 characters that is appended to the log message.

- **For initialization, check, home data, and shutdown messages:** Audit log messages (for RSA ACE/Servers on both Windows NT and UNIX platforms)

  If an External Authorization routine fails, it returns a non-zero error number. This error number is appended to the log message and stored in the audit log record.

The sections that follow list syslog or event log messages and audit log messages that may appear when External Authorization is enabled.

syslog or Event Log Messages

These messages appear in the UNIX system log (syslog) or the Windows NT event log.

General

External Authorization daemon failed
ACE/Server Fatal Error Execing sdxauthd (for UNIX-based systems)
ACE/Server Fatal Error Starting sdxauthd (for Windows NT-based systems)

iSDExtAuthorInit()+

External Authorization Initialized
External Authorization Initialization error, *errstring*

iSDExtAuthorShutdown()

External Authorization Shutdown
External Authorization Shutdown error, *errstring*

Audit Log Messages

These messages appear in the RSA ACE/Server audit log.

General

date time ExtAuthor daemon failed
Can't read ExtAuthor Packet
ACCESS DENIED, ExtAuth failed
XR ACCESS DENIED, ExtAuth failed

iSDExtAuthorCheck()

date time ExtAuthor Check *login user client server token*
date time ExtAuthor Check error *–nnnnn login user client server token*
date time ExtAuthor Homedata error *login user client server token*
date time ExtAuthor Homedata off *login user client server token*

iSDExtAuthorGetHomeData()

date time ExtAuthor HomeData *login user client server token*
date time ExtAuthor Homedata error *–nnnnn login user client server token*

Sample Audit Log

This log includes messages that appear on RSA ACE/Servers with External Authorization enabled.

```
Activity report      Date: 11/17/98 16:41

Date     Time   Current User/Client (Group)  Affected User
         Description                   (Site) Server

11/17/98 14:57L ExtAuthor Check              server_pc.abc.com
11/17/98 14:59U jones/client_pc              jones
11/17/98 14:59L ExtAuthor Check error-00012  server_pc.abc.com
11/17/98 15:02U fred/paris_pc                fred
11/17/98 15:02L ExtAuthor HomeData error-12345     server_pc.abc.com
11/17/98 15:07U wesson/client_pc             wesson
11/17/98 15:07L ExtAuthor Check              server_pc.abc.com
```