RSA ACE/Server 5.2 Administration Toolkit Reference Guide

Contact Information

See our Web sites for regional Customer Support telephone and fax numbers.

RSA Security Inc.
www.rsasecurity.com

RSA Security Ireland Limited
www.rsasecurity.ie

Trademarks

License agreement

Note on encryption technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Distribution

Limit distribution of this document to trusted personnel.

RSA Security Notice

First printing: October 2003

Contents

Preface 9
About this Guide 9
Getting Support and Service 9
Before You Call for Customer Support 9

RSA ACE/Server Administration Toolkit 11
Contents 11
Configuring the apidemon 13
Directory Requirements 16
C Applications 16
Tcl Scripts 16
Running Tcl Scripts on UNIX 17
Testing the Toolkit 17
Database Connections 18
Known Issues 18
Error Codes and Messages 19
Function Descriptions 19
Sd_AddAgentHost 20
Sd_AddAttributeToProfile 22
Sd_AddClient 24
Sd_AddClientExtension 26
Sd_AddGroup 27
Sd_AddGroupExtension 28
Sd_AddLoginToGroup 29
Sd_AddProfile 30
Sd_AddSecondaryNode 31
Sd_AddSite 32
Sd_AddSiteExtension 33
Sd_AddSysExtension 34
Sd_AddTokenExtension 35
Sd_AddUser 36
Sd_AddUserExtension 37
Sd_AdmContinueLogin 38
Sd_AdmLogin 39
Sd_ApiEnd 40
Sd_ApiInit 41
Sd_ApiInitSingle 43
Sd_ApiRev 45
Sd_ApplyPassword 46
Sd_AssignAnotherToken 48
Sd_AssignNewToken 49
Sd_AssignPassword 50

Sd_AssignProfile 51
Sd_AssignToken 52
Sd_ChangeAuthWith 53
Sd_ContinueLogin 54
Sd_DelClientExtension 55
Sd_DeleteAttributeFromProfile 56
Sd_DeleteClient 57
Sd_DeleteGroup 58
Sd_DeleteProfile 59
Sd_DeleteSecondaryNode 60
Sd_DeleteSite 61
Sd_DeleteToken 62
Sd_DeleteUser 63
Sd_DelGroupExtension 64
Sd_DelLoginFromClient 65
Sd_DelLoginFromGroup 66
Sd_DelSiteExtension 67
Sd_DelSysExtension 68
Sd_DelTokenExtension 69
Sd_DelUserExtension 70
Sd_DeployToken 71
Sd_DisableGroupOnClient 74
Sd_DisableToken 75
Sd_DumpHistory 76
Sd_DynamicSelect 79
Sd_EditAttributeInProfile 81
Sd_EmergencyAccessFixed 83
Sd_EmergencyAccessOff 84
Sd_EmergencyAccessOTP 85
Sd_EmergencyAccessOn 87
Sd_EnableGroupOnClient 89
Sd_EnableLoginOnClient 90
Sd_EnableToken 91
Sd_ExportTokens 92
Sd_GetAdminLevel 94
Sd_GetClientInfo 95
Sd_GetLastError 97
Sd_GetLDAPData 98
Sd_GetReplacementStatus 99
Sd_GetSerialByLogin 100
Sd_HexMD5 101
Sd_ImportToken 102
Sd_ImportTokenFile 103
Sd_ImportTokenFileExt 106

Sd_IsEmergencyAccess 108
Sd_LastErrorMode 109
Sd_ListAdministrators 110
Sd_ListAgentHostInfo 111
Sd_ListAssignedTokens 113
Sd_ListAttributes 114
Sd_ListAttributesInProfile 115
Sd_ListClientActivations 116
Sd_ListClientExtension 118
Sd_ListClients 119
Sd_ListClientsBySite 120
Sd_ListClientsForGroup 121
Sd_ListClientType 122
Sd_ListExtensionsForClient 123
Sd_ListExtensionsforGroup 124
Sd_ListExtensionsForSite 125
Sd_ListExtensionsForSys 126
Sd_ListExtensionsForToken 127
Sd_ListExtensionsForUser 128
Sd_ListGroupExtension 130
Sd_ListGroupMembership 131
Sd_ListGroups 133
Sd_ListGroupsByClient 134
Sd_ListGroupsBySite 135
Sd_ListHistory 136
Sd_ListJob 139
Sd_ListJobInfo 140
Sd_ListProfiles 142
Sd_ListReplicas 143
Sd_ListSecondaryNodes 144
Sd_ListSerialByLogin 145
Sd_ListSerialByName 147
Sd_ListSiteExtension 148
Sd_ListSites 149
Sd_ListSysExtension 150
Sd_ListTaskLists 151
Sd_ListTokensByField 152
Sd_ListTokenExtension 155
Sd_ListTokenInfo 156
Sd_ListTokenInfoExt 159
Sd_ListTokens 160
Sd_ListUnassignedTokens 161
Sd_ListUserByClient 162
Sd_ListUserByGroup 164

Sd_ListUserExtension 166
Sd_ListUserInfo 167
Sd_ListUserInfoExt 169
Sd_ListUsersByField 171
Sd_ListValuesForAttribute 175
Sd_Login 176
Sd_MakeUserRemote 177
Sd_MkSoftIDExt 178
Sd_MonitorHistory 181
Sd_NewPin 183
Sd_RemoveAdminPrivileges 184
Sd_ReplaceToken 185
Sd_RescindToken 186
Sd_ResetLastError 187
Sd_ResetToken 188
Sd_Resync 189
Sd_SetAgentHost 190
Sd_SetClient 193
Sd_SetClientExtension 195
Sd_SetClientSite 196
Sd_SetCreatePin 197
Sd_SetGroup 198
Sd_SetGroupExtension 199
Sd_SetGroupSite 200
Sd_SetLDAPData 201
Sd_SetLoggingOff 202
Sd_SetLoggingOn 203
Sd_SetPin 204
Sd_SetPinToNTC 206
Sd_SetProfileName 208
Sd_SetSite 209
Sd_SetSiteExtension 210
Sd_SetSymbols 211
Sd_SetSysExtension 213
Sd_SetTempUser 214
Sd_SetTokenExtension 215
Sd_SetUser 216
Sd_SetUserExtension 217
Sd_SortOrder 218
Sd_SwitchAdmin 220
Sd_TaskListDetails 221
Sd_Time 222
Sd_UnassignProfile 223
Sd_UnassignToken 224

RSA ACE/Server Administration Toolkit Executables 225
emergency 226
repltok 228
resync 229
setpin 230

Database Schema 231
Server Tables 232
CustClientExtension 233
CustGroupExtension 233
CustRealmExtension 233
CustSiteExtension 234
CustSystemExtension 234
CustTokenExtension 234
CustUserExtension 235
SDAdministrativeRole 235
SDAdministrator 236
SDAttribute 236
SDAttributeValue 237
SDClient 237
SDClientType 240
SDEnabledGroup 240
SDEnabledUser 241
SDGroup 241
SDGroupMember 242
SDOneTimePassword 242
SDProfile 243
SDRealm 243
SDRealmEnabledGroup 244
SDRealmEnabledUser 244
SDReplica 245
SDSchedJob 246
SDSecondaryNode 247
SDSite 247
SDSubTask 248
SDSystem 248
SDTask 251
SDTaskCategory 251
SDTaskList 252
SDTaskListItem 252
SDToken 253
SDUser 256
SDUserScope 257
SDValue 258

Log Tables 258
CustLogExtension 258
SDAALMTbl 259
SDLogEntry 262
SDLogMessage 263
SDLogReportFormat 263
SDSysLogCriteria 264
Error Messages and Codes 265

Preface

This guide describes the RSA ACE/Server Administration Toolkit function calls and the RSA ACE/Server databases. To use the toolkit, you must have C or Tcl programming experience.

About this Guide

This reference guide is meant only for security administrators and trusted personnel. Do not make it available to the general user population. The guide is divided into two sections.

The "RSA ACE/Server Administration Toolkit" section contains information about:

- Toolkit installation contents
- The **apidemon.ini** file
- Toolkit function calls

The "Database Schema" section is provided for reference purposes. It describes each table in the RSA ACE/Server databases (**sdserv** and **sdlog**), and indicates the fields to which your programs can write, as well as those fields that are read-only.

Getting Support and Service

RSA SecurCare Online	**https://knowledge.rsasecurity.com**
Customer Support Information	**www.rsasecurity.com/support**

Before You Call for Customer Support

Make sure you have direct access to the computer running the RSA ACE/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 Windows platforms, or by typing 'sdinfo' on any UNIX platform.
- ❑ RSA ACE/Server software version number.
- ❑ The make and model of the machine on which the problem occurs.
- ❑ The name and version of the operating system under which the problem occurs.

RSA ACE/Server Administration Toolkit

Contents

The RSA ACE/Server Administration Toolkit is installed as part of RSA ACE/Server. The following table lists the toolkit directories that are installed, and describes the contents of each directory.

Directory	Contents	Description
ace\utils\oldutil		PIN and token utilities
	repltok.exe (Windows), repltok (UNIX)	Replaces a token.
	resync.exe (Windows), resync (UNIX)	Resynchronizes a token.
	setpin.exe (Windows), setpin (UNIX)	Assigns a new PIN to a token.
	emergency.exe (Windows), emergency (UNIX)	Sets, clears, or queries information pertaining to lost token status and emergency access mode.
ace\utils\tcl\bin (Windows)		Windows binary files
	tcl-sd.exe	The Tcl application.
	tcl76.dll	The **.dll** for the Visual Tcl application.
	test.tcl	A Tcl test script.
	wish-sd.exe	The Visual Tcl application.
	sdtask.txt	A list of RSA ACE/Server administrative tasks.
	tk42.dll	The **.dll** for the Visual C application.
ace\utils\tcl\bin (UNIX)		UNIX binary files
	sdtask.txt	A list of RSA ACE/Server administrative tasks.
	tcl-sd	The Tcl application.
	test.tcl	A Tcl test script.
	wish-sd	The Visual Tcl application.
	admenv (UNIX)	Lists the environment variables that you need to set.

Directory	Contents	Description
ace\utils\toolkit		Library files, makefiles, header files, and utilities
	ace_api.lib (Windows) ace_api.a (UNIX)	The API library. All applications that use toolkit functions must link to this library. **Note:** A **.dll** replacement for **ace_api.lib** is available on request from RSA Security.
	admexamp.c	Contains sample C code that shows the use of toolkit functions.
	admexamp.mak	The toolkit makefile. This file runs on Windows, HP, AIX, and Solaris. You need to properly configure **admexamp.mak** for your platform. Refer to the introductory comments at the beginning of the file.
	api_errors.h	Contains strings that map to messages defined in **message.h**.
	api_msgs.h	Contains error messages specific to each toolkit function.
	apidemon.exe	Processes all toolkit function calls.
	apiuser.h	Supports functions that are defined in toolkit library. You must include this file in your application.
	dumpsamusers.exe (Windows only)	Reads user records from one or more SAM databases and writes them to a comma-separated text file. Each record contains the login, first name, and last name of a single user.
	loadsamusers.exe (Windows) loadsamusers (UNIX)	Reads and parses a text file generated by **dumpsamusers.exe**. For each user in the file, a record is created in the RSA ACE/Server database containing the login, first name, and last name for that user.
	message.h	Contains error messages that pertain to communication between the **apidemon** and the toolkit library.
	prompt_api.h	Contains coded authentication prompts and error messages.
	prompt_api.lib	A static library containing authentication prompts.
	prompt_api_dll.dll	The **.dll** to which your application must be linked in order to use the prompts in **prompt_api.lib**.
	prompt_api_dll.lib	An import library used by C applications for internal linking to the **prompt_api.dll** file.
	sdtask.txt	A list of RSA ACE/Server administrative tasks.

Configuring the apidemon

You can configure certain parameters in the **apidemon** by creating a text file called **apidemon.ini**. You must place this file in the same directory as your application. If you make subsequent changes to the file while the **apidemon** is running, you need to restart the **apidemon** for the changes to take effect.

Note: If you use the default **apidemon** configuration settings, you do not need to create the **apidemon.ini** file.

The **apidemon.ini** file must have one or more text lines that define each parameter. The definitions must be expressed in the following form:

```
[space]Key[space]Value[space]<New line>
```

[space]	Any combination of spaces, equals signs (=), and/or tab characters. This component is optional, however at least one character (space, tab, or equals sign) is required between **Key** and **Value**.
Key	The name that identifies a parameter to the **apidemon**. If the name is not recognized, the line is ignored.
Value	The value assigned to the parameter.
<New line>	A character sequence that begins a new line. No line can exceed 255 characters.

Parameters

The following table lists definable parameters in **apidemon.ini** and shows their acceptable values.

Important: TRUE and FALSE values are case-sensitive.

Parameter	Description
EXITDELAY	The number of milliseconds the **Sd_ApiEnd** function waits before disconnecting from the database when the last internal commit operation has been performed. Default: 3000 **Note:** 3000 is the minimum required in systems that frequently connect and disconnect from the database. If this is not a factor in your environment you can lower the default, but do not use a value below 300.

Parameter	Description
USESITE	Determines criteria for processing an argument that contains a group name. FALSE: The entire string is read as a group name. TRUE: The string is split into a group name and a site name. By default, the @ symbol is recognized as the separator between the group and site name. However, the RSA ACE/Server accepts group and site names containing this character. If your group or site names contain the @ symbol, use **Sd_SetSymbols** to define a different separator. **Note:** If you set the flag to FALSE, you can still create sites in your database and associate groups with those sites. The flag does not affect the functionality of sites. Default: FALSE
NOWAIT	Determines the actions taken by the **apidemon** in situations when two or more instances of the program simultaneously lock the same record. FALSE: The **apidemon** waits until the conflict is resolved. TRUE: The **apidemon** returns an error immediately. **Note:** If the value is FALSE, it is possible for the **apidemon** to halt indefinitely while it waits for a signal. You can eliminate this problem by using the TERMWAIT key to set an appropriate timeout limit. Default: TRUE
TERMWAIT	The number of seconds before the **apidemon** shuts down if it has halted indefinitely. This value must be positive and greater than 30 seconds, which is the minimum required to prevent a shutdown during an operation. Default: 300 **Note:** This setting does not affect **Sd_DumpHistory**, **Sd_ExportTokens**, or **Sd_ImportTokens**. These functions can take an unpredictable amount of time depending on the size of the log database.
REQAUTH	Determines if authentication is required before the function **Sd_ApiInit** is called to establish a database connection. FALSE: Authentication is not required. TRUE: Authentication is required. **Sd_Admlogin** and **Sd_AdmContinueLogin** must be called to initiate the authentication process *before* you call **Sd_ApiInit**. Without proper authentication, **Sd_ApiInit** fails. Default: FALSE **Note:** Do not set this flag to TRUE if you are running scripts and applications that connect to the database in single-user mode.

Parameter	Description
PROMPTMODE	Specifies the type of authentication prompts that are returned when any of the following login functions are called: • Sd_Login • Sd_ContinueLogin • Sd_AdmLogin • Sd_AdmContinueLogin For a list of defined prompts, refer to the header file **prompt_api.h** in the **\ace\utils\toolkit** directory. Acceptable values: "0" (Compatibility Mode): **Sd_Login** and **Sd_ContinueLogin** return hard coded messages. **Sd_AdmLogin** and **Sd_AdmContinueLogin** return the coded prompts defined in **prompt_api.h**. This setting is the default for Tcl. It is also maintained for backward compatibility. "1" (Abstract Mode): All four functions return the coded prompts defined in **promt_api.h**. "2" (Determined Mode): All four functions return the coded prompts you build using the **prompt_api** library. "3" (Mixed Mode): **Sd_AdmLogin** and **Sd_AdmContinueLogin** function as specified in "Determined Mode." **Sd_Login** and **Sd_ContinueLogin** function as specified in "Compatibility Mode."
SCOPE	Uses administrative scope to filter data that is returned with any of these functions: • Sd_ListTokenInfo • Sd_ListTokenInfoExt • Sd_ListGroupMembership • Sd_ListUsersbyField • Sd_ListTokensbyField FALSE: All data is returned regardless of administrative scope. TRUE: Data that is out of an administrator's scope is NOT returned. Default: FALSE

Directory Requirements

A user must have write access to the directory containing your application, **apidemon**, and **apidemon.ini** when he or she

- Starts the application.
- Uses the toolkit to write a new application.

In addition, the RSA ACE/Server uses this directory for temporary disk storage, which also requires write access. You can add an entry to the **startup.pf** file that specifies the directory. Type

```
-T dirname
```

where *dirname* is the path and name of the directory.

C Applications

The size of the message buffer passed to any API function must be at least **4097** bytes. This value is defined in **apiuser.h** as **MAX_RESULT_MSG_SIZE**. Be sure to use this constant in your code for future compatibility.

The **ace_api** library is NOT multithreaded on Windows. Administrative calls can be used only from a single thread inside a multithreaded application. Note the following:

- When linking to a single-threaded application, be sure to set the /**NODEFAULTLIB** flag to the value **libcmt.lib** as in **admexamp.mak**.
- When a multithreaded application is linked with the library, do not call the **printf** function in your application before you call **Sd_ApiInit**.

Tcl Scripts

Sd_ApiInit function and Tcl exec statement. When a Tcl script contains an executable statement before a call to **Sd_ApiInit**, a problem can occur that prevents the **apidemon** from being properly initialized. This causes the function call to fail. If you encounter this problem, edit the script so that it executes the auto_reset command *before* calling **Sd_ApiInit**. This clears internal structures and enables **Sd_ApiInit** to successfully start the daemon.

Catching return values. When a Tcl script calls any function, the function call should be evaluated by a statement that catches the return value (1 or 0). If the function returns 1 (failure) and this value is not caught, the execution of the Tcl script halts midway, and efforts to start it again may cause errors. For information on the catch command, refer to your Tcl manual.

Optional new parameters. Tcl function calls are completely backward-compatible. New parameters are optional and can be omitted.

Running Tcl Scripts on UNIX

Follow these steps before you run a Tcl script on a UNIX platform.

1. In the **ace/utils/tcl/bin** directory, run the **admenv** script. This script lists the path you should set as the value for each environment variable needed to run Tcl scripts
2. Verify that you are an administrator in the RSA ACE/Server database, and that UNIX permissions allow you to administer the application you are running. If the application is started with an account that does not have administrative status, pass authentication using **Sd_AdmLogin** or **Sd_AdmContinueLogin** to connect to the database.

   Note: If an authentication is successful, the **Sd_ApiInit** function verifies the administrative status of the user. If the user is not an administrator, **Sd_ApiInit** fails.

3. In the **utils/tcl/bin** directory, run **tcl-sd** to enter the Tcl environment. An error message appears if the Tcl environment variables are not properly set.
4. Create several test Agent Hosts in the RSA ACE/Server database.
5. In the Tcl environment, run the **test.tcl** script. This script lists all Agent Hosts in the RSA ACE/Server database. Verify that:
   - The VAR_ACE environment variable points to the directory containing the Server database files (**sdserv.db** and **sdlog.db**).
   - The user running the Tcl scripts is an administrator.
   - The RSA ACE/Server processes are running properly (you can verify this by running **sdadmin**.)
   - The X-Windows System is functioning properly on the workstation. In the Tcl/Tk GUI environment, run **wish-sd**. In the shell window, the **%** prompt appears, and a second X window is displayed.

Testing the Toolkit

You should set up a test environment to ensure that your compiled executable can run successfully.

To test the toolkit:

1. Stop the RSA ACE/Server database brokers.
2. Create a new database by running **sdnewdb**.
3. Add a Primary Server by running **sdrepmgmt**.
4. Add a realm administrator by running **sdcreadm**.
5. Start the RSA ACE/Server database brokers.
6. Run the **sdadmin** utility.
7. Import at least 20 tokens.
8. Add the Primary Server as an Agent Host, plus 2 or 3 additional Agent Hosts.

9. Assign a valid token or password to the administrator. Verify that he or she is enabled on an Agent Host.
10. To be sure that everything is set up properly, authenticate on the Server as the administrative user.

To see how the logging function **Sd_MonitorHistory** works, start **admexamp** in one window program using the monitor option on the command line. Without closing the monitor, start **admexamp** in another window with no command line parameters. As the program executes, you will see the log messages generated at each step appearing in the monitor window.

Database Connections

- **Sd_ApiInit** is the only function that starts the **apidemon** *and* establishes a connection with the database. **Sd_AdminLogin**, **Sd_Login**, **Sd_ApiRev**, **Sd_HexMD5**, **Sd_Time**, and **Sd_SetSymbols** only start the **apidemon**. They do not establish a connection with the database.
- All toolkit functions are available on a Primary Server, including those that make changes to the database. On a Replica Server, you can use only those functions that list or view information. When you call any function that produces a change to the database on a Replica Server, the error message "Function can't be called in read only mode." is returned.

Known Issues

- On UNIX platforms you may receive the error message "Sd_ApiInit error: *user* is not an ACE/Server administrator" even though the user is an administrator. Check that the **propath** environmental variable includes the path to the **ace/prog/proapi** directory.
- Tcl programmers should be aware of a potential error that does not generate a message. This error can affect any parameter in the argument list that is declared as an integer. The string provided in the Tcl function call is converted to an integer through the **atoi** function, which eliminates all characters that are not digits. If the string value includes no digits and cannot be converted to an integer, it is converted to zero. Ensure that your script provides appropriate values for all "int" parameters.
- Parameters with long strings are usually truncated to the length of the database field. The default limitation for all fields is 1024 bytes.
- For administrative ease, RSA Security recommends that you use a unique name for each extension data key. Do not use the same name twice for two different keys.
- Most functions that are made from inside an existing loop should execute properly. However if you plan to use looping functions, you should test all of them to ensure that they do not return any errors.

Error Codes and Messages

Function calls can return error codes contained in the header file **api_errors.h**.

If your applications were built with a version earlier than 5.1 and you are linking them to the API library contained in this version, you can suppress error codes so that your functions return only the following backward-compatible values.

- "0": Success
- "1": An error condition

To suppress error codes:

1. Create the following global variable. Type

   ```
   int iSuppressIfNonZero
   ```

2. Set the value to "1". Type

   ```
   iSuppressIfNonZero = 1
   ```

Function Descriptions

Toolkit functions are listed alphabetically by name. Function descriptions contain the following headings:

Function Prototype. Lists the types, names, and appropriate order of all parameters.

Tcl Function Call. Lists the names and appropriate order of Tcl parameters. If a parameter is shown in brackets, you can either omit it, or pass it as an empty string.

Description. Provides a detailed description of the function.

Parameters. Lists and defines all allowable parameters. All C versions of the API functions documented have the following two parameters that are included in the prototype, but not defined as parameters:

- **msgBuf**: A buffer that the user supplies. Return text is copied into this buffer as an empty terminated character string.
- **bufSize**: The number of bytes allocated for **msgBuf**. To ensure successful operation, this parameter must be defined as "MAX_RESULT_MSG_SIZE", or given the value of **4097**.

Return Values. Function calls return one of the following:

- Backward-compatible codes ("0" for success, "1" for failure). You can suppress error codes to maintain backward compatibility. For more information, see "Error Codes and Messages" on page 19.
- Error codes defined in **api_errors.h**

Sd_AddAgentHost

Function Prototype

Sd_AddAgentHost(char *agentHostName, char *agentHostAddr, char *siteName, int agentHostType, int encryptionType, int agentHostFlags, char *actingMaster, char *actingSlave, char *sharedSecret, char *msgBuf, int bufSize)

Tcl Function Call

Sd_AddAgentHost agentHostName agentHostAddr siteName agentHostType encryptionType agentHostFlags actingMaster actingSlave sharedSecret

Description

Creates a new Agent Host record in the database, assigns it to a specified site, and sets the Agent Host type, encryption type, and Agent Host flags.

Parameters

agentHostName	Name of the new Agent Host: either the full version of the name as in the RSA ACE/Server database (for example, **pc_agenthost.server.com**) or the short version (for example, **pc_agenthost**). Maximum 48 characters.
agentHostAddr	The IP address of the new Agent Host.
siteName	An existing site to which the Agent Host can be assigned. If you do not want to assign the Agent Host to an existing site, pass an empty string (“”).
agentHostType	0 UNIX agenthost 1 Communication server 2 Single-transaction communication server 3 Net OS server 4 NetSP server
encryption Type	0 SDI 1 DES

agentHostFlags	Flag 1 Create Node Secret? Flag 2 Open to All? Flag 3 Search Remote Realms? Flag 4 Require NameLock Note the following points about the flag settings: All four flags are defined as constants in the header file. Pipe notation can be used in C function calls with these constant names: CLIENT_SEND_NODE (= 1), CLIENT_OPEN_TOALL (= 2), CLIENT_REMOTE_SEARCH (= 4), CLIENT_NAME_LOCK (= 8). For example, to set Flags 2 and 3 to TRUE, use the following argument: “clientFlags = CLIENT_OPEN_TOALL | CLIENT_REMOTE_SEARCH”. Flag 1 (Sent Node Secret): This flag should be set to FALSE (0). It cannot be set to TRUE with this function, because the Agent Host has no prior existence in the database. To set any one of these flags, pass its defined value. To set a combination of the flags, add each of the defined values together, and pass the total value.
actingMaster	The name and IP address of the Acting Master.
actingSlave	The name and IP address of the ActingSlave.
sharedsecret	The encryption key used to establish a connection between the RADIUS Server and the Agent Host.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADDED_CLIENT

Sd_AddAttributeToProfile

Function Prototype

Sd_AddAttributeToProfile(char *profileName, int attributeNum, char *attributeValue, int valueFormat, char msgBuf, int bufSize);

Tcl Function Call

Sd_AddAttributeToProfile profileName attributeNum value [valueFormat]

Description

Adds an attribute to a profile and assigns either a predefined or formatted value. If there are pre-defined attribute values in the database, **valueFormat** is ignored (if supplied), and the **attributeValue** is used to retrieve the predefined value from the database. When **valueFormat** is used, the function interprets a value based upon its type. The following table lists allowed value types, **valueFormat** variables, and explains each value interpretation, and corresponding actions taken in the database.

Value Type	ValueFormat Variable	Interpretation/Action Taken
String	0	Assigns the value to an attribute as a string.
	1	First name value is dynamically assigned during RADIUS authentication
	2	Last name value is dynamically assigned during RADIUS authentication.
	3	Login value is dynamically assigned during RADIUS authentication.
	4	Shell value is dynamically assigned during RADIUS authentication.
	5	During user authentication, interprets the value as a prefix to the extension data key of the user record. **Note:** When specifying this variable, ensure that the attribute is defined in the dictionary so as to allow for multiple instances in the profile.
Integer	0	Value is applied decimal integer.
	1	Value is applied hexadecimal integer.
IP Address	0	Value is interpreted as an IP address in dotted format.
	1	Value is interpreted as an IP address in hexadecimal format.
	2	Value is interpreted as an IP address in decimal format.
Date	None	Interprets the string as a date when entered in the following format: MM/DD/YYYY HH:MM:SS If the time portion of the string is omitted, the date portion of the string is used to calculate the time.

Parameters

profilename	The name given to the profile.
attributeNum	The attribute number in the database.
attributeValue	The value assigned to the attribute.
valueFormat	A format type used to interpret the value. See the table under "Description."

Note: If **valueFormat** is omitted, "0" is used by default.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Internal sequence number of attribute value in the profile.

Logged Events

EDITED_PROFILE

Sd_AddClient

Function Prototype

int Sd_AddClient(char *clientName, char *siteName, int clientType,
int encryptionType, int clientFlags, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AddClient clientName siteName clientType encryptionType clientFlags

Description

Creates a new Agent Host record in the database, assigns the Agent Host to specified site, and sets the Agent type, encryption type, and Agent Host flags.

Note: This function is maintained for backward compatibility with prior versions of the RSA ACE/Server. For 5.2 administration, call the function Sd_AddAgentHost.

Parameters

clientName	Name of the new Agent Host: either the full version of the name as in the RSA ACE/Server database (for example, **pc_client.server.com**) or the short version (for example, **pc_client**). Maximum 48 characters.
siteName	An existing site to which the Agent Host can be assigned. If you do not want to assign the Agent Host to an existing site, pass an empty string (“”). Maximum 48 characters.
clientType	0: A UNIX Agent 1: A communication server 2: A single-transaction communications server 3: A Net OS server 4: A Net SP server
encryptionType	0: SDI 1: DES
clientFlags	Flag 1 Create Node Secret? Flag 2 Open to All? Flag 3 Search Remote Realms? Flag 4 Require NameLock

Note the following points about the flag settings:

All four flags are defined as constants in the header file. Pipe notation can be used in C function calls with these constant names: CLIENT_SEND_NODE (= 1), CLIENT_OPEN_TOALL (= 2), CLIENT_REMOTE_SEARCH (= 4), CLIENT_NAME_LOCK (= 8). For example, to set Flags 2 and 3 to TRUE, use the following argument: “clientFlags = CLIENT_OPEN_TOALL | CLIENT_REMOTE_SEARCH;”.

- Flag 1 (Sent Node Secret): This flag should be set to FALSE (0). It cannot be set to TRUE with this function, because the Client has no prior existence in the database.
- To set any one of these flags, pass its defined value. To set a combination of the flags, add each of the defined values together, and pass the total value.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADDED_CLIENT

Sd_AddClientExtension

Function Prototype

int Sd_AddClientExtension(char *key, char *data, char *clientName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AddClientExtension key data clientName

Description

Adds an extension field to an Agent Host record and updates data (maximum 80 characters). The function call must include a key (maximum 48 characters) that uniquely identifies this extension field.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.
clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_CLIENT

Sd_AddGroup

Function Prototype

int Sd_AddGroup(char *groupName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AddGroup groupName

Description

Creates a record in the database for a group specified by name. The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite. See "USESITE" on page 14 for more information on defining separators.

Parameters

groupName	The name of the group (optionally with a site name suffixed).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADDED_GROUP

Sd_AddGroupExtension

Function Prototype

int Sd_AddGroupExtension(char *key, char *data, char *groupName, *msgBuf, int bufSize);

Tcl Function Call

Sd_AddGroupExtension key data groupName

Description

Adds an extension field to a group record and inserts data (maximum 80 characters). The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite. The function call must include a key (maximum 48 characters) that uniquely identifies this extension field.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.
groupName	The name of the group (optionally with a site name suffixed).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_GROUP

Sd_AddLoginToGroup

Function Prototype

int Sd_AddLoginToGroup(char *groupLogin, char *groupName, char *groupShell, char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AddLoginToGroup groupLogin groupName groupShell tokenSerialOrLogin

Description

Makes a user, specified by token serial number or by login (with prefix), a member of an existing group in the database. The login, which must be unique within the group, can be the user's default login or a special login used for the group. To specify the user's default login and shell, pass **groupLogin** and **groupShell** as empty strings (""). The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite.

Parameters

groupLogin	Group login. May be user's default login or a special login to be used for the group. Maximum 48 characters.
groupName	Name of the group of which to add the user (optionally including a suffixed site name.)
groupShell	Group shell (256 characters maximum). May be user's default shell or a different shell to be used for the group.
tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADD_MBR_TO_GROUP

Sd_AddProfile

Function Prototype

intSd_AddProfile (char *profileName,char *msgBuf, int bufSize)

Tcl Function Call

Sd_AddProfile profileName

Description

Creates a new profile in the database specified by a profile name.

Parameters

profileName	The name of the profile. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADDED_PROFILE

Sd_AddSecondaryNode

Function Prototype

intSd_AddSecondaryNode(char *agentHostName, char *secondaryNodeName, char* secondaryNodeAddress, char* msgBuf int bufSize)

Tcl Function Call

Sd_AddSecondaryNode agentHostMane secondaryNodeName secondaryNodeAddress

Description

Creates a new secondary node record in the database, for the specifed Agent Host.

Parameters

agentHostName	The full name of the Agent Host as recorded in RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.
SecondaryNodeName	The secondary node name to be resolved by the secondary node address, pass an empty string (“”).
SecondaryNodeAddress	The secondary node address to be added to the Agent Host. If you want the secondary node name to be resolved by the secondary node name, pass an empty string (“”). **Note:** Both the secondaryNodeName and secondaryNodeAddress cannot be empty strings.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADD_CLIENT_SEC_NODE

Sd_AddSite

Function Prototype

int Sd_AddSite(char *siteName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AddSite siteName

Description

Creates a new record for a site specified by name and adds it to the database.

Parameters

siteName	Name of the site. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADDED_SITE

Sd_AddSiteExtension

Function Prototype

int Sd_ AddSiteExtension(char *key, char *data, char *siteName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AddSiteExtension key data siteName

Description

Adds an extension field to a site record and inserts data (maximum 80 characters). The function call must include a key (maximum 48 characters) that uniquely identifies this extension field.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.
siteName	Name of the site. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_SITE

Sd_AddSysExtension

Function Prototype

int Sd_AddSysExtension(char *key, char *data, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AddSysExtension key data

Description

Adds an extension field to the system record and inserts data. The function call must include a key that uniquely identifies this extension field.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ENTER_EDIT_SYSTEM_TEXT

Sd_AddTokenExtension

Function Prototype

int Sd_ AddTokenExtension(char *key, char *data, char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AddTokenExtension key data tokenSerialNumber

Description

Adds an extension field to a token record and inserts data. The function call must include a key that uniquely identifies this extension field.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field Maximum 80 characters.
tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_TOKEN

Sd_AddUser

Function Prototype

int Sd_ AddUser(char *lastName, char *firstName, char *defaultLogin, char *defaultShell, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AddUser lastName firstName defaultLogin defaultShell

Description

Creates a new user record including name, login, and shell. All other fields in the record are set to the default values.

Parameters

lastName	User's last name. Maximum 24 characters.
firstName	User's first name. Maximum 24 characters.
defaultLogin	User's default login. Maximum 48 characters.
defaultShell	User's default shell. Maximum 256 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADDED_USER

Sd_AddUserExtension

Function Prototype

int Sd_AddUserExtension(char *key, char *data, char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AddUserExtension key data tokenSerialOrLogin

Description

Adds a user extension record, specifying the user by token serial number or by login (with prefix), and inserts data. The function call must include a key that uniquely identifies this extension field.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.
tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_USER

Sd_AdmContinueLogin

Function Prototype

int Sd_AdmContinueLogin (char *response, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AdmContinueLogin response

Description

The **Sd_AdmLogin** function is used to initiate the authentication procedure for a specified administrative user, whose default login is valid in the RSA ACE/Server database. **Sd_AdmContinueLogin**, which can be called as many times as necessary, is used to supply the responses requested by the system, such as a tokencode, next tokencode, or PIN.

Parameters

response	Any response requested by the RSA ACE/Server during the authentication procedure.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists, and REPEAT (value 2) if another call is required because it could not be determined whether the user authenticated successfully.

Return Text

If successful, “Enter Passcode” is displayed.

Logged Events

Any message that can be logged during the authentication process. Logging is done by administrative server, not by the API.

Sd_AdmLogin

Function Prototype

int Sd_AdmLogin (char *login, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AdmLogin

Description

Starts the authentication procedure for a specified adminstrator. **Sd_AdmContinueLogin**, which can be called as many times as necessary, is used to supply the responses requested by the system, such as a tokencode, next tokencode, or PIN. In instances where **Sd_AdmLogin** and **Sd_AdmContinueLogin** are used, and **Sd_ApiInit** is called immediately afterwards, the toolkit is designed to initialize the apidaemon with the user ID of authenticated user. For **Sd_AdmLogin** and **Sd_AdmContinueLogin** to be complete successfully, the user must have administrative status in the RSA ACE/Server database.

Parameters

login	Login of the administrative user to authenticate.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

If successful, “Enter Passcode” is displayed.

Logged Events

None.

Sd_ApiEnd

Function Prototype

int Sd_ApiEnd(char *msgBuf, int bufSize);

Tcl Function Call

Sd_ApiEnd

Description

Shuts down the **apidemon** and logs out of the RSA ACE/Server database if a connection currently exists. (The daemon may be running without a database connection if the program or script has called a function such as **Sd_Time** or **Sd_Login**, but has not yet called **Sd_ApiInit**.) **Sd_ApiEnd** is the only proper way to terminate the database connection, which can be re-established by calling **Sd_ApiInit**.

Every program or script should call **Sd_ApiEnd** just before ending. The **apidemon** shuts down automatically as soon as it detects that the calling application has terminated (within two minutes), but you are not advised to rely on this feature. Terminating the database connection and the daemon with **Sd_ApiEnd** is the best way to avoid errors.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

SDADMIN_TERMINATED (only if **Sd_ApiInit** has been called previously)

Sd_ApiInit

Function Prototype

int Sd_ApiInit(char *serverDb, char *logDb, char *mode, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ApiInit serverDB logDB mode

Description

Initializes and connects to the Server databases in multi-user mode. Checks for license compliance and returns a warning message or an error message under the following conditions:

- The evaluation license has expired. In this case, an attempt to connect to the database fails.
- The number of users or Replica Servers has exceeded the number allowed by temporary upgrade license. In this case, a connection can be established with the database, however a warning message is displayed indicating that the RSA ACE/Server is being used in license violation.
- The number of users in the database exceeds the number allowed. In this case, a connection to the database is established, however the following functions are disabled due to the fact that they can directly or indirectly add to the number of users in the database: **Sd_AddUser**, **Sd_AssignPassword**, **Sd_AssignToken**, **Sd_ImportTokenFile**, and **Sd_ImportTokenFileExt**.
- The number of Replica Servers in the database exceeds the number allowed.

This function must be called before any other function that performs database operations. To connect successfully, the user must be an RSA ACE/Server administrator. The database commit mode is always “automatic,” meaning that each API function call is a separate database transaction.

Note: If the REQAUTH definable key is set to TRUE, additional authentication information is required. See the description of REQAUTH under “Parameters” on page 14.

Parameters

serverDB	Path and filename of the Server database file. May be passed as an empty string if VAR_ACE and USR_ACE are set correctly.
logDB	Path and filename of the Log database file. May be passed as an empty string if VAR_ACE and USR_ACE are set correctly.
mode	Any nonzero value for automatic commit mode. Manual commit mode (value 0) is not supported in the current version and causes an error.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Connected User: UserName on Host: HostName

Logged Events

SDADMIN_UNAUTHORIZED or SDADMIN_STARTED

Sd_ApiInitSingle

Function Prototype

int Sd_ ApiInitSingle(char *serverDb, char *logDb, char *mode, char* msgBuf, int bufSize);

Tcl Function Call

Sd_ApiInitSingle serverDb logDb mode

Description

Initializes and connects to the Server databases in single-user mode. Checks for license compliance and returns a warning message or an error message under the following conditions:

- The evaluation license has expired. In this case, an attempt to connect to the database fails.
- The number of users or Replica Servers has exceeded the number allowed by temporary upgrade license. In this case, a connection can be established with the database, however a warning message is displayed indicating that the RSA ACE/Server is being used in license violation.
- The number of users in the database exceeds the number allowed. In this case, a connection to the database is established, however the following functions are disabled due to the fact that they can directly or indirectly add to the number of users in the database: **Sd_AddUser**, **Sd_AssignPassword**, **Sd_AssignToken**, **Sd_ImportTokenFile**, and **Sd_ImportTokenFileExt**.
- The number of Replica Servers in the database exceeds the number allowed.

Either this function or **Sd_ApiInit** must be called before any function that performs database operations. To connect successfully, the user must be an RSA ACE/Server administrator. The database commit mode is always “automatic,” meaning that each API function call is a separate database transaction. This function will fail if the REQAUTH defineable key is set to TRUE.

Parameters

serverDB	Filename of the Server database file. May be passed as an empty string if VAR_ACE and USR_ACE are set correctly.
logDB	Filename of the Log database file. May be passed as an empty string if VAR_ACE and USR_ACE are set correctly.
mode	Any nonzero value for automatic commit mode. Manual commit mode (value 0) is not supported in the current version and causes an error.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Connected User: UserName on Host: HostName.

Logged Events

SDADMIN_UNAUTHORIZED or SDADMIN_STARTED

Sd_ApiRev

Function Prototype

int Sd_ApiRev(char *msgBuf, int bufSize);

Tcl Function Call

Sd_ApiRev

Description

Returns the revision number and build date of the API.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Date and time of build, internal build revision number.

Logged Events

None.

Sd_ApplyPassword

Function Prototype

int Sd_ApplyPassword(char *tokenSerialOrLogin, char *passWord, int days, int hours, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ApplyPassword tokenSerialOrLogin passWord days hours

Description

Assigns the password specified in the function call to an existing user identified by login or token serial number. The password is valid only for the number of days and hours specified in the call. Its expiration can be determined by adding the specified days and hours to the current date and time. (This is inconsistent with the way password expirations are specified in RSA ACE/Server administration, but it is left unchanged for backward compatibility).

Note: Only one password is allowed per user.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

passWord	Temporary password to assign to the user. The password must conform to system-defined standards for number of characters and whether the characters are numeric or alphanumeric.
days	Number of days password is valid.
hours	Number of hours that the password is valid (this time is added to number of days, and has a maximum value of 23 hours).

Examples: 0 days, 6 hours; 1 days, 12 hours (= 36 hours); 7 days, 0 hours (= 1 week)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The serial number of the token that contains the password. A user password is implemented as a special token of which the password is the PIN.

Logged Events

CREATE_PW_TOKEN, EDITED_USER

Sd_AssignAnotherToken

Function Prototype

int Sd_AssignAnotherToken(char *TokenSerialOrLogin,
char *newTokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AssignAnotherToken TokenSerialOrLogin newTokenSerialNumber

Description

Assigns a new token to a user identified by token serial number or login. (Users can be assigned a maximum of three tokens.) User information is not changed.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

newTokenSerialNumber	Serial number of new token to assign to user.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ASSIGN_TOKEN

Sd_AssignNewToken

Function Prototype

int Sd_AssignNewToken(char *oldTokenSerialNumber,
char *newTokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AssignNewToken oldTokenSerialNumber newTokenSerialNumber

Description

Assigns a new token to a user. The user's information is preserved, and the PIN that was assigned to the old token is now assigned to the new token.

Note: This function, reflecting former RSA ACE/Server usage, does not organize a replacement pair. For a replacement function that operates according to current standards, use **Sd_ReplaceToken**.

Parameters

oldTokenSerialNumber	Serial number of user's current token. Must be 12 characters. Insert leading zeros as needed, for example, 000000123456.
newTokenSerialNumber	Serial number of new token to assign to user.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

UNASSIGN_TOKEN, ASSIGN_TOKEN

Sd_AssignPassword

Function Prototype

int Sd_AssignPassword(char *lastName, char *firstName, char *defaultLogin, char *defaultShell, char *passWord, int days, int hours, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AssignPassword lastName firstName defaultLogin defaultShell passWord days hours

Description

Adds a new user to the RSA ACE/Server database, assigning a password valid for the number of days and hours specified in the function call. (The expiration of the password can be determined by adding the specified days and hours to the current date and time).

Parameters

lastName	Last name of user. Maximum 24 characters.
firstName	First name of user. Maximum 24 characters.
defaultLogin	Default login of user. Maximum 48 characters.
defaultShell	Default shell of user. Maximum 256 characters.
passWord	User's initial password. It must conform to system-defined standards for number of characters and whether the characters are numeric or alphanumeric.
days	Number of days password is valid.
hours	Number of hours that the password is valid (this time is added to number of days, and has a maximum value of 23 hours).

Examples: 0 days, 6 hours; 1 days, 12 hours (= 36 hours); 7 days, 0 hours (= 1 week)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The serial number of the token that contains the password. A user password is implemented as a special token of which the password is the PIN.

Logged Events

CREATE_PW_TOKEN and ADDED_USER

Sd_AssignProfile

Function Prototype

intSd_AssignProfile(char *tokenSerialOrLogin, char *profileName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AssignProfile tokenSerialOrLogin profileName

Description

Assigns a profile specified by **profileName** to a user specified by **tokenSerialOrLogin**.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.
profileName	The name of the profile.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADDED_PROFILE_TO_USER

Sd_AssignToken

Function Prototype

int Sd_AssignToken(char *lastName, char *firstName, char *defaultLogin, char *defaultShell, char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_AssignToken lastName firstName defaultLogin defaultShell tokenSerialNumber

Description

Adds a user to the RSA ACE/Server database and assigns the token specified by tokenSerialNumber. The token is enabled, the PIN is cleared, and both BadTokenCodes and BadPINs are set to zero. Whether the user is allowed to create a new PIN, required to create a new PIN, or issued a system-generated PIN depends on the policy defined for the system.

Parameters

lastName	Last name of user. Maximum 24 characters.
firstName	First name of user. Maximum 24 characters.
defaultLogin	Default login of user. Maximum 48 characters.
defaultShell	Default shell of user. Maximum 256 characters.
tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADDED_USER, ASSIGN_TOKEN, and ENABLE_TOKEN

Sd_ChangeAuthWith

Function Prototype

int Sd_ChangeAuthWith(int i tknAuthWith, char*chTokenSerial);

Tcl Function Call

Sd_ChangeAuthWith i tknAuthWith chTokenSerial

Description

Changes the type of PIN for a token.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

CHANGED_PIN_TYPE

Sd_ContinueLogin

Function Prototype

int Sd_ContinueLogin(char *response, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ContinueLogin response

Description

The **Sd_Login** function is used to initiate the authentication procedure for a specified user. **Sd_ContinueLogin**, which can be called as many times as necessary, is used to supply the responses requested by the system, such as a tokencode, next tokencode, or PIN.

Parameters

response	Any response requested by the system during the authentication procedure.

Return Values

Values returned are OK (value 0) upon successful completion of function, ERROR (value 1) if an error condition exists, and REPEAT (value 2) if another call is required because it could not be determined whether the user authenticated successfully.

Return Text

System response to the value passed in the function call.

Logged Events

Any message that can be logged during the authentication process. Logging is done by the RSA ACE/Server, not by the API.

Sd_DelClientExtension

Function Prototype

int Sd_DelClientExtension(char *key, char *clientName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DelClientExtension key clientName

Description

Deletes an extension field with its data from an Agent Host record in the database. The field is identified by the key specified in the function call.

Note: This function assumes that key names are unique. If more than one Agent Host extension key has the same name, a function call deletes only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Key that identifies the field to be deleted. To delete the first extension field encountered, set the argument value to "*" (used in repeated calls to delete all extension fields for the specified Agent Host).
clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_CLIENT

Sd_DeleteAttributeFromProfile

Function Prototype

intSd_DeleteAttributeFromProfile(char * profileName, int attributeNum, int sequenceNum, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DeleteAttributeFromProfile profileName attributeNum [sequenceNum]

Description

This function deletes the attribute value identified by attributeNum from the profile specified by profileName.

If the sequenceNum is 0 or is omitted, the function deletes all multiply-defined attribute values for a specified attribute in the profile.

If the sequenceNum is not 0, the function deletes a specific instance of the attribute in the profile.

Parameters

profileName	Name of the profile.
attributeNum	Number value which defines an attribute.
sequenceNum	Distinguishes a multiply-defined attribute.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_PROFILE

Sd_DeleteClient

Function Prototype

int Sd_DeleteClient(char *clientName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DeleteClient clientName

Description

Deletes a specified Agent Host record from the database. For the deletion to succeed, all users or groups must be disabled on this Agent Host, and all dependent records such as extensions must be deleted, before the function is called.

Parameters

clientName	Name of the Agent Host to delete: must be the full version of the name stored in the Server database, for example, **pc_client.server.com**.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DELETED_CLIENT

Sd_DeleteGroup

Function Prototype

int Sd_DeleteGroup(char *groupName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DeleteGroup groupName

Description

Deletes a specified group record from the database. For the deletion to succeed, all users must be removed from the group, and all dependent records such as extensions must be deleted, before this function is called. A group that has a group administrator cannot be deleted until the administrator is removed. (This can be done through the **sdadmin** utility.)

The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite. See "USESITE" on page 14 for more information on defining separators.

Parameters

groupName	Name of the group to delete (optionally including a suffixed site name).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DELETED_GROUP

Sd_DeleteProfile

Function Prototype

intSd_DeleteProfile (char *profileName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DeleteProfile profileName

Description

Deletes a specified profile record from the database. For the deletion to succeed, the profile should be unassigned from all users.

Parameters

profile name	The name of the profile. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DELETED_PROFILE

Sd_DeleteSecondaryNode

Function Prototype

intSd_DeleteSecondaryNode(char *agentHostName, char *secondaryNodeName, char* secondaryNodeAddress, char* msgBuf int bufSize)

Tcl Function Call

Sd_DeleteSecondaryNode agentHostName secondaryNodeName secondaryNodeAddress

Description

Deletes the specified secondary node record from the database. For the deletion to succeed, both the Agent Host and the secondary node must be valid. The secondary node must be associated with the specified Agent Host.

Parameters

agentHostName	Name of the Agent Host whose secondary node to delete: must be the full version of the name stored in the RSA ACE/Server database (for example, **pc_client.server.com**).
secondaryNodeName	Name of the secondary node to be deleted: must be the full version of the name stored in the RSA ACE/Server database (for example, **pc_secondarynode.server.com**).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DEL_CLIENT_SEC_NODE

Sd_DeleteSite

Function Prototype

int Sd_DeleteSite(char *siteName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DeleteSite siteName

Description

Deletes a specified site record from the database. For the deletion to succeed, all associated groups and Agent Hosts must be removed, or unassigned from the site, and all dependent records such as extensions must be deleted, before the function is called. A site that has a site administrator cannot be deleted.

Parameters

siteName	Name of the site to delete.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DELETED_SITE

Sd_DeleteToken

Function Prototype

int Sd_DeleteToken(char * tokenSerialNumber char *msgBuf, int bufSize);

Tcl Function Call

Sd_DeleteToken tokenSerialNumber

Description

Deletes from the database the record of an unassigned token identified by the specified serial number. For the deletion to succeed, any extension records dependent on the token record must first be deleted.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

TOKEN_DELETED

Sd_DeleteUser

Function Prototype

int Sd_DeleteUser(char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DeleteUser tokenSerialOrLogin

Description

Deletes a user record from the database. The user is specified in the function call by the serial number of an associated token or by login (with prefix). This function unassigns the user's token or tokens and removes references to the user from related tables.

Note: Administrators cannot be deleted.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DELETED_USER

Sd_DelGroupExtension

Function Prototype

int Sd_DelGroupExtension(char *key, char *groupName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DelGroupExtension key groupName

Description

Deletes an extension field with its data from a group record in the database. The field is identified by the key specified in the function call. The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite. See "USESITE" on page 14 for more information on defining separators.

Note: This function assumes that key names are unique. If more than one group extension key has the same name, a function call deletes only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Key that identifies the field to be deleted. To delete the first extension field encountered, set the argument value to "*" (used in repeated calls to delete all extension fields for the group).
groupName	The name of the group (optionally with a site name suffixed).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_GROUP

Sd_DelLoginFromClient

Function Prototype

int Sd_DelLoginFromClient(char *clientLogin, char *clientName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DelLoginFromClient clientLogin clientName

Description

Deletes the login with which a user is enabled on a specified Agent Host, preventing the user from logging in on that Agent Host.

Parameters

clientLogin	Login with which the user is enabled on the Agent Host
clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

REM_USER_FROM_CLIENT

Sd_DelLoginFromGroup

Function Prototype

int Sd_DelLoginFromGroup(char *groupLogin, char *groupName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DelLoginFromGroup groupLogin groupName

Description

Deletes the login with which a user is enabled in a specified group, preventing the user from logging in as a member of that group. The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite. See "USESITE" on page 14 for more information on defining separators.

Parameters

groupLogin	Login with which the user is enabled in the group.
groupName	The name of the group (optionally with a site name suffixed).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DEL_MBR_FROM_GROUP

Sd_DelSiteExtension

Function Prototype

int Sd_DelSiteExtension(char *key, char *siteName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DelSiteExtension key siteName

Description

Deletes an extension field with its data from a site record in the database. The field is identified by the key specified in the function call.

Note: This function assumes that key names are unique. If more than one site extension key has the same name, a function call deletes only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Key that identifies the field to be deleted. To delete the first extension field encountered, set the argument value to "*" (used in repeated calls to delete all extension fields for the site).
siteName	Name of the site.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_SITE

Sd_DelSysExtension

Function Prototype

int Sd_DelSysExtension(char *key, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DelSysExtension key

Description

Deletes an extension field with its data from the system record in the database. The field is identified by the key specified in the function call.

Note: This function assumes that key names are unique. If more than one system extension key has the same name, a function call deletes only the first. For more information, see "Known Issues" on page 18.

Parameters

key	A key that identifies the field to be deleted. To delete the first extension field encountered, set the argument value to "*" (used in repeated calls to delete all extension fields for the system).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ENTER_EDIT_SYSTEM_TEXT

Sd_DelTokenExtension

Function Prototype

int Sd_DelTokenExtension(char *key, char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DelTokenExtension key tokenSerialNumber

Description

Deletes an extension field with its data from a token record in the database. The field is identified by the key specified in the function call.

Note: This function assumes that key names are unique. If more than one token extension key has the same name, a function call deletes only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Key that identifies the field to be deleted. To delete the first extension field encountered, set the argument value to "*" (used in repeated calls to delete all extension fields for the token).
tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_TOKEN

Sd_DelUserExtension

Function Prototype

int Sd_DelUserExtension(char *key, char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DelUserExtension key tokenSerialOrLogin

Description

Deletes an extension field with its data from a user record in the database. The user record may be identified by login or by a token serial number assigned to the user. The extension field is identified by the key specified in the function call.

Note: This function assumes that key names are unique. If more than one user extension key has the same name, a function call deletes only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Key that identifies the extension to be deleted. To delete the first extension encountered, set the argument value to "*" (used in repeated calls to delete all extensions for the specified user).
tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_USER

Sd_DeployToken

Function Prototype

int Sd_DeployToken (int rangeMode, char *startRange, char *endRange, char *passWord, int copyProtect, int overOption, char *logFile, char *closeOption, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DeployToken rangeMode startRange endRange passWord, [copyProtect] [overOption] [logFile] [closeOption]

Description

Specifies a range of software tokens and deploys them to assigned users. Prior to calling this function, all users must be in the RSA ACE/Server database, and all software tokens included in the range must be assigned to a user. The tokens can be deployed to users as specified by serial number, or default login. The following table lists acceptable values for use with **rangeMode**, the affect each value has on subsequent parameters, and the action taken when the function is called.

rangeMode	startRange	endRange	Action Taken
0	token serial number	Ignored	Deploys one assigned software token specified by the serial number in the **startRange** parameter.
1	Ignored	Ignored	Deploys all assigned software tokens.
2	token serial number	token serial number	Deploys all assigned software tokens within the specified range of serial numbers.
3	User default login	Ignored	Deploys all software tokens that are assigned to a user with the default login specified in **startRange**.
4	User default login	User default login	Deploys all software tokens that are assigned to a range of users specified by a default login provided with **startRange** and **endRange**.

Parameters

rangeMode	Specifies criteria used to deploy assigned software tokens either by serial number or user default login. See the table under “Description” for details.
startRange	The beginning software token serial number or user default login in a range. If **rangeMode** is set to “1”, this argument is ignored. See the table under “Description” for details.
endRange	The ending software token serial number or user default login in a range. If **rangeMode** is set to either “0”, “1”, or “3”, this argument is ignored. See the table under “Description” for details.
passWord	Specifies that a password be provided by an administrator in order to access an RSA SecurID software token XML file. If there is no password associated with the file, an empty string may be passed.
copyProtect	Specifies whether the copy protection is enabled for the software tokens in the XML file. If copy protection is enabled, the Software Token record cannot be removed from the directory in which it is installed on a user’s computer. If “0”, copy protection is disabled; if any other value, copy protection is enabled.
overOption	Overwrites the output of a previously generated XML file. If any value other than “0”, the file is overwritten; if “0” is used, and a previous version of the file exists, an error is returned.
logfile	The name of a log file containing the status of the deployment operation.
CloseOption	Use one of these values when calling the function once, or repeatedly: **-a**: adds ranges of software tokens to an XML file resulting from each instance of the function being called repeatedly. The file is kept open and updated with each range of software tokens. **-c**: when the function is called once, software tokens are NOT added to an XML file, and the file is closed when the function completes. If the function is called once and an empty string is passed, the resulting range of software tokens are added to an XML file, and the file is closed when the function completes.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value “1”, “2”, “3” or “5”) if an error condition exists.

Return Text

A status string indicating success or failure.

- “1”: An invalid range of software tokens was passed.
- “2”: An empty range of software tokens was passed.
- “3”: The function completed, but not all specified software tokens were added to the .XML file
- “5”: The function completed, but only one specified software token was added to the .XML file.

Refer to the RSA ACE/Server database log for details.

Logged Events

DEPLOY_XML

Sd_DisableGroupOnClient

Function Prototype

int Sd_DisableGroupOnClient(char *groupName, char *clientName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DisableGroupOnClient groupName clientName

Description

Disables a group on a specified Agent Host so that members of the group cannot authenticate on that Agent Host.

The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite. See "USESITE" on page 14 for more information on defining separators.

Parameters

groupName	Name of the group to be disabled (optionally including a suffixed site name).
clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DEL_GROUP_FROM_CLIENT

Sd_DisableToken

Function Prototype

int Sd_DisableToken(char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DisableToken tokenSerialNumber

Description

Disables the specified token so that it cannot be used to authenticate.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DISABLE_TOKEN

Sd_DumpHistory

Function Prototype

int Sd_DumpHistory (int month, int day, int year, int daysHistory, char * fileName, int truncateOpt, int burstSize, char *msgBuf, int bufSize);

Tcl Function Call

Sd_DumpHistory month day year daysHistory [-f fileName] [-b burstSize] [truncateOpt]

Description

Dumps events in the RSA ACE/Server log from the beginning to a specified date. The events can be dumped to a file, truncated (removed) from the log, or both. There are two ways to specify the date:

- Absolutely, using the **month**, **day**, and **year** parameters.
- Relatively, using the **daysHistory** argument to specify a number of days prior to but not including the current date. If this argument is zero or an empty string (“”), month, day, and year are used. If its value is greater than zero, month, day, and year are ignored.

Note the following:

- The events dumped include all those from the beginning of the log up to but not including the specified date.
- The current date is not included in the calculation.

To ensure that all the data is dumped for a given time period, be sure to add an extra day to the number of days you specify with the **daysHistory** argument. Or, if you use the **month**, **day**, and **year** parameters, specify a date that includes one extra day added on to the specified day.

You can dump the log events of the current day by using month, day, and year to specify a future date. This leaves the log empty except for the single DEL_LOG_BY_DATE entry placed there by this function call.

The following table shows how the **fileName** and **truncateOpt** parameters are used to determine whether log entries are written only, written and deleted, or deleted only.

Value of fileName	Value of truncateOpt	Action taken on specified log entries
myfile.log	0	Copied to myfile.log.
myfile.log	Greater than 0	Written to myfile.log and deleted from the log.
“”	Greater than 0	Deleted from the log.
“”	0	No action. Function call returns an error.

- A value greater than zero for **truncateOpt** removes the specified events permanently from the log. It does not eliminate them from the dump file.
- You must either use the **fileName** argument to dump events to a file, use the **truncateOpt** argument to remove events from the log, or do both. If you do neither, the function call results in an error.
- Depending on the number of records being truncated, the operation of **Sd_DumpHistory** can take a long time to complete. To prevent problems that might otherwise occur, this function is coded with an infinite timeout interval.

Parameters

month Two-digit number representing the month. The date is the first date to be exempted from the dump operation. All **dates** prior to it will be affected.

day Two-digit number representing the day.

year Four-digit number representing the year.

daysHistory Number of days prior to the current date to exempt from the dump operation. For example, if the value is 3, events prior to the date three days before the current date are dumped. If this argument has a value greater than zero, **month**, **day**, and **year** parameters are disregarded.

fileName If not an empty string (“”), the selected log entries are dumped in text format to a file that is given the name specified in this argument. If **fileName** is an empty string, **truncateOpt** must have a value greater than zero.

truncateOpt	If the value is greater than zero, the selected entries are deleted permanently from the log. If the value is zero, nothing is deleted, but **fileName** must specify a file to which the log entries are to be written.
burstSize	This value controls the number of entries accumulated before they are removed from the log table when the **truncateOpt** argument is used. Value is an integer and does not include -b as in the Tcl function call. Increasing **burstSize** from its default of 100 to reduce the number of deletion operations can improve performance. The maximum value is 5000.

Input Parameters: Tcl

All input parameters except **fileName**, **burstSize** and **truncateOpt** are identical to the corresponding C input parameters. Note that the order of these two parameters is different: in a C function call, **truncateOpt** comes first. In a Tcl function call, **burstSize** comes first.

fileName	Value must be either of -f followed by a file name to have the selected log entries saved to that file, or an empty string (“”) to have the entries removed without being saved. If **fileName** is “”, **truncateOpt** must be **-t**.
burstSize	This value controls the number of entries accumulated before they are removed from the log table (when the **truncateOpt** argument is used). Value is **-b** followed by a number, for example, -b 500. See the description of the C input argument for additional details.
truncateOpt	Value must be **-t**. If this argument is present, the selected log entries are deleted. If this argument is omitted, **fileName** must specify a file.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DEL_LOG_BY_DATE

Sd_DynamicSelect

Function Prototype

int Sd_DynamicSelect(char *fileOrPoolName, int iFormat, int iHeader, int iNumRecors, int iParseData, char *szColumnAlias, char * szExtender, char *szSQL1, char *szSQL2, char *szSQL3, char *szSQL4, char *msgBuf,int bufSize);

Tcl Function Call

Sd_DynamicSelect fileOrPoolName iFormat iHeader iNumRecords iParseData szColumnAlias szExtender szSQL1 [szSQL2] [szSQL3] [szSQL4]

Description

Creates customized queries of RSA ACE/Server database information. Query statements are formatted in SQL (Structure Query Language). Refer to the description of this function in **admexamp.c** which contains an example SQL statement.

Note: When formatting SQL statements in Tcl, you must place backslashes immediately before any instances of double quotes within your SQL statement. In addition, use either of the following formats to specify a date:

mm`.`*dd*`.`*yyyy* or `DATE(\"`*mm*`/`*dd*`/`*yyyy*`\")`

where *mm* is the month, *dd* is the day, and *yyyy* is the year.

Parameters

fileOrPoolName	The name of an output file.
iFormat	Determines the format of output file. The value of this argument supersedes any extension to a name you provide with **fileOrPoolName**. 0: CSV 1: XML 2: HTML
iHeader	Specifies criteria for column headings in an output file. 0: Do not include column headings. 1: Include column headings. When you use this value with XML files, column names are used as XML tags.
iNumRecords	Specifies the maximum number of records to include in an output file. If you use zero or any negative number, all records are included.
iParseData	Determines criteria for parsing any special characters which are invalid for the output format. You can use any value to parse special characters. If you use “0”, special characters are not parsed.

szColumnAlias	Specifies the character that separates column names in the output data.
szExtender	The value of this parameter is based on the file format you specify in the **iFormat** parameter. • In CSV files: A string in which the first byte is treated as a field separator and the second byte as a text qualifier. If you pass an empty string, a comma (,) is used as the default value for field separators, and double quotes (“”) are used as the default value for text qualifiers. • In HTML files: The title of the HTML output page. • In XML files: The name of a global XML tag specified in the output file.
szSQL1	An SQL SELECT statement. The maximum length of this parameter is 1024 bytes. If you exceed this limit, you can concatenate the statement with additional parameters (**szSQL2**, **szSQL3**, and **szSQL4**.) If you do not exceed this limit, pass empty strings for **szSQL2**, **szSQL3**, and **SQL4**. **Note:** Tcl Select statements should include backslashes to terminate double quotes.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

None.

Sd_EditAttributeInProfile

Function Prototype

intSd_EditAttributeinProfile(char * profileName, int attributeNum, int sequenceNum, char *value, int valueFormat, char *msgBuf, int bufSize);

Tcl Function Call

Sd_EditAttributeInProfile profileName attributeNum sequenceNum value [valueFormat]

Description

Modifies an attribute value identified by number and sequence in a specified profile. If 0 is used with the **sequenceNum** argument, the function modifies the first instance of the attribute value. When the **valueFormat** argument is used, a value is interpreted depending upon its type. The following table lists allowed value types, variables for use with **valueFormat**, and explains the interpretation of a value and actions taken in the database.

Value Type	ValueFormat Variable	Interpretation/Action Taken
String	0	Assigns the value to an attribute as a string.
	1	First name value is assigned dynamically during RADIUS authentication.
	2	Last name value is assigned dynamically during RADIUS authentication.
	3	Login value is assigned dynamically during RADIUS authentication.
	4	Shell value is assigned dynamically during RADIUS authentication.
	5	Interprets the value as a prefix to the extension data key of a user record during user authentication; assigns the attribute value to all user extension records containing the prefix. **Note:** When specifying this variable, ensure that the attribute is defined in the dictionary so as to allow for multiple instances in the profile.
Integer	0	Value is interpreted as a decimal integer.
	1	Value is interpreted as a hexadecimal integer.
IP Address	0	Value is interpreted as an IP address in dotted format.
	1	Value is interpreted as an IP address in hexadecimal format.
	2	Value is interpreted as an IP address in decimal format.

Value Type	ValueFormat Variable	Interpretation/Action Taken
Date	None	Interprets the string as a date when entered in the following format: MM/DD/YYYYY HH:MM:SS If the time portion of the string is omitted, the date portion of the string is used calculate the time.

Parameters

profileName Name of the profile.

attributeNum Attribute number in the database.

sequenceNum Specifies the order in which to apply multiply-defined attributes.

value Value assigned to the attribute.

valueFormat Format type used to generate the value and assign it to the attribute. See the table under "Description**."**

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Internal sequence number of an attribute value.

Logged Events

EDITED_PROFILE

Sd_EmergencyAccessFixed

Function Prototype

intSd_EmergencyAccessFixed (char tokenSerial, char *lostPassword,
char *dateExpire, int hourExpire, int lifeTime, char *msgBuf, int bufSize);

Tcl Function Call

Sd_EmergencyAccessFixed tokenSerial lostPassword [dateExpire hourExpire]
[lifeTime]

Description

Sets the status of token (identified by a token serial number) to “Fixed” and assigns a fixed password. The lifetime of the fixed password can be defined in local time by using either the **dateExpire**, **hourExpire**, or **lifeTime** parameters.

Parameters

tokenSerial	Token serial number: must be 12 characters. Insert leading zeros as needed to meet this requirement, for example, 000000123456.
lostPassword	Specifies a fixed password for the token.
dateExpire	Expiration date of the fixed password.
hourExpire	Hour in which the fixed password expires.
lifeTime	Number of hours until emergency access mode expires (default 24). RSA ACE/Server cannot accept fixed passwords with expiration dates of more than 210,240 hours (24 years). If the **lifeTime** argument is greater than 0, **dateExpire** and **hourExpire** are ignored.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

LOST_TOKEN

Sd_EmergencyAccessOff

Function Prototype

int Sd_EmergencyAccessOff(char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_EmergencyAccessOff tokenSerialNumber

Description

Switches off emergency access mode for the specified token. The user's one-time password is destroyed and the status of the original token is changed from Lost to Not Lost.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

FOUND_TOKEN

Sd_EmergencyAccessOTP

Function Prototype

intSd_EmergencyAccessOTP (char tokenSerial, int number, int length, int flags, char *dateExpire, int hourExpire, int lifeTime, char *msgBuf, int bufSize);

Tcl Function Call

Sd_EmergencyAccessOTP tokenSerial [number] [length] [flags] [dateExpire hourExpire] [lifeTime]

Description

Sets the status of a token (identified by a token serial number) to “lost” and generates a set of one-time passwords for the token. By default, this function returns a set of two one-time passwords (default 6 digits). You can specify a larger number of passwords. These are given to the user and can be used for authentication. The lifetime of the one-time password can be defined in local time by using either the **dateExpire**, **hourExpire**, or **lifeTime** parameters. (You cannot use **Sd_EmergencyAccessOn** to generate fixed passwords for lost tokens. It generates one-time passwords only.)

Note: You can specify a greater number of one-time passwords to be generated, provided the maximum of 50 such passwords on file for a single user is not exceeded. If you request a number that will raise the total above 50, the request is automatically reduced to a smaller number. To clear tokencodes generated through **Sd_EmergencyAccessOTP**, call the **Sd_EmergencyAccessOff** function. To generate new tokencodes to replace the ones you have cleared, call **Sd_EmergencyAccessOTP** again.

Parameters

tokenSerial	Token serial number: must be 12 characters. Insert leading zeros as needed to meet this requirement, for example, 000000123456.
number	Number of tokencodes to generate, (default is 2).
length	Length of the one-time password. The range is 4-8, and the default is 6.

flags	Settings for the following flags: 1 digits only 2 letters only 3 digits and characters only 4 punctuation only 5 digits and symbols only 6 letters and symbols only 7 letters, integers and symbols
dateExpire	Expiration date of the one-time password.
hourExpire	Hour of expiration.
lifeTime	Number of hours until emergency access mode expires (default 24). RSA ACE/Server cannot accept one-time passwords with expiration dates of more than 210,240 hours (24 years). If the **lifeTime** argument is greater than 0, **dateExpire** and **hourExpire** are ignored.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

LOST_TOKEN

Sd_EmergencyAccessOn

Function Prototype

int Sd_EmergencyAccessOn(char *tokenSerialNumber, int number, int lifetime, int digits, char *msgBuf, int bufSize);

Tcl Function Call

Sd_EmergencyAccessOn tokenSerialNumber number lifetime digits

Description

Sets the status of a token (identified by a token serial number) to Lost and puts the token in emergency access mode for the number of hours specified in the lifetime argument.

By default, this function returns a set of two one-time passwords (default 6 digits). You can specify a larger number of passwords. These are given to the user and can be used for authentication. (You cannot use **Sd_EmergencyAccessOn** to generate fixed passwords for lost tokens. It generates one-time passwords only.)

This action does not affect other tokens, if any, assigned to the same user. Once a token is in emergency access mode, subsequent calls to this function specifying the same token generate additional sets of different one-time passwords.

Note: This function is retained for backward compatibility with prior versions of the RSA ACE/Server. Use the function **Sd_EmergencyAccessOTP**.

Note: You can specify a greater number of one-time passwords to be generated, provided the maximum of 50 such passwords on file for a single token is not exceeded. If you request a number that will raise the total above 50, the request is automatically reduced to a smaller number.

Note: To clear tokencodes generated through **Sd_EmergencyAccessOn**, call the **Sd_EmergencyAccessOff** function. To generate new tokencodes to replace the ones you have cleared, call **Sd_EmergencyAccessOn** again.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.
number	Number of tokencodes to generate (default 2).
lifetime	Number of hours until emergency access mode expires (default 24). RSA ACE/Server cannot accept one-time passwords with expiration dates of more than 210,240 hours (24 years).
digits	Number of digits in tokencode to be generated.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Comma-separated list of tokencodes (two by default) to be given to the user for authentication while in emergency access mode.

Logged Events

LOST_TOKEN

Sd_EnableGroupOnClient

Function Prototype

int Sd_EnableGroupOnClient(char *groupName, char *clientName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_EnableGroupOnClient groupName clientName

Description

Enables a group of users on an Agent Host so that all members of the group can authenticate on that Agent Host. The function call must specify an existing group and Agent Host.

The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite.

Parameters

groupName	Name of the group to be enabled (optionally including a suffixed site name).
clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADD_GROUP_TO_CLIENT

Sd_EnableLoginOnClient

Function Prototype

int Sd_EnableLoginOnClient(char *clientLogin, char *clientName, char *clientShell, char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_EnableLoginOnClient clientLogin clientName clientShell tokenSerialOrLogin

Description

Enables a user identified by login or by token serial number on an Agent Host. A special login and shell for the Agent Host can be specified.

Parameters

clientLogin	Agent Host login (48 characters maximum). If value is an empty string (“”), the user’s default login is used.
clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.
clientShell	Agent Host shell. If value is an empty string (“”), the user’s default shell is used. Maximum 256 characters.
tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user’s default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ADD_USER_TO_CLIENT

Sd_EnableToken

Function Prototype

int Sd_EnableToken(char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_EnableToken tokenSerialNumber

Description

Enables the specified token so that the user can authenticate on Agent Hosts where he or she is enabled.

Note: Any attempt to enable a token that is *already* enabled with not generate an error message.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ENABLE_TOKEN

Sd_ExportTokens

Function Prototype

intSd_ExportTokens(char *filename, int ExportMode, char *param1, char *param2, OutputFormat char *msgBuf, int bufSize);

Tcl Function Call

Sd_ExportTokens filename ExportMode param1 param2 OutputFormat

Description

Exports a category of tokens and user records to a specified file. The following table shows the tokens that are exported when each listed value is used with the **ExportMode** argument, and the required input for param1 and param2.

Mode value	Tokens exported	param1	param2
0	all tokens	Ignored	Ignored
1	one token	Serial number	Ignored
2	all assigned tokens	Ignored	Ignored
3	all unassigned tokens	Ignored	Ignored
4	all available tokens	Ignored	Ignored
5	all enabled tokens	Ignored	Ignored
6	all assigned disabled tokens	Ignored	Ignored
7	all lost tokens	Ignored	Ignored
8	a range of tokens by expiration	Date or number of days	Ignored
9	a range of tokens by expiration, and all tokens that have expired	Date or number of days	Ignored
10	a range of tokens by serial number; **Note:** the serial numbers provided for param1 and param2 should be in ascending order	serial number	serial number

Parameters

filename	File where the exported tokens are sent.
ExportMode	Specifies the category of tokens to export.
param1	Sets criteria for the category of tokens selected with the mode argument. If expressed in the form ***mm/dd/yyyy***, a date; otherwise, a number of days, or a serial number (see table under "Description").
param2	Sets criteria for the category of tokens selected with the mode argument, when it is set to "10". In all other instances, this argument is ignored (see table under "Description").
OutputFormat	Determines the output format. If 0, exports tokens in ASCII format (.ASC). If any other numeric value, the dump format is used (.DMP).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EXPORT_ALL_TOKENS
EXPORT_ONE_TOKEN
EXPORT_ASSIGNED_TOKENS
EXPORT_UNASSIGNED_TOKENS
EXPORT_AVAILABLE_TOKENS
EXPORT_ENABLED_TOKENS
EXPORT_DISABLED_TOKENS
EXPORT_LOST_TOKENS
EXPORT_EXPIRE_TOKENS
EXPORT_RANGE_TOKENS

Sd_GetAdminLevel

Function Prototype

intSd_GetAdminLevel (char tokenSerialOrLogin, char msgBugf, int bufSize)

Tcl Function Call

Sd_GetAdminLevel [tokenSerialOrLogin]

Description

Retrieves scope information about a specified administrator. If **tokenSerialOrLogin** is omitted (Tcl), or is an empty string, the privileges of the current logged administrator are reported.

Information	Meaning
Administrative level	realm, site, group
Task list name	Name of the administrator's assigned task list.
Site name	Name of the site the administrator is able to administer.
Group name	Name of the group the administrator is able to administer.

Parameters

tokenSerialOrLogin When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Retrieved information, comma-delimited. See "Description" of this function for details.

Logged Events

LIST_ADMIN_USERS

Sd_GetClientInfo

Function Prototype

int Sd_GetClientInfo(char *clientName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_GetClientInfo clientName

Description

Note: This function is retained for backward compatibility. Use the function **Sd_ListAgentHost**.

Retrieves the following information about the specified Agent Host.

Information	Value	Meaning
Agent Host name	None	The name of the Agent Host.
Agent Host network address	None	The network IP address of the Agent Host.
Site (if assigned to a site)	None	The name of a site (or sites) to which the Agent Host is assigned.
Agent type	0 1 2 3 4	UNIX Agent Communication server Single-transaction communication Server Net OS Agent NetSP Agent
Encryption type	0 1	SDI DES
Create Node Secret?	0 1	FALSE TRUE
Open to all?	0 1	FALSE TRUE
Can search remote realms?	0 1	FALSE TRUE

Parameters

clientName Name of the Agent Host for which information is to be retrieved: must be the full version of the name stored in the RSA ACE/Server database, for example, **pc_client.server.com**. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Retrieved information, comma-delimited. See “Description” of this function for details.

Logged Events

LIST_ONE_CLIENT

Sd_GetLastError

Function Prototype

int Sd_GetLastError (int mode, char *msgBuf, int bufSize);

Tcl Function Call

Sd_GetLastError [mode]

Description

This function retrieves an error code returned from the last function that was called (if the function failed). Use one of the values listed in the following table with the **mode** argument to specify the output format of the retrieved error information.

Mode value	Output Information
0	An error code indicating failure.
1	An error message indicating failure.
2	The name of the function that failed.

Parameters

mode	Determines the output format of error information. See the table under “Description” for details.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The error code, message, or function name.

Logged Events

None.

Sd_GetLDAPData

Function Prototype

intSd_GetLDAPData(char *tokenSerialOrLogin,char *msgBuf, int bufSize);

Tcl Function Call

Sd_GetLDAPData tokenSerialOrLogin

Description

Uses either the user's login or a token serial number assigned to the user to retrieve an LDAP source field.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

LDAP data field.

Logged Events

LISTED_LDAP_INFO

Sd_GetReplacementStatus

Function Prototype

int Sd_GetReplacementStatus(char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_GetReplacementStatus tokenSerialNumber

Description

Identifies the specified token as the original or the replacement within a replacement pair, and reports the serial number of the other member of the pair.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

If the token is an original: “O <- *nnnnnnnnnnnn*” where *nnnnnnnnnnnn* is the serial number of the replacement token. If the token is a replacement: “R -> *nnnnnnnnnnnn*” where *nnnnnnnnnnnn* is the serial number of the original token. If the token is not a member or a replacement pair: “Token Not In Pair.”

Logged Events

LIST_ONE_TOKEN

Sd_GetSerialByLogin

Function Prototype

int Sd_GetSerialByLogin(char *defaultLogin, char *number, char *msgBuf, int bufSize);

Tcl Function Call

Sd_GetSerialByLogin defaultLogin [number]

Description

Since a user is uniquely identified by a token serial number, most API functions use the token serial number to specify the user. When this number is not known, this function provides a way to look up token serial numbers by the user's default login name. **Sd_GetSerialByLogin** returns the serial numbers of all tokens assigned to the specified user.

Note: The older function **Sd_ListSerialByLogin** is retained for backward compatibility.

Parameters

number	Number of token serial numbers listed as specified by the **defaultLogin** argument. For all tokens assigned to the user, set the value to "0".
defaultLogin	Default login of the user for whom token serial numbers are to be retrieved.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Token serial number or numbers assigned to the user, comma-delimited. If none are assigned, the default list terminator Done (or another string specified through **Sd_SetSymbols**).

Logged Events

LIST_RANGE_TOKENS

Sd_HexMD5

Function Prototype

int Sd_HexMD5(char *text, char *msgBuf, int bufSize);

Tcl Function Call

Sd_HexMD5 text

Description

Reads a text string passed in the function call, creates an MD5 digest of the string, and returns its value in hexadecimal notation.

Parameters

text	Any text string.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Hexadecimal value of the MD5 digest.

Logged Events

None.

Sd_ImportToken

Function Prototype

intSd_ImportToken (char tokenSerialNumber, char *ascFilePath, char*msgBuf, int bufSize);

Tcl Function Call

Sd_ImportToken tokenSerialNumber ascFilePath

Description

Extracts a token record identified by serial number from a specified token (ASCII) file and imports it into the RSA ACE/Server database. If there is any user data associated with the token in the ASCII file, the data is discarded during import. If the token previously exists in the Server database, the import operation will fail.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.
ascFilePath	File specification of the token record file (ASCII) from which the token is to be extracted. For example: **c:/ace/prog/examples**. This file must be accessible from the RSA ACE/Server machine.

Input Parameters: Tcl

Identical to C input parameters. For **ascFilePath**, path and **fileName** must be specified using forward slashes.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

IMPORT_TOKENS

Sd_ImportTokenFile

Function Prototype

Sd_ImportTokenFile(char *fileName, int importMode, char *reportFile, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ImportTokenFile fileName importMode [reportFile]

Description

Imports a token record identified by its serial number from a file into the RSA ACE/Server database. The import operation performed is specified by applied flags.

Mode Value	Import Operation Performed/Status in Server Database
0	All tokens not found in the database are imported with no associated user data.
1	All tokens are imported with no associated user data, and are unassigned in the database. Pre-existing tokens not belonging to an administrator are overwritten and unassigned. Pre-existing tokens belonging to an administrator remain assigned to the administrator.
2	All tokens, except those pre-existing in the database, are imported with associated user data. The following conditions apply when a token is assigned in an ASCII file: When the user and token are not in the database already, the token is imported, a user record is created, and the token is assigned to the user. When the user is in the database already with the same login as found in the ASCII file, and the token is not in the database, the existing user record is not overwritten, and the token is imported as unassigned if: • The user is an administrator. • The user is a remote user. Otherwise, if the user has not reached the limit of three assigned tokens, imports token and assigns it to the user without overwriting the user record. When the token already exists in the database, the token is not imported. When the user and token are in database already, the token is not imported. When the user and token are in the database already, but the token is unassigned, the token is not imported. When the user and token are in the database already, but the token is not unassigned in the database, the user and token records are not overwritten, and the token is not assigned to the user. When a token is already in the database, but the token assignment in the database is different from the ASCII file assignment, the token is not imported.

Mode Value	Import Operation Performed/Status in Server Database
3	Imports tokens with associated user data, and overwrites any duplicate records. If you specify IMPORT_PRESERVE, the function does not overwrite any existing RADIUS profile or LDAP information for the user. The following six conditions apply when a token is assigned in an ASCII file: When the user and token are not in the database already, imports token, creates user record for user and assigns token to user. When the user is in the database already and the token is not in the database, overwrites the existing user record, imports token and assigns token to user if: • Last names are the same. • The user is not an administrator. • The user is not a remote user. • The user has not reached the limit of three assigned tokens. Otherwise, retains existing user record, imports token as unassigned. When the user is not in the database already, and the token is in the database, creates a user record and assigns token to user. Does not import token if: • The token is assigned to an administrator already. • The token is a replacement token. • The token has an assigned replacement token. When the user and token are in database already, and the token is assigned to the user, overwrites token and user. Does not import token or overwrite user record if: • The token is assigned to an administrator already. • The token is a replacement token. • The token has an assigned replacement token. Otherwise, overwrites the token record as unassigned.

Parameters

fileName — Complete file specification (path and file name using forward slashes) of the token record file (ASCII) from which the token is to be extracted. For example: **c:/ace/prog/examples**. This file must be accessible from the RSA ACE/Server machine.

importMode — Determines the import operation performed (a combination of the following flag settings). See table under "Description" for details of each mode of the import operation.

Flag 1: Overwrite all token records previously existing in the database.

Flag 2: Import all user data associated with each token record.

	Flag 3: Maintain all LDAP data and user profile information for a user already existing in the Server database. Flag 4: Commit all import operations specified by flags 1, 2, and 3 to the RSA ACE/Server database.
reportFile	File containing the results of the import file.

Note: Because the **importMode** flags are defined as constants in the header file, pipe notation can be used in C function calls with these constant names: IMPORT_OVER (= 1), IMPORT_USERS (= 2), IMPORT_PRESERVE (= 4), IMPORT_COMMIT (= 8). For example, to set all flags to TRUE, use the following argument:

"Mode = IMPORT_OVER | IMPORT_USERS | IMPORT_PRESERVE | IMPORT_COMMIT."

When importing a large token record file, RSA Security recommends that you specify IMPORT_COMMIT so that each token record can be committed to the database individually during the import process.

Input Parameters: Tcl

Identical to C input parameters. For **fileName**, path and file name must be specified using forward slashes.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

IMPORT_TOKENS

Sd_ImportTokenFileExt

Function Prototype

Sd_ImportTokenFileExt(char *fileOrPoolName, int importMode, char *passWord, char *certOrLicenseName, char *reportFile, char *reportFile, char *msgBuf, int bufsize)

Tcl Function Prototype

Sd_ImportTokenFileExt fileOrPoolName importMode [passWord] [certOrLicenseName][reportFile]

Description

Imports all tokens from either an ASCII, XML, or DMP file into the RSA ACE/Server database. The import operation performed is specified by applied flags.

Note: The flags used are identical to those used with Sd_ImportTokenFile. See "Description" under "Sd_ImportTokenFile" on page 103 for details.

Parameters

fileOrPoolName	Complete file specification (path and file name using forward slashes) of the token record file from which the token is to be extracted. For example: **c:/ace/prog/examples**. This file must be accessible from the RSA ACE/Server machine.
importMode	Determines the import operation performed (a combination of the following flag settings). Flag 1 Overwrite all token records previously existing in the database. Flag 2 Import all user data associated with each token record. Flag 3 Maintain all LDAP data and user profile information for a user already existing in the Server database. Flag 4 Commit all import operations specified by flags 1,2, and 3 to the RSA ACE/Server database.
passWord	The password (if any) associated with the token file. If the XML file containing the tokens was purchased from RSA Security and a password is required to access the file, this argument must be used.

certOrLicenseName	The name of a license record (when importing a .DMP file.) For this release, a certificate file is not required; the argument is named as such for future implementation.
reportFile	The name of a report file generated from the import operation.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

IMPORT_TOKENS

Sd_IsEmergencyAccess

Function Prototype

int Sd_IsEmergencyAccess (char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_IsEmergencyAccess tokenSerialNumber

Description

Checks a token for emergency access mode, and returns TRUE if the result is positive or FALSE if it is negative.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

TRUE (the specified token is in emergency access mode) or FALSE (the token is not in emergency access mode).

Logged Events

LIST_ONE_TOKEN

Sd_LastErrorMode

Function Prototype

int Sd_LastErrorMode (int mode, char *msgBuf, int bufSize);

Tcl Function Call

Sd_LastErrorMode Mode

Description

This function determines how the error information (code, error message, and function name) of the last failed function is stored and maintained in the database.

Parameters

Mode	Values can be either “0” or any non-zero integer. Use “0” to clear the error information upon successful completion of any function other than **Sd_LastErrorMode** or **Sd_GetLastError**. Use any non-zero integer to maintain the last failed function error information in the database upon successful completion of any subsequent function.

Return Values

OK (value 0).

Logged Events

None.

Sd_ListAdministrators

Function Prototype

int Sd_ListAdministrators(char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListAdministrators [closeOpt]

Description

Lists the default login for each administrator. This function must be called repeatedly to get all administrator logins, and the database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Administrator's default login. The list terminator string is returned when no more administrators are found.

Logged Events

LIST_ALL_ADMIN_USERS

Sd_ListAgentHostInfo

Function Prototype

int Sd_ListAgentHostInfo(char *agentHostName, int outputFormat, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListAgentHostInfo AgentHostName [outputFormat]

Description

Retrieves information about the specified Agent Host when each of the following values are used with the output format argument.

Value	Returned Fields	Meaning
0	Agent Host name	The name of the Agent Host machine
	Agent Host's network address	The IP address of the Agent Host machine.
	Site Name	The name of a site to which the Agent Host belongs.
	Agent Host type	UNIX Agent Host Communication server Single-transaction communication server Net OS Agent Host NetSP Agent Host
	Encryption type	SDI DES
	Create Node Secret?	FALSE TRUE
	Open to all?	FALSE TRUE
	Can search remote realms	FALSE TRUE
	Require Name Lock	FALSE TRUE
1	All fields returned with "0", plus:	
	Acting Master name	Name of an Acting Master Server
	Acting Master IP address	IP address of an Acting Master Server.
	Acting Slave name	Name of an Acting Slave Server
	Acting Slave IP address	IP address of an Acting Slave Server.
2	All fields returned with "0", plus:	
	Shared secret	The encryption key established between the Agent Host and RADIUS Server.

Value	Returned Fields	Meaning
3	All fields returned with “0”, plus: Acting Master Acting Slave Shared secret	
4	Acting Master (name and IP address) Acting Slave (name and IP address)	
5	Shared secret only	

Parameters

agentHostName The full name of the Agent Host as recorded in RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.

outputFormat Specifies fields to be retrieved. See table under “Description” for details.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Retrieved information, comma-delimited. See “Description” of this function for details.

Logged Events

LIST_ONE_CLIENT

Sd_ListAssignedTokens

Function Prototype

int Sd_ListAssignedTokens(char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListAssignedTokens [closeOpt]

Description

Lists the serial numbers of assigned tokens in the RSA ACE/Server database. The function returns one token each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Token serial number. When all assigned tokens have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_ALL_TOKENS

Sd_ListAttributes

Function Prototype

int Sd_ListAttributes(char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListAttributes [closeOpt]

Description

Lists all attributes in the database. The function returns one attribute each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

CloseOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

AttributeNum, AttributeName, Type, Multiple instances, User Configurable Where type is: string | integer | date | ipaddr

Multiple instances – 0 or 1

User Configurable – 0 or 1

Logged Events

LIST_DICTIONARY

Sd_ListAttributesInProfile

Function Prototype

intSd_ListAttributesInProfile(char * profileName, int attributeNum, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListAttributesInProfile profileName [attributeNum] [closeOpt]

Description

Lists attribute values for a profile specified by **profileName**. If a valid attribute number is used with the **attributeNum** argument, the function lists only values for that particular attribute in the profile. If an invalid number is used, all attributes in the profile are listed.

Parameters

profileName	Name of the profile in the database.
attributeNum	Number of the attribute in the database.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

AttributeNum, screenValue, sequenceNum where the screenValue is the same value one will see in the sdadmin profile administration.

Logged Events

LIST_ONE_PROFILE

Sd_ListClientActivations

Function Prototype

int Sd_ListClientActivations(char *tokenSerialOrLogin, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListClientActivations tokenSerialOrLogin [closeOpt]

Description

Lists the Agent Hosts on which a user (identified either by login or by token serial number) is activated. The function returns one Agent Host each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The text output for each Agent Host consists of the Agent Host login, Agent Host shell, Agent Host name, and site name. This output is comma-delimited. When all Agent Hosts on which the user is activated have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_RANGE_CLIENTS

Sd_ListClientExtension

Function Prototype

int Sd_ListClientExtension(char *key, char *clientName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListClientExtension key clientName

Description

Lists the contents of one extension field, identified by key, from the record of the specified Agent Host.

Note: This function assumes that key names are unique. If more than one Agent Host extension key has the same name, a function call retrieves only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Extension field data.

Logged Events

EDITED_CLIENT

Sd_ListClients

Function Prototype

int Sd_ListClients(char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListClients [closeOpt]

Description

Lists the Agent Hosts in the RSA ACE/Server database. The function returns one Agent Host each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The text output consists of the Agent Host name and site name, comma-delimited. When all Agent Hosts have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_ALL_CLIENTS

Sd_ListClientsBySite

Function Prototype

int Sd_ListClientsBySite(char *siteName, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListClientsBySite siteName closeOpt

Description

Lists all Agent Hosts associated with the specified site. The function returns one Agent Host each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

siteName	Name of the site for which Agent Hosts are to be listed
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Agent Host name and site name. When all Agent Hosts have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_CLIENTS_IN_SITE

Sd_ListClientsForGroup

Function Prototype

int Sd_ListClientsForGroup(char *groupName, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListClientsForGroup groupName [closeOpt]

Description

Lists the Agent Hosts on which a specified group is enabled. The function returns one Agent Host each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite.

Parameters

groupName	Name of the group for which Agent Hosts are to be listed (optionally including a suffixed site name).
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Agent Host name and site name. When all Agent Hosts on which the group is enabled have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_RANGE_CLIENTS

Sd_ListClientType

Function Prototype

int Sd_ListClientType(char *clientName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListClientType clientName

Description

Retrieves from the Agent Host record and returns the type of the specified Agent Host: UNIX Agent, Communication server, Single-transaction communication server, Net OS Agent, or NetSP Agent. Use **Sd_GetClientInfo** to retrieve all other information about an Agent Host.

Parameters

clientName	Name of the Agent Host whose type is to be returned: must be the full version of the name stored in the RSA ACE/Server database, for example, **pc_client.server.com**. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Code indicating whether Next Token Code mode is available on the Agent Host (0 = not available, 1 = available), followed by the Agent type: UNIX Agent, communications server, single-transaction communication server, net OS Agent, or NetSP Agent.

Logged Events

LIST_AH_TYPE

Sd_ListExtensionsForClient

Function Prototype

int Sd_ListExtensionsForClient(char *clientName, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListExtensionsForClient clientName [closeOpt]

Description

Lists extension fields, with their data, from the record of the specified Agent Host. The function returns one field each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Consists of the key name and data for one extension field per function call. When all extension fields in the record have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

EDITED_CLIENT

Sd_ListExtensionsforGroup

Function Prototype

int Sd_ListExtensionsforGroup(char *groupName, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListExtensionsforGroup groupName [closeOpt]

Description

Lists extension fields, with their data, from the record of the specified group. The function returns one field each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite.

Parameters

groupName	Name of the group for which to list extension fields (optionally including a suffixed site name).
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Consists of the key name and data for one extension field per function call. When all extension fields in the record have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

EDITED_GROUP

Sd_ListExtensionsForSite

Function Prototype

int Sd_ListExtensionsForSite(char *siteName, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListExtensionsForSite siteName [closeOpt]

Description

Lists extension fields, with their data, from the record of the specified site. The function returns one field each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

siteName	Name of the site for which to list extension fields.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Consists of the key name and data for one extension field per function call. When all extension fields in the record have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

EDITED_SITE

Sd_ListExtensionsForSys

Function Prototype

int Sd_ListExtensionsForSys(char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListExtensionsForSys [closeOpt]

Description

Lists extension fields, with their data, from the system record. The function returns one field each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Key name and data for one extension field per function call. When all extension fields in the record have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

ENTER_EDIT_SYSTEM_TEXT

Sd_ListExtensionsForToken

Function Prototype

int Sd_ListExtensionsForToken(char *tokenSerialNumber, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListExtensionsForToken tokenSerialNumber [closeOpt]

Description

Lists extension fields, with their data, from the record of the specified token. The function returns one field each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Key name and data for each extension field. When all extension fields in the record have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

EDITED_TOKEN

Sd_ListExtensionsForUser

Function Prototype

int Sd_ListExtensionsForUser(char *tokenSerialOrLogin, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListExtensionsForUser tokenSerialOrLogin [closeOpt]

Description

Lists extension fields, with their data, from the record of the specified user, identified by login or by token serial number. The function returns one field each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Key name and data for one extension field per function call. When all extension fields in the record have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

EDITED_USER

Sd_ListGroupExtension

Function Prototype

int Sd_ListGroupExtension(char *key, char *groupName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListGroupExtension key groupName

Description

Lists the contents of one extension field, identified by key, from the record of the specified group.

Note: This function assumes that key names are unique. If more than one group extension key has the same name, a function call retrieves only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Key that identifies the extension to list.
groupName	The name of the group (optionally with a site name suffixed).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Extension field data.

Logged Events

EDITED_GROUP

Sd_ListGroupMembership

Function Prototype

int Sd_ListGroupMembership(char *tokenSerialOrLogin, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListGroupMembership tokenSerialOrLogin [closeOpt]

Description

Lists the groups of which a user, identified by login or by token serial number, is a member. The function returns one group each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The text output for each group consists of the group login, shell, and name, and the site name. When all groups to which the user belongs have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

ENTER_GROUP_MEMBER

Sd_ListGroups

Function Prototype

int Sd_ListGroups(char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListGroups [closeOpt]

Description

Lists all the groups in the RSA ACE/Server database. The function returns one group each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The text output for each group consists of the group name and site name. When all groups have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_ALL_GROUPS

Sd_ListGroupsByClient

Function Prototype

int Sd_ListGroupsByClient(char *clientName, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListGroupsByClient clientName closeOpt

Description

Lists all groups enabled on a specified Agent Host. The function returns one group each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The text output for each group consists of the group and site name, separated by a comma. When all groups have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_GROUPS_IN_CLIENT

Sd_ListGroupsBySite

Function Prototype

int Sd_ListGroupsBySite(char *siteName, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListGroupsBySite siteName [closeOpt]

Description

Lists all groups assigned to a specified site. The function returns one group each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

siteName	Name of the site for which to list groups.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The text output for each group consists of the group name. When all groups have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_GROUPS_IN_SITE

Sd_ListHistory

Function Prototype

int Sd_ListHistory(char *daysHistory, char *tokenSerialOrLogin, char *outputOpt, char *filterOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListHistory daysHistory tokenSerialOrLogin [-o outputOpt] [-f filterOpt]

Description

Lists the events in the activity log relating to the user (identified by login or by token serial number) that occurred between a date *n* days in the past (where *n* is the value of the daysHistory argument) and the current date. The function returns one event each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **outputOpt** argument set to **-c**. (Note that there is no **closeOpt** argument as with other functions of this type.)

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Note: When the user is specified by login, only records with the login passed in the argument are returned. Some users may have multiple logins. For example, special group or Agent Host logins that differ from the default login. The only way to be sure of retrieving all log records for such users is to set up a listing loop for each of these logins.

Note: You can use this function to list all log events for the specified period, regardless of user, if you set the value of **tokenSerialOrLogin** to the wildcard symbol "*" (asterisk). The symbol must be the only value of the argument; partial wildcards such as "abc*" or "*123" are not accepted. This feature can make **Sd_ListHistory** a convenient means for capturing recent history, because (unlike **Sd_DumpHistory**) the period it covers is specified by counting back from the current date rather than forward from the beginning of the log.

Parameters

daysHistory	Number of days before today's date to list.
tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456. Use the asterisk (*) value to list all log events regardless of user (see the note in the "Description" of this function). The **filterOpt** option is ignored.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

outputOpt	Set the value to **pretty** to have output returned in columns instead of being separated by commas. To close the search before the list terminator is returned, set the value to **-c** or close after all the events sought have been returned.
filterOpt	Filter option: if value is **no_admin**, most events performed by administrators are filtered out, making it easier to view authentication events. If the value is an empty string (""), all events are listed without filtering.

Parameters: Tcl

Only the option values are different for Tcl. The first two parameters, **daysHistory** and **TokenSerialOrLogin**, are identical to the C input parameters.

outputOpt	Set the value to **-o pretty** to have output returned in columns instead of being separated by commas. To close the search before the list terminator is returned, set the value to **-c** after all the events sought have been returned.
filterOpt	Filter option: if value is **-f no_admin**, most events performed by administrators are filtered out, making it easier to view authentication events. If the argument is omitted, all events are listed without filtering.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

For each event, output consists of the following information from the log (byte limitations are listed in parentheses): Event name (34), local date (10), local time (8), user login (48), affected user name (49), group (48), Agent Host (48), site (48), and Server names (48), message number (6-prefaced with zeros if less than 6). This output is comma-delimited by default, but is column-aligned if the function call includes the **-o pretty** option.

Logged Events

DLOAD_ALL_LOGS

Sd_ListJob

Function Prototype

int Sd_ListJob(char *jobClass, char *CloseOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListJob[jobClass] [closeOpt]

Description

The function returns one job each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Parameters

jobClass	The type of LDAP job. For RSA ACE/Server 5.2, the only input option for this argument is “ldapsync.”
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The name and class of each job (comma-separated.)

Logged Events

LIST_RANGE_JOBS

Sd_ListJobInfo

Function Prototype

int Sd_ListJobInfo(int *outputFormat, char *JobName, char *jobClass, char *msgbuf int bufSize);

Tcl Function Call

Sd_ListJobInfo outputFormat jobName [jobClass]

Description

Lists configuration data for an LDAP synchronization job based on the following allowable values used with the **outputFormat** argument.

Value	Returned Fields
0	Job Number, JobName, Job Class, Enabled flag, Run once flag, date of initial start (UTC), time of initial start (UTC), Job run interval, Name of the executable, working directory of the executable, block of parameter data, date the job was last time started (UTC), time when the job was last time started (UTC), date the job last time completed (UTC), time when the job last time completed (UTC), exit status, summary line, state.
1	The state of the job. 0: idle, 1: running, 2: has yet to run, 3: in the process of being deleted.
2	Summary: details how the job affected the database (number of users added, deleted, and updated).
3	The most recent date and time that the job was run, and the most recent date and time that the job was run and completed.
4	Exit Status: Status returned from the last time the job was run. One of the following: 7: Never run 6: Currently running 5: Deleted **Note:** These values are converted from decimal form to hexadecimal form; thus the characters “FFFFFFF” are displayed after the initial values of “5”, “6”, and “7”.

Parameters

outputFormat	Specifies the level of detail of the data listed. See table under “Description” for details.
jobName	The name of the sychronization job.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

LIST_ONE_JOB

Sd_ListProfiles

Function Prototype

intSd_ListProfiles (char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListProfiles [closeOpt]

Description

Lists the profiles in the RSA ACE/Server database. The function returns one profile each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The text output for each profile consists of the profile name. When all profiles have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_ALL_PROFILES

Sd_ListReplicas

Function Prototype

Sd_ListReplicas (char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListReplicas [closeOpt]

Description

Lists the Replica Servers in the RSA ACE/Server database. The function returns one Replica each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The text output for each Replica consists of all information from the Replica table. When all Replicas have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LISTED_REPLICAS

Sd_ListSecondaryNodes

Function Prototype

int Sd_ListSecondaryNodes(char *agentHostName, char *closeOpt, char *msgBuf

Tcl Function Call

Sd_ListSecondaryNodes agentHostName closeOpt

Description

Lists all the secondary nodes associated with the specified Agent Host. The function returns one secondary node each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

agentHostName	Name of the Agent Host for which secondary nodes are to be listed
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Secondary node name and secondary node address. When all the secondary nodes have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_SEC_NODE

Sd_ListSerialByLogin

Function Prototype

int Sd_ListSerialByLogin(char *defaultLogin, char *count, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListSerialByLogin defaultLogin [count]

Description

Since a user is uniquely identified by a token serial number, most API functions use a token serial number to specify the user. When no serial number is known, this function provides a way to look up token serial numbers by the user's default login name. There is a better way to achieve the same purpose, however; see the following note.

Note: Sd_ListSerialByLogin is an obsolete function whose behavior can cause confusion. It is retained for backward compatibility, but you are advised to use in its place the newer function **Sd_GetSerialByLogin**, which is not subject to the problem described here.

Note: To close the list properly and ensure that subsequent calls to this and other functions do not fail, **Sd_ListSerialByLogin** must always be called until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is the *only* value returned.

Note: This list function, unlike others, includes the list terminator string as part of the data it returns; the string is appended to each set of token serial numbers. To determine that a calling sequence is completed, the algorithm must require the list terminator to be returned alone and not at the end of a line of data. Calling the function twice in succession for the same login value has this effect: the second call returns only the list terminator.

Parameters

defaultLogin	User's default login string.
count	This argument, intended to designate one of several identical logins, is now obsolete because logins are required to be unique within the RSA ACE/Server database. The argument is retained for the sake of backward compatibility. To avoid errors, the value can now be any number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Returns token serial numbers assigned to the user specified by the input parameters with the list terminator at the end of the line. If the user has no tokens assigned, returns 0 and the list terminator. If there is no such user or if all users with the specified name have already been listed, returns the list terminator only.

Logged Events

LIST_RANGE_TOKENS

Sd_ListSerialByName

Function Prototype

int Sd_ListSerialByName(char *lastName, char *firstName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListSerialByName lastName [firstName]

Description

Since a user is uniquely identified by a token serial number, most API functions use the token serial number to specify the user. When this number is not known, this function provides a way to look up token serial numbers by the user's default login name. The **lastName** argument can be specified using a partial string beginning with the first letter of the user's last name. The search is not case-sensitive. In a database where multiple users have the same name, call **Sd_ListSerialByName** repeatedly to retrieve token serial numbers for all of them. The function must always be called until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned in order to close the list properly and ensure that subsequent calls to this and other functions do not fail.

Parameters

lastName	Last name string used to search for the user. May be the full name or just the first part.
firstName	First name string; optional.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Returns token serial numbers assigned to the user specified by the input parameters with the list terminator at the end of the line. If the user has no tokens assigned, returns 0 and the list terminator. If there is no such user or if all users with the specified name have already been listed, returns the list terminator only.

Logged Events

LIST_RANGE_TOKENS

Sd_ListSiteExtension

Function Prototype

int Sd_ListSiteExtension(char *key, char *siteName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListSiteExtension key siteName

Description

Lists the contents of one extension field, identified by key, from the record of the specified site.

Note: This function assumes that key names are unique. If more than one site extension key has the same name, a function call retrieves only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Key that identifies the extension to list.
siteName	Name of the site.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Extension field data.

Logged Events

EDITED_SITE

Sd_ListSites

Function Prototype

int Sd_ListSites(char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListSites [closeOpt]

Description

Lists all the sites in the RSA ACE/Server database. The function returns one site each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The text output for each site consists of the site name. When all sites have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_ALL_SITES

Sd_ListSysExtension

Function Prototype

int Sd_ListSysExtension(char *key, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListSysExtension key

Description

Lists the contents of one extension field, identified by key, from the system record.

Note: This function assumes that key names are unique. If more than one system extension key has the same name, a function call retrieves only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Key that identifies the extension to list.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Extension field data.

Logged Events

ENTER_EDIT_SYSTEM_EXT

Sd_ListTaskLists

Function Prototype

int Sd_ListTaskLists(char *closeOpt,char *msgBuf,int bufSize);

Tcl Function Call

Sd_ListTaskLists [closeOpt]

Description

Lists all the tsk lists in the RSA ACE/Server database. The function returns one list each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The text output for each task list consists of the task list name. When all task lists have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_ALL_TASKLISTS

Sd_ListTokensByField

Function Prototype

intSd_ListTokensByField (int field, int compareType,char *value, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListTokensByField field compareType value [closeOpt]

Description

Lists all tokens in the database that have the same values in one or more specified fields. This function returns one token each time it is called and must be called repeatedly to list more. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Fields and comparison type are specified as integers, using the values in the following table. The value of **Field** determines what field is compared; the value of **compareType** determines the type of comparison for some fields and supplies a value (making the value argument unnecessary) for others. The integer values in the table are associated with macro definitions in the header file **apiuser.h**, and it is recommended that you use these rather than hard-coded constants.

Tokens to List	Field	compareType	value
All	0 Any	Any 0	Ignored Ignored
By assignment All replacing tokens All unassigned tokens that have been replaced	1	 1 2	 Ignored Ignored
By token type All assigned tokens of specific type All unassigned tokens of a specific type All tokens of specific type (both assigned and unassigned)	2	 1 2 3	 Token type Token type Token type
By replacement All original tokens All replacement tokens All unassigned replacement tokens	3	 1 2 3	 Ignored Ignored Ignored

Tokens to List	Field	compareType	value
By expiration	4		
All expired tokens		1	Ignored
All assigned tokens that have expired		2	Ignored
All expired tokens and tokens expiring within a given number of days		3	Number of days
All expired assigned tokens and assigned tokens expiring within a given number of days		4	Number of days
All tokens expiring within a given number of days.		5	Number of days
All assigned tokens expiring within a given number of days		6	Number of days
By Status	5		
All assigned and disabled		1	Ignored
All assigned and enabled		2	Ignored
All with cleared PINs		3	Ignored
All in new PIN mode		4	Ignored
All in next tokencode mode		5	Ignored
All lost tokens		6	Ignored
All with lost status expired		7	Ignored
All with lost status expired, or set to expire in a number of days		8	Number of days
All with lost status expiring in a number of days		9	Number of days
By Token extension	6		
All that have extension records		1	Ignored
All that do not have extension records		2	Ignored
All that have extension records with a provided key		3	String
All that have extension records without a provided key.		4	String
By Seed	7		
All assigned		1	64|128
All unassigned		2	64|128
All		3	64|128
By Seed and Token Type	8		
All assigned		1	64|128
All unassigned		2	64|128
All		3	64|128

Tokens to List	Field	compareType	value
By Deployment (SecurID Software Tokens only)	9		64|128
Assigned and deployed more than the specified number of times		1	Number
Assigned and deployed equal to the specified number of times		2	Number
Assigned and deployed less than the specified number of times		3	Number
Unassigned and deployed		4	Ignored
Assigned but not deployed		5	Ignored
By Assignment Date	10		
		1	All token types
		2	All tokens with a password
		3	All tokens without a password

Parameters

field Specifies a category of tokens.

compareType Specifies a subcategory of tokens.

value One of several parameters (either “Number of days”, “Token type”, or “bit size”) for use with the **compareType** argument defining the scope of a subcategory (see the table under “Description”).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The token serial number.

Logged Events

LISTED_RANGE_TOKENS

Sd_ListTokenExtension

Function Prototype

int Sd_ListTokenExtension(char *key, char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListTokenExtension key tokenSerialNumber

Description

Lists the contents of one extension field, identified by key, from the record of the specified token.

Note: This function assumes that key names are unique. If more than one token extension key has the same name, a function call retrieves only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Key that identifies the extension to list.
tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456..

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Extension field data.

Logged Events

EDITED_TOKEN

Sd_ListTokenInfo

Function Prototype

int Sd_ListTokenInfo(char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListTokenInfo tokenSerialNumber

Description

Uses a token serial number to retrieve the following fields from the SDToken table in the database (see page 253). Output is comma-delimited. Where applicable, date and time is expressed in UTC (Coordinated Universal Time).

Field	Description
chSerialNum	Token serial number
PINClear	PIN status:
	0 No PIN associated with token (old PIN, if any, cleared)
	1 Token has a PIN associated with it
iNumDigits	Number of digits in the token display
iInterval	Number of seconds between display changes
dateBirth	Date the token was activated
todBirth	Time (in seconds) the token was activated
dateDeath	Date the token will shut down
todDeath	Time (in seconds) the token will shut down
dateLastLogin	Date of the last login with this token
todLastLogin	Time (in seconds) of the last login
iType	Token type:
	“0”: RSA SecurID Standard Card
	“1”: RSA SecurID PINPAD Card
	“2”: RSA SecurID Key Fob
	“4”: RSA SecurID Software Token
	“6”: RSA SecurID modem
bHex	Whether the display is hexadecimal (TRUE/FALSE)

Field	Description
bEnabled	Whether the token is enabled (TRUE/FALSE)
bNewPINMode	Whether the token is in New PIN mode (TRUE/FALSE)
iUserNum	Number of user to whom token is assigned
iNextCodeStatus	Next tokencode status: 0 Not in next tokencode mode 1 Awaiting one more good tokencode, no PIN 2 Awaiting two good tokencodes, no PIN
iBadTokenCodes	Number of bad tokencodes entered
iBadPINs	Number of bad PINs entered
datePIN	Date PIN was last changed
todPIN	Time (in seconds) PIN was last changed
dateEnabled	Date token was last enabled or disabled
todEnabled	Time (in seconds) token was enabled or disabled
dateCountsLastModified	Date token counts were last modified
todCountsLastModified	Time (in seconds) counts were last modified

Note: The following fields apply to RSA SecurID Software Tokens only. For all other tokens, the first four fields are filled with zeros and the **softPassword** field is an empty string.

Field	Description
bProtected	Whether software token was copy-protected on last deployment: 0 No 1 Yes
bDeployed	Whether software token is currently deployed: 0 No 1 Yes
iCount	Number of times token has been deployed
Reserved	Always 0; reserved for future use.
softPassword	Password stored in the software inf file

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Fields from the SDToken table in the database. See the Description of this function for a list.

Logged Events

LIST_ONE_TOKEN

Sd_ListTokenInfoExt

Function Prototype

int Sd_ListTokenInfoExt (char *tokenSerialNumber, int *outputFormat, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListTokenInfoExt tokenSerialNumber [OutputFormat]

Description

Uses a token serial number to retrieve fields from the SDToken table (See page 253). Retrievable fields can be specified by the following output values. Output is comma-delimited.

Value	Fields Retrieved
0	All fields as with Sd_ListTokenInfo.
1	All fields as with Sd_ListTokenInfo, plus: • Seed size: the size of the seed record. • bKeypad: applicable if the token has a keypad. • bLocalPin: applicable if the Pin is stored locally on a user's computer. • TokenVersion: the version of the token algorithm. • Form Factor: this field is for future use.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.
OutputFormat	Determines the output format. If 0, lists standard token information. If 1, lists additional seed information.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Fields from the SDToken table in the database. See the Description of this function for a list.

Logged Events

LIST_ONE_TOKEN

Sd_ListTokens

Function Prototype

int Sd_ListTokens(char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListTokens [closeOpt]

Description

Lists the serial numbers of all tokens, assigned and unassigned, in the RSA ACE/Server database. The function returns one token each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Token serial number. When all tokens have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_ALL_TOKENS

Sd_ListUnassignedTokens

Function Prototype

int Sd_ListUnassignedTokens(char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListUnassignedTokens [closeOpt]

Description

Lists the serial numbers of unassigned tokens in the RSA ACE/Server database. The function returns one token each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Token serial number. When all unassigned tokens have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_ALL_TOKENS

Sd_ListUserByClient

Function Prototype

int Sd_ListUserByClient(char *clientName, char *format, char *delim,
char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListUserByClient clientName [format delim] [closeOpt]

Description

Lists all users enabled on a specified Agent Host. The **format** argument specifies short format (default and Agent Host logins only) or long format, which returns the same fields as the **Sd_ListUserInfo** function. The **delim** argument specifies the delimiter to be inserted between fields. This function returns one user each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.
format	**-s** (“short”) to return the default login and Agent Host login; **-l** (“long”) to return the same data as the **Sd_ListUserInfo** function. If passed as an empty or invalid (“”), the default **-s** (short) is assumed.
delim	Character other than a comma to insert between fields in return text. If the argument is an empty string (“”), the default delimiter is a comma.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Parameters: Tcl

Identical to C input parameters, with two exceptions:

delim	Character other than a comma to insert between fields in return text. If the argument is an empty string, the default delimiter is a comma. *Position:* if only two parameters are passed in the function call, they are assumed to be **clientName** and **closeOpt**. If there are three parameters, they are assumed to be **clientName**, **format**, and **delim**. The only way to pass **closeOpt** with either **format** or **delim** is to pass all four parameters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

In short format, the user's default login and Agent Host login. In long format, data from the user record (see "Sd_ListUserInfo" for the fields). Fields are delimited by commas or by the character specified in the delim argument. When all enabled users have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_CLIENT_USERS

Sd_ListUserByGroup

Function Prototype

int Sd_ListUserByGroup(char *groupName, char *format, char *delim, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListUserByGroup groupName [format delim] [closeOpt]

Description

Lists all users that are members of a specified group. The format argument specifies short format (returns the default and group logins only) or long format, which returns the same fields as the **Sd_ListUserInfo** function. The delim argument specifies the delimiter to be inserted between fields. This function returns one user each time it is called and must be called repeatedly to list more. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

groupName	Name of the group for which to list users
format	**-s** (“short”) to return the default login and group login; **-l** (“long”) to return the same data as the **Sd_ListUser** function. If passed as an empty or invalid string (“”), the default **-s** is assumed.
delim	Character other than a comma to insert between fields in return text. If the argument is an empty or invalid (“”), the default delimiter is a comma.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Parameters: Tcl

Identical to C input parameters, with two exceptions:

delim	Character other than a comma to insert between fields in return text. If the argument is an empty string (“”), the default delimiter is a comma. *Argument position*: if only two parameters are passed in the function call, they are assumed to be **clientName** and **closeOpt**. If there are three parameters, they are assumed to be **clientName**, **format**, and **delim**. The only way to pass **closeOpt** with either **format** or **delim** is to pass all four parameters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

In short format, the user’s default login and group login. In long format, data from the user record (see “Sd_ListUserInfo” for the fields). Fields are delimited by commas or by the character specified in the delim argument. When all enabled users have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_GROUP_USERS

Sd_ListUserExtension

Function Prototype

int Sd_ListUserExtension(char *key, char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListUserExtension key tokenSerialOrLogin

Description

Lists the contents of one extension field, identified by key, from the record of a user specified by login or by token serial number.

Note: This function assumes that key names are unique. If more than one user extension key has the same name, a function call retrieves only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Extension field data.

Logged Events

EDITED_USER

Sd_ListUserInfo

Function Prototype

int Sd_ListUserInfo(char *tokenSerialOrLogin, char *delim, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListUserInfo tokenSerialOrLogin [-d character]

Description

Uses either the user's login or a token serial number assigned to the user to retrieve the following fields from the SDUser table in the database (page 256). Output is comma-delimited.

Field	Description
iUserNum	Number used internally as a unique identifier for the user record
chLastName	User's last name
chFirstName	User's first name
chDefaultLogin	User's default login
bCreatePIN	Whether user can create PIN (TRUE/FALSE)
bMustCreatePIN	Whether user must create PIN (TRUE/FALSE)
chDefaultShell	User's default shell
bTempUser	Whether user is a temporary user (TRUE/FALSE)
dateStart	Start date for temporary user
todStart	Start time (in seconds) for temporary user
dateEnd	End date for temporary user
todEnd	End time (in seconds) for temporary user

Note that the last four fields are significant only if **bTempUser** = TRUE.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

delim	Specifies what character should be used as a delimiter instead of the default. Select a delimiter character that will not occur in field data. The pipe symbol (|) may be a good choice.

Parameters: Tcl

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

-d *character*	Specifies that *character* should be used as a delimiter instead of the default. Select a delimiter character that will not occur in field data. The pipe symbol (|) may be a good choice.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Output consists of most of the information in the SDUser table. See the "Description" of this function for the fields. The output is comma-delimited by default.

Logged Events

LIST_ONE_USER

Sd_ListUserInfoExt

Function Prototype

intSd_ListUserInfoExt (char *tokenSerialOrLogin, int iFormat, char *chSeparator, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListUserInfoExt tokenSerialOrLogin iFormat [chSeparator]

Description

Uses either the user's login or a token serial number assigned to the user to retrieve user extension fields. This function is an extension of **Sd_ListUserInfo**. The following table describes the fields that are retrieved from the SDUser table when the **iFormat** argument is used with each listed value. Output is comma-delimited.

iFormat Value	Extension Fields Retrieved
0	All fields retrieved with **Sd_ListUserInfo**
1	All fields retrieved with **Sd_ListUserInfo**, plus **chLDAPSource**
2	All fields retrieved with **Sd_ListUserInfo**, plus remote alias.
3	All fields retrieved with **Sd_ListUserInfo**, plus profileName.
4	All fields retrieved with **Sd_ListUserInfo**, plus **chLDAPSource**, **chRmoteAlias**, and **profileName**
5	**chLDAPSource** only
6	**chRemoteAlias** only
7	**profileName** only
8	**JobName**, **JobClass**, **datesynch**, and **timesynch**
9	All fields as with 0 and 8
10	All fields as with 0, 5, 6, 7, and 8

Note: If a value greater 7 is used, 0 is assumed.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

iFormat	Specifies extension fields to be retrieved. See table under description.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Output consists of most of the information in the SDUser table; see the "Description" of this function for the fields.

Logged Events

LIST_ONE_USER

Sd_ListUsersByField

Function Prototype

int Sd_ListUsersByField(int field, int compareType, char *value, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListUsersByField field compareType value [closeOpt]

Description

Lists all users in the database who have the same values in one or more specified fields. This function returns one user each time it is called and must be called repeatedly to list more. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Fields and comparison type are specified as integers, using the values in the following tables. The value of field determines what field is compared; the value of **compareType** determines the type of comparison for some fields and supplies a value (making the value argument unnecessary) for others. The integer values in the table are associated with macro definitions in the header file, and it is recommended that you use these rather than hard-coded constants.

Users Listed	field	compareType	value
All	0	Any	Ignored
	Any	0	Ignored
By last name	1		
All beginning with		1	String
All equal to		2	String
All containing the provided string		3	String
All without a value (empty)		4	Ignored
All with any value (not empty)		5	Ignored

Users Listed	field	compareType	value
By first name	2		
All beginning with		1	String
All matching		2	String
All containing		3	String
All without a value (empty)		4	Ignored
All with any value (not empty)		5	Ignored
By default login	3		
All beginning with		1	String
All matching		2	String
All containing		3	String
All without a value (empty)		4	Ignored
All with any value (empty)		5	Ignored
By default shell	4		
All beginning with		1	String
All matching		2	String
All containing		3	String
All without a value (empty)		4	Ignored
All with any value (empty)		5	Ignored
Local or remote	5		
All local		1	Ignored
All remote		2	Ignored
All remote by realm		3	Realm name
All remote by realm		4	Realm IP address
Permanent or temporary	6		
All permanent		1	Ignored
All temporary		2	Ignored
By tokens assigned	7		
All with specified number of tokens		1	Number
All with at least one replacement token pair		2	Ignored
All with passwords		3	Ignored
All with expired tokens		4	Ignored
All with lost tokens		5	Ignored
All with token type		6	Number
All with tokens that will expire in a number of days		7	Number
All with a given seed size		8	64/128
All of a given seed size and token type		9	x|y (where “x” is seed size, and “y” is token type

Users Listed	field	compareType	value
By LDAP data	8		
All beginning with		1	String
All equal to		2	String
All containing		3	String
All without a value (empty)		4	Ignored
All with any value (not empty)		5	Ignored
By LDAP job		6	Job name
By LDAP job time		7	Time the job started (m/d/y)
By profile	9		
All with a profile		1	Ignored
All without a profile		2	Ignored
All with a named profile		3	String
By user extension	10		
All with extensions		1	Ignored
All without extensions		2	Ignored
All with extension keys		3	string
All without extension keys		4	string

Note that, if either the field or the compareType argument is 0, **Sd_ListUsersByField** ignores other parameters and lists all users.

Parameters

field Field to use for selection (see the table in “Description”.)

compareType Type of filter to use in selecting user records (see the previous table.)

value Value to use in comparing records. Not used for all comparisons (see the previous table.)

closeOpt The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Default login of a selected user. When all users who meet the selection criteria have been listed, or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

LIST_RANGE_USERS

Sd_ListValuesForAttribute

Function Prototype

intSd_ListValuesForAttribute(int attributeNum, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ListValuesForAttribute attributeNum [closeOpt]

Description

Lists the predefined values for an attribute in the RSA ACE/Server database. The function returns one value each time it is called and must be called repeatedly to list others. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Parameters

attributeNum	An attribute number.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

ValueName, ValueItself

Logged Events

LIST_VALUES

Sd_Login

Function Prototype

int Sd_Login(char *login, char *msgBuf, int bufSize);

Tcl Function Call

Sd_Login login

Description

Starts the authentication process for the user with the specified login. If called before **Sd_ApiInit**, this function determines the owner of an administrative session. If this function is called at any time after **Sd_ApiInit**, it can be used for other administrative purposes, such as performing test authentications on behalf of any user who may require administrative assistance. In either case, this function must be followed by **Sd_ContinueLogin** to avoid an "Authentication process interrupted" error.

Note: Users with administrative status in the RSA ACE/Server database should authenticate by calling the functions **Sd_AdmLogin**, and **Sd_ContinueAdmLogin**.

Parameters

login	Login of the user to authenticate.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

"Enter PASSCODE."

Logged Events

None.

Sd_MakeUserRemote

Function Prototype

int Sd_MakeUserRemote(char *defaultLogin, char *remoteAlias, char *realmName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_MakeUserRemote defaultLogin remoteAlias realmName

Description

For an existing user, this function modifies the user record to indicate that he or she should be authenticated in the specified remote realm using the specified login name (**remoteAlias**). The function requires that the user already exist in the database, that they have no administrative privileges, and no assigned tokens.

Note that this function only modifies an existing user, and does not create a new user. To create a new remote user, first call **Sd_AddUser** then **Sd_MakeUserRemote**. Also note that there is no contact with the RSA ACE/Server in the remote realm. The only changes are made locally. The login name in the remote realm (**remoteAlias**) is not verified.

Parameters

defaultLogin	The local login name for the user record to be modified. Do not include a minus sign prefix. (48 characters maximum)
remoteAlias	The user's login name in his or her home realm. (48 characters maximum)
realmName	The name of the realm in which the user is local. (48 characters maximum)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_USER

Sd_MkSoftIDExt

Function Prototype

int Sd_MkSoftIDExt(char *tokenSerialNumber, char *fileName, int key, int protect, int method, char passWord, char *msgBuf, int bufSize);

Tcl Function Call

Sd_MkSoftIDExt tokenSerialNumber fileName [key] [protect] [method] [passWord]

Description

Makes an RSA SecurID Software Token database file, which can be used as the file **setup.inf** for RSA SecurID Software Token distribution. The database file **fileName** is created with the user and token information specified by the **tokenSerialNumber** argument. This extended version of the function provides for encryption, copy protection, and password settings.

Note: Sd_MkSoftIDExt also differs from the earlier version in that it overwrites an existing file that has the same name. If you call this function iteratively, make sure that each file it creates (which contains one software token only) is named differently.

Note: To produce a 128 bit SecurID Software Token database file, call the function **Sd_DeployToken**. **Sd_MakeSoftIDExt** functions with 64 bit tokens only.

Parameters

tokenSerialNumber	Token serial number associated with the user for whom the RSA SecurID Software Token database file is to be created. Must be 12 characters. Insert leading zeros as needed to meet this requirement, for example, 000000123456.
fileName	Name of database RSA SecurID Software Token file in which to save the user and token information. If a file with this name already exists, the function call overwrites it.
key	Encryption key type: "1": RSA SecurID Software Token 2.x key "2": RSA SecurID Software Token 2.x key with password
protect	Copy protection flag: "0": Copy protection off "1": Copy protection on

method	Password usage and interpretation method: "0": No password "1": Static password :2": Default login "3": Default login appended to static password
passWord	Password (maximum 8 characters — characters in excess of this limit are truncated).

The following table illustrates the interaction of the key, method, and passWord parameter values. (Copy protection may be off or on for any of these combinations.)

Key	Method	Password	Action
1	0	Any	Creates **inf** file using RSA SecurID Software Token 2.x key without password protection.
2	0	Any	Invalid combination. Function returns error.
0,1,2	1	Empty	Invalid combination. Function returns error.
1	1	Valid	Creates **inf** file using RSA SecurID Software Token 2.x key with password protection.
2	1	Valid	Creates **inf** file using RSA SecurID Software Token 2.x key with password key protection.
1	2	Any	Creates **inf** file using RSA SecurID Software Token 2.x key with password protection. The first 8 bytes of the user default login are used as a password; the **passWord** argument is ignored.
2	2	Any	Creates **inf** file using RSA SecurID Software Token 2.x key with password key protection. The first 8 bytes of the user default login are used as a password; the **passWord** argument is ignored.
1	3	Empty	Creates **inf** file using RSA SecurID Software Token 2.x key with password protection. The first 8 bytes of the user default login are used as a password.
2	3	Empty	Creates **inf** file using RSA SecurID Software Token 2.x key with password key protection. The first 8 bytes of the user default login are used as a password.

Key	Method	Password	Action
1	3	Valid	Creates **inf** file using RSA SecurID Software Token 2.x key with password protection. The user default login is appended to the entered password. The resulting password is truncated to 8 bytes if longer.
2	3	Valid	Creates **inf** file using RSA SecurID Software Token 2.x key with password key protection. The user default login is appended to the entered password. The resulting password is truncated to 8 bytes if longer.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

SOFTID_ISSUED

Sd_MonitorHistory

Function Prototype

int Sd_MonitorHistory(char *outFile, char *closeOpt, char *msgBuf, int bufSize);

Tcl Function Call

Sd_MonitorHistory [-o outFile] [closeOpt]

Description

Returns activities that occurred since the last time the function was called during the current session; should be called periodically for an approximation of real-time monitoring of activities. This function returns one log event each time it is called and must be called repeatedly to list more. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**. (This is recommended in all circumstances.)

Note: When something close to real-time monitoring is required (and justified by the cost in system resources), you can implement it in this way: begin a listing cycle every few seconds to list events that occurred during that interval. List to the end of the cycle—that is, until **Done** is returned. (See the next note.)

Note: The first time it is called during a session, **Sd_MonitorHistory** initializes itself and returns only **Done**. Subsequent calls return events that occurred after this first call. Therefore, when you use repeated calling sequences as described in the previous note, the first sequence terminates without returning any events. Event listing begins with the second calling sequence.

Note: To prevent the generation of a large number of redundant log entries, **Sd_MonitorHistory** logs the DLOAD_ALL_LOGS event only when the **closeOpt** argument is set to **-c**. Therefore, you should always call the function with this argument at the end of each monitoring session, even after **Done** has been returned, to ensure that the event is logged once.

Parameters

outFile	This optional argument directs the output to a file rather than Return Text. The argument consists of the file name.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Parameters: Tcl

outFile	If used, this argument must have the value o *filename*, where *filename* is the name of the file to which output is to be directed instead of to Return Text.
closeOpt	See C input parameters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

For each event, output consists of the following information, constituting one log entry: event name, local date, local time, users login, affected username, Agent Host, site, and Server names, message number. When all log entries since this function was last called have been listed or the function is called with the **closeOpt** argument, the list terminator string is returned.

Logged Events

DLOAD_ALL_LOGS (logged only when **closeOpt** = **-c**; see notes under the "Description" of this function)

Sd_NewPin

Function Prototype

int Sd_NewPin(char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_NewPin tokenSerialNumber

Description

Puts the specified token in New PIN mode, where the user enters the old PIN and tokencode to authenticate and then enters the new PIN.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ACM_GEN_PIN

Sd_RemoveAdminPrivileges

Function Prototype

int Sd_RemoveAdminPriviliges(char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_RemoveAdminPrivileges tokenSerialOrLogin

Description

Sets a user's status and level as an administrator to None (that is, the user's User Type is changed to Regular User). This function is useful for resetting the status of users imported from external realms, whose administrator status is automatically imported as part of the user record. A user who is an administrator in the external realm can be made an ordinary user in the realm to which the record is imported.

Note: This function is obsolete, and is maintained for backward compatibility.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_USER

Sd_ReplaceToken

Function Prototype

int Sd_ReplaceToken(char *oldTokenSerialNumber, char *newTokenSerialNumber, int pinFlag, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ReplaceToken oldTokenSerialNumber newTokenSerialNumber [pinFlag]

Description

Replaces a user's old token with a new token, creating a replacement pair. If the **pinFlag** argument is included with a value greater than 0, the PIN associated with the old token is cleared and the token is put in New PIN mode. If the **pinFlag** argument is 0, the current PIN is copied from the old token to the new one.

Parameters

oldTokenSerialNumber	Serial number of user's current token. Must be 12 characters. Insert leading zeros as needed to meet this requirement, for example, 000000123456.
newTokenSerialNumber	Serial number of new token to assign to user. Must be 12 characters.
pinFlag	Set to a nonzero value to clear the user's current PIN and put the new token in New PIN mode. Set to 0 to have the current PIN apply to the new token.

Parameters: Tcl

Parameters other than **pinFlag** are identical to C input parameters.

pinFlag	Set to a nonzero value to clear the user's current PIN and put the new token in New PIN mode. Set to 0 or omit to have the current PIN apply to the new token.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ASSIGN_REPLACEMENT_TOKEN

Sd_RescindToken

Function Prototype

int Sd_RescindToken(char *tokenSerialNumber, int revoke, char *msgBuf, int bufSize);

Tcl Function Call

Sd_RescindToken tokenSerialNumber [revoke]

Description

Like **Sd_UnassignToken**, **Sd_RescindToken** unassigns an assigned token — but unlike that function **Sd_RescindToken** does not delete the user record when it is the user's last token that is unassigned. This function also clears all fields in the token record, resetting the record to the condition of a new token. It is therefore useful with tokens that have previously been assigned and are now unassigned.

Parameters

tokenSerialNumber	Serial number of the token to be reset. Must be 12 characters. Insert leading zeros as needed to meet this requirement, for example, 000000123456.
revoke	Revoke flag; applies to software tokens only. If the value is nonzero, clears the deployment flag, password, and copy protection flag, regardless of whether the token is assigned. Ignored for tokens of other types.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

UNASSIGNED_TOKEN if the token was assigned; ENABLED_TOKEN if the token was not assigned.

Sd_ResetLastError

Function Prototype

int Sd_ResetError

Tcl Function Call

Sd_ResetError

Description

Clears and resets the error code, error message, and function name of the last failed function that was called. There are no parameters with this function.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1 or 2) if an error condition exists.

Logged Events

None.

Sd_ResetToken

Function Prototype

int Sd_ResetToken(char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ResetToken tokenSerialNumber

Description

Resets the token status to a known state, so that (in the SDToken table in the database) the token is Enabled, and the **nextCodeStatus**, **BadTokenCodes**, and **BadPINs** fields are all 0. This should be done before assigning the token to a user. It can also be done to remedy token problems. (For a more complete resetting of an unassigned token's values, use **Sd_RescindToken**.)

Note: This function no longer affects New PIN mode, as it did in earlier versions of the API toolkit.

Parameters

tokenSerialNumber	Serial number of the token to be reset. Must be 12 characters. Insert leading zeros as needed to meet this requirement, for example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ENABLE_TOKEN, EDITED_TOKEN

Sd_Resync

Function Prototype

int Sd_Resync(char *firstTokenCode char *secondTokenCode,
char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_Resync firstTokenCode secondTokenCode tokenSerialNumber

Description

Resynchronizes a token identified by serial number with respect to the RSA ACE/Server.

Important: This function must always be called twice with sequential tokencodes. In the first call, pass **firstTokenCode** with the code currently displayed on the token; pass **secondTokenCode** as an empty or invalid (“”). In the second call, pass **firstTokenCode** with the same value as before, and **secondTokenCode** with the next sequential tokencode.

Parameters

firstTokenCode	Code currently displayed on the token you are resynchronizing
secondTokenCode	Next code in the sequence to be displayed on the token. (This argument must be passed as an empty string the first time; see “Description”).
tokenSerialNumber	Serial number of the token to be resynchronized. Must be 12 characters. Insert leading zeros as needed to meet this requirement, for example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

RESYNC_TOKEN

Sd_SetAgentHost

Function Prototype

Sd_SetAgentHost(char *agentHostNameOrAddress, int whatInFirstParameter, int whatToSet, char *agentHostName, char *agentHostAddr, char *siteName, int agentHostType, int encryptionType, int agentHostFlags, char *actingMaster, char *actingSlave, char *sharedSecret, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetAgentHost agentHostNameOrAddress whatInFirstParameter whatToSet agentHostName agentHostAddr siteName agentHostType encryptionType agentHostFlags [actingMaster actingSlave] [sharedSecret]

Description

Sets field information for an Agent Host identified by Agent Host name or IP address in the Server database. The Agent Host name, site, Agent Host type, encryption type, and several flags can be affected. See "**whatToSet**" under the parameters for a list of the fields that can be set.

Parameters:

agentHostNameOrAddress The original name of the Agent Host or IP address as in the RSA ACE/Server database.

whatInFirstParameter Determines criteria to be used for identifying an Agent Host. If "1", identifies the Agent Host by name. If "0" identifies the Agent Host by IP address.

whatToSet Specifies a field to set with new Agent Host data. The following table lists the fields that can be set, and all acceptable parameters. To set information in a particular field, pass the value listed in the Value column.

Argument and Applicable Field	Value
CHANGE_NAME chName	1
CHANGE_TYPE iClientType	2
CHANGE_ENCRYPTION iEncryptionType	4

Argument and Applicable Field	Value
CHANGE_FLAGS bSentNodeSecret bOpenToAllUsers bRemoteUserSearch	8
CHANGE_SITE iSiteNum	16
CHANGE_SECRET chSharedSecret	32
CHANGE_MASTER chMasterName	64
CHANGE_SLAVE chSlaveName	128

Note: If the bit mask used does not provide a bit to change a particular field, but the field is not empty, the argument is ignored, and the field is not changed in the database. Because these flags are defined as constants in the header file, pipe notation can be used in C function calls with these constant names: CHANGE_NAME, CHANGE_TYPE, CHANGE_ENCRYPTION, CHANGE_FLAGS, CHANGE_SITE, CHANGE_SECRET, CHANGE_MASTER, CHANGE_SLAVE. For example, to set the name, type, and site of an Agent Host, use the following argument: “whatToSet = CHANGE_NAME | CHANGE_TYPE | CHANGE_SITE

agentHostName Name of the new Agent Host.

agentHostAddr IP address of the new Agent Host.

siteName An existing site to which the Agent Host can be assigned. If the Agent Host is already assigned to a site, pass a empty string (“”) to leave it unchanged.

agentHostType 0 UNIX Agent

1 Communication server

2 Single-transaction communication server

3 Net OS Agent

4 NetSP Agent

encryptionType	0: SDI 1: DES
agentHostFlags	Flag 1 Create Node Secret? Flag 2 Open to All? Flag 3 Search Remote Realms? Flag 4 Require NameLock Note the following points about the flag settings: All four flags are defined as constants in the header file. Pipe notation can be used in C function calls with these constant names: CLIENT_SEND_NODE (= 1), CLIENT_OPEN_TOALL (= 2), CLIENT_REMOTE_SEARCH (= 4), CLIENT_NAME_LOCK (= 8). For example, to set Flags 2 and 3 to TRUE, use the following argument: “clientFlags = CLIENT_OPEN_TOALL | CLIENT_REMOTE_SEARCH;”. Flag 1 (Sent Node Secret): If you set this flag to FALSE, and it was previously TRUE, the node secret file (typically **securid**) must be deleted to enable successful authentication on the Agent Host. To set any one of these flags individually, pass its defined value. To set a combination of the flags, add each of the defined values together, and pass the total value.
actingMaster	Name of the Acting Master Server.
actingSlave	Name of the Acting Slave Server.
sharedSecret	A string of pseudorandom data known only to the RADIUS NAS Server and the RSA ACE/Server.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_CLIENT

Sd_SetClient

Function Prototype

int Sd_ SetClient(char *clientName, char *siteName, int clientType,
int encryptionType, int clientFlags, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ SetClient clientName siteName clientType encryptionType clientFlags

Description

Sets values in the record of an Agent Host identified by Agent Host name. The site, Agent type, encryption type, and several flags can be affected. See the list of parameters.

Note: This function is maintained for backward compatibility with prior versions of the RSA ACE/Server. For 5.2 administration, call the function **Sd_SetAgentHost**.

Parameters

clientName	Name of the Agent Host: either the full version of the name as in the RSA ACE/Server database (for example, **pc_client.server.com**) or the short version (for example, **pc_client**).
siteName	An existing site to which the Agent Host can be assigned. If the Agent Host is already assigned to a site, pass a empty string (“”) to leave it unchanged.
clientType	0: A UNIX Agent 1: A communication server 2: A single-transaction communications server 3: A Net OS server 4: A Net SP server
encryptionType	0: SDI 1: DES
clientFlags	Flag 1 Create Node Secret? Flag 2 Open to All? Flag 3 Search Remote Realms? Flag 4 Require NameLock

Note the following points about the flag settings:

All four flags are defined as constants in the header file. Pipe notation can be used in C function calls with these constant names: CLIENT_SEND_NODE (= 1), CLIENT_OPEN_TOALL (= 2), CLIENT_REMOTE_SEARCH (= 4), CLIENT_NAME_LOCK (= 8). For example, to set Flags 2 and 3 to TRUE, use the following argument: “clientFlags = CLIENT_OPEN_TOALL | CLIENT_REMOTE_SEARCH;”.

Flag 1 (Sent Node Secret): If you set this flag to FALSE, and it was previously TRUE, the node secret file (typically **securid**) must be deleted to enable successful authentication on the Client.

To set any one of these flags individually, pass its defined value. To set a combination of the flags, add each of the defined values together, and pass the total value.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_CLIENT

Sd_SetClientExtension

Function Prototype

int Sd_ SetClientExtension(char *key, char *data, char * clientName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ SetClientExtension key data clientName

Description

Updates data in an existing extension field (specified by key) in an Agent Host record. This new data replaces the previous contents of the field.

Note: This function assumes that key names are unique. If more than one Agent Host extension key has the same name, a function call affects only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.
clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_CLIENT

Sd_SetClientSite

Function Prototype

int Sd_ SetClientSite(char *clientName, char *siteName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ SetClientSite clientName siteName

Description

Assigns the specified Agent Host to the specified site instead of its previous site, if any. You can also use this function to unassign an Agent Host from its current site by passing the **siteName** argument as an empty string.

Parameters

clientName	The full name of the Agent Host as recorded in the RSA ACE/Server database. For example, **pc_client.server.com**. Maximum 48 characters.
siteName	Site to which to assign the client. If passed as an empty string (“”), the Agent Host is no longer assigned to any site.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_CLIENT

Sd_SetCreatePin

Function Prototype

int Sd_SetCreatePin(char *mode, char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetCreatePin mode tokenSerialOrLogin

Description

Sets the create PIN mode for the user identified by a login or a token serial number. You can use any of these three mode parameters:

USER	User must create own PIN.
SYSTEM	System must generate user's PIN.
EITHER	User may create own PIN.

Parameters

mode	One of USER, SYSTEM, or EITHER (see "Description").
tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

One of the following combinations:

- When mode = USER: USRCREATED_PIN_REQUIRED_ON and USR_CREATABLE_PIN_ON
- When mode = SYSTEM: USRCREATED_PIN_REQUIRED_OFF and USR_CREATABLE_PIN_OFF
- When mode = EITHER: USRCREATED_PIN_REQUIRED_OFF and USR_CREATABLE_PIN_ON

Sd_SetGroup

Function Prototype

int Sd_ SetGroup(char *oldName, char *newName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetGroup oldName newName

Description

Replaces the name of a specified group.

The **oldName** and **newName** parameters can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite.

Parameters

oldName	Original name of the group (optionally including a suffixed site name). Maximum 96 characters.
newName	New name to replace the original name (optionally with a site name suffixed). Maximum 96 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_GROUP

Sd_SetGroupExtension

Function Prototype

int Sd_SetGroupExtension(char *key, char *data, char *groupName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetGroupExtension key data groupName

Description

Updates data in an existing extension field (specified by key) in a group record. This new data replaces the previous contents of the field.

Note: This function assumes that key names are unique. If more than one group extension has the same name, a function call affects only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.
groupName	Name of the group. The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_GROUP

Sd_SetGroupSite

Function Prototype

int Sd_ SetGroupSite(char *groupName, char *siteName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_ SetGroupSite groupName siteName

Description

Assigns a specified group to a specified site. You can also use this function to unassign a group from its current site by passing the **siteName** argument as an empty string.

The **groupName** argument can include a site name separated from the group name by @ (or a different group/site separator established through **Sd_SetSymbols**) — for example, ourgroup@oursite.

Parameters

groupName	Group to assign (optionally with a site name suffixed).
siteName	Site to which to assign group. If passed as an empty string (“”), the group is no longer assigned to any site.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_GROUP

Sd_SetLDAPData

Function Prototype

int Sd_SetLDAPData(char *tokenSerialOrLogin, char *LDAPData,char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetLDAPData tokenSerialOrLogin LDAPData

Description

Sets LDAP data for the user specified by **tokenSerialOrLogin**. To clear LDAP data pass empty string in the “data” parameter.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user’s default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.
LDAPData	Data from an LDAP source.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_USER

Sd_SetLoggingOff

Function Prototype

int Sd_ SetLoggingOff(char *msgBuf, int bufSize);

Tcl Function Call

Sd_ SetLoggingOff

Description

Turns off the logging of administrator activity for the current session. After the call to this function, no event is logged until the next call to either **Sd_ApiEnd** or **Sd_SetLoggingOn**.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EXIT_LOG_MON

Sd_SetLoggingOn

Function Prototype

int Sd_ SetLoggingOn(char *msgBuf, int bufSize);

Tcl Function Call

Sd_ SetLoggingOn

Description

Turns on the logging of administrator activity for the current session. There are no parameters with this function.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ENTER_LOG_MON

Sd_SetPin

Function Prototype

int Sd_SetPin(char *PIN, char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetPin PIN tokenSerialNumber

Description

Sets a PIN for a specified token that is assigned to a user. The new PIN can be passed explicitly, or you can have the system create a random PIN. You can also use this function to clear the PIN and put the token in New PIN mode or to check the status of the current PIN.

Parameters

PIN Can be any of the following:

New PIN (Enter literally)

-1 Assign randomly generated PIN

"" Clear PIN, set new PIN mode

PIN Determine whether or not a PIN exists

Note: Any value for this argument other than **-1**, **""**, or **PIN** is treated as a new PIN and saved in the token record.

Note: Sd_SetPin can be used to change user passwords assigned directly through the **Sd_AssignPassword** function. Passwords of this type (whose token serial numbers have the prefix UPW) cannot be cleared, thus passing an empty string as the **PIN** argument causes an error.

tokenSerialNumber Serial number of the token for which to set the PIN. Must be 12 characters. Insert leading zeros as needed to meet this requirement, for example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

When a new PIN is passed or generated, the new PIN value is returned. When the PIN is cleared, no text is returned. For a PIN status check, TRUE is returned if a current PIN exists, FALSE if there is no current PIN.

Logged Events

For a new or cleared PIN, PIN_SET _BY_ADM. For a status check, PIN_STATUS_CHECKED.

Sd_SetPinToNTC

Function Prototype

intSd_SetPinToNTC(char *tokenSerialNumber, char *currentCode, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetPinToNTC tokenSerialNumber currentCode

Description

Sets a PIN for a specified assigned token to next tokencode. The current tokencode is retrieved and a new PIN is generated based on the next tokencode that is displayed. If a token contains hexadecimal characters, or if the system is configured to allow alphanumeric PINs, all letters are converted to digits as follows:

A=0, B=1, C=2, D=3, E=4, F=5... etc.

If a user's tokencode is longer than the allowed PIN length, the final digits of the code are ignored. For example, if the PIN can contain only 6 digits and the tokencode is **78064657**, the PIN is **780646**. If the user's RSA SecurID code is shorter than the system's minimum PIN length, the first digits of the code are repeated until the PIN is long enough. For example, if the PIN must contain 7 digits, but the RSA SecurID code is **68450**, the PIN is **6845068**.

The following applies to Software tokens and PINPads:

- If the next tokencode begins with "0", that code cannot be used as PIN. If it is, the operation fails and must be repeated.
- PIN length is dynamically set to be the length of the next successive tokencode. If the number of digits in the tokencode is less than the configured minimum PIN length, the number of digits in the tokencode is used in determining the PIN length.

Parameters

tokenSerialNumber	Serial number of the token for which to set the PIN. Must be 12 characters. Insert leading zeros as needed to meet this requirement, for example, 000000123456.
currentCode	The current code generated in the Server database for the displayed token.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Number of digits to be used from the next code.

Logged Events

NEWPIN_NEXTPRN

Sd_SetProfileName

Function Prototype

intSd_SetProfileName (char * oldprofileName, char * newprofileName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetProfileName oldprofileName newprofileName

Description

Changes the name of a profile designated by **oldprofile**.

Parameters

oldprofileName	Original name of the profile in the database.
newprofileName	New profile name. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_PROFILE

Sd_SetSite

Function Prototype

int Sd_SetSite(char *oldSite, char *newSite, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetSite oldSite newSite

Description

Changes the name of a site from the name passed by the **oldSite** argument to the name passed by the **newSite** argument.

Parameters

oldSite	Name of the site before the change.
newSite	Name of the site after the change. Maximum 48 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_SITE

Sd_SetSiteExtension

Function Prototype

int Sd_SetSiteExtension(char *key, char *data, char *siteName, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetSiteExtension key data siteName

Description

Updates data in an existing extension field (specified by key) in a site record. This new data replaces the previous contents of the field.

Note: This function assumes that key names are unique. If more than one site extension key has the same name, a function call affects only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.
siteName	Name of the site.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_SITE

Sd_SetSymbols

Function Prototype

int Sd_SetSymbols(char *setReadOpt, char *parameter, char *value, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetSymbols setReadOpt parameter value

Description

This function can retrieve the current value or set a new value for any of the following:

List terminator	The string value returned when all the items in a list have been retrieved. The default is Done.
Login prefix	The character prefixed to the value of the **tokenSerialOrLogin** argument to ensure that it is parsed as a login. The default is a minus sign (-).
Group-site separator	The character used to separate a group name from a suffixed site name in parameters that specify groups to Administration Toolkit functions, for example, ourgroup@oursite. The default is an “at” sign (@). Change it if group or site names can include this character.
Default separator	The ASCII code of the symbol used to separate an extension key from its associated data in parameters that specify lists of extension information to Administration Toolkit functions, such as **Sd_ListExtentsionsForGroup**. The default is a comma (,). Change it if you want keys and associated extension data to be separated by a character other than this character.

It is the programmer’s responsibility to be aware of conflicts with existing application code that may arise when any of these symbols is changed. If your application changes a symbol to handle a specific problem, take care that the code changes it back to the original value after the problem is dealt with, unless you are certain that no other code can be affected.

Parameters

setReadOpt	Option flag that defines the operation: **-s** Set new value **-r** Return current value "" Default (return current value) parameter Flag that specifies the symbol to be read or set: 1 List terminator 2 Login prefix 3 Group-site separator 4 Default separator "" or 0 Default (all four)
value	New value to be assigned to the symbol when **setReadOpt** is **-s**. Ignored when setReadOpt has any other value. When parameter is 0 or "" and **setReadOpt** is **-s**, any value for this argument other than an empty string ("") is ignored. An empty string sets the value to the default Done, minus sign (-), "at" character (@), or all four, depending on what is specified in the parameter argument.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

The current value of the specified parameter, or of all three if the parameter argument is "" (default). When the option is **-s**, this is the value or values set by the current function call. Sample output, reporting the default values for all four parameters:

```
Sd_SetSymbols Current value for List Terminator: Done, for
Login Symbol: -, for Site Symbol: @, for default separator: 44
```

Logged Events

None.

Sd_SetSysExtension

Function Prototype

int Sd_SetSysExtension(char *key, char *data, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetSysExtension key data

Description

Updates data in an existing extension field (specified by key) in the system record. This new data replaces the previous contents of the field.

Note: This function assumes that key names are unique. If more than one system extension key has the same name, a function call affects only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

ENTER_EDIT_SYSTEM_EXT

Sd_SetTempUser

Function Prototype

int Sd_SetTempUser(char *startDate, char startHour, char *endDate, char *endHour, char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetTempUser [[startDate startHour] endDate endHour] tokenSerialOrLogin

Description

Puts a user (identified by token serial number or login) in temporary mode. You can set both starting and ending times, or you can set only an ending time (“temporary mode lasts from now until this time”). Both date and hour must be specified for starting and ending times.

Parameters

startDate	Date when temporary mode begins.
startHour	Hour when temporary mode begins.
endDate	Date when temporary mode ends.
endHour	Hour when temporary mode ends.

Note: To take the specified user out of temporary mode, pass **startDate**, **startHour**, **endDate**, and **endHour** as empty strings (“”).

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user’s default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_USER

Sd_SetTokenExtension

Function Prototype

int Sd_SetTokenExtension(char *key, char *data, char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetTokenExtension key data tokenSerialNumber

Description

Updates data in an existing extension field (specified by key) in a token record. This new data replaces the previous contents of the field.

Note: This function assumes that key names are unique. If more than one token extension key has the same name, a function call affects only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.
tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_TOKEN

Sd_SetUser

Function Prototype

int Sd_SetUser(char *lastName, char *firstName, char *defaultLogin,
char *defaultShell, char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetUser lastName firstName defaultLogin defaultShell tokenSerialOrLogin

Description

Updates information contained in the first four input parameters into the record of a user identified by login or by token serial number.

Parameters

lastName	Last name of user. Maximum 24 characters.
firstName	First name of user. Maximum 24 characters.
defaultLogin	New default login of user. Maximum 48 characters.
defaultShell	New default shell of user. Maximum 256 characters.
tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

CHG_USER_NAME, CHG_USER_SHELL, CHG_LOGIN_ID

Sd_SetUserExtension

Function Prototype

int Sd_SetUserExtension(char* key, char *data, char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_SetUserExtension key data tokenSerialOrLogin.

Description

Updates data in an existing extension field (specified by key) in a user record. The user can be identified by login or by token serial number. The new data replaces the previous contents of the field.

Note: This function assumes that key names are unique. If more than one user extension key has the same name, a function call affects only the first. For more information, see "Known Issues" on page 18.

Parameters

key	Unique key used to identify the field. Maximum 48 characters.
data	Data contents for the field. Maximum 80 characters.
tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

EDITED_USER

Sd_SortOrder

Function Prototype

int Sd_SortOrder(Mode, char *msgBuf, int bufSize)

Tcl Function Call

Sd_SortOrder Mode

Description

Sets criteria for sorting of database tables that is performed when any listing function is called (Sd_ListToken, for example) to list the contents of these tables: SDToken, SDUser, SDClient (Agent Host), SDGroup, SDSite, SDProfile, SDSecondaryNode, SDSharedJob. By default the RSA ACE/Server database sorts and lists the contents of these tables in chronological order according the time that each database record was created.

Depending on the value that is passed with the **Mode** argument (“0”, “1”, or “2”), the function either maintains the default sorting, or sorts by alphabetical or numerical order. See the table below for details. To view the current sorting method used with a database table, use “echo” with **Mode**. To clear current sorting for a database table, use “clear” or “default” with **Mode**.

Note: When specifying a database table, use its mnemonic name. In the C example below, sorting of the database tables SDClient and SDProfile are set to sort alphabetically by the name of the Agent Host, and the name of the group.

```
Sd_SortOrder (“AgentHost=1”,”group=1” msgBuf, sizeof(bufSize));
```

Database Table	Mnemonic Name	Value	Type of Database Sorting
SDToken	Token	0 1	Default Numerical order by token serial number
SDUser	User	0 1 2	Default Alphabetical order by the user’s last name Alphabetical order by the user’s default login
SDClient	Client	0 1	Default Alphabetical order by client name
SDGroup	Group	0 1	Default Alphabetical order by group name
SDSite	Site	0 1	Default Alphabetical order by site name

Database Table	Mnemonic Name	Value	Type of Database Sorting
SDProfile	Profile	0 1	Default Alphabetical order by profile name
SDSecondaryNode	Node	0 1	Default Numerical order by node
SDSharedJob	Job	0 1	Default Numerical order by job number

Parameters

Mode	Sets sorting criteria for the database tables mentioned under “Description.” See the table under “Description” for applicable values, and the types of sorting performed by the RSA ACE/Server.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

None.

Sd_SwitchAdmin

Function Prototype

intSd_SwitchAdmin(char *defaultLogin, char *msgBuf, int bufSize);

Tcl Function Call

None.

Description

Specifies which administrator owns a session as determined by default login. If the default login for a specific administrator is not specified, ownership of the session is returned to the login that initiated the session.

Parameters

defaultLogin	The default login of an administrator.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

None.

Sd_TaskListDetails

Function Prototype

int Sd_TaskListDetails (char taskListName, char closeOpt, char msgBuf int bufSize)

Tcl Function Call

Sd_TaskListDetails taskListName CloseOpt

Description

Lists each task number for a specific task list. This function should be called repeatedly to retrieve each task number in the list. For a list of names associated with each task number, refer to the file **sd_task.txt** located in the **\ace\utils\toolkit** directory. The database search must be closed in either of the following ways:

- Call the function repeatedly until the list terminator (the default **Done** or a custom string defined through the **Sd_SetSymbols** function) is returned.
- Call the function with the **closeOpt** argument: **-c**.

Failure to close the search properly causes subsequent calls to this and other functions to fail.

Parameters

taskListName	Name of the task list in the database.
closeOpt	The value must be **-c** ("close the database search"). Use this argument only when calling the function after all the items you were seeking have been retrieved. (If **closeOpt** is never used, the call must be repeated until the list terminator string is returned.)

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

LIST_ONE_TASKLIST

Sd_Time

Function Prototype

int Sd_Time (char *msgBuf, int bufSize);

Tcl Function Call

Sd_Time

Description

Returns Coordinated Universal Time date and number of seconds since midnight. There are no parameters with this function.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

Date and number of seconds, for example, `04/29/01 , 73587`.

Logged Events

None.

Sd_UnassignProfile

Function Prototype

intSd_UnassignProfile (char *tokenSerialOrLogin, char *msgBuf, int bufSize);

Tcl Function Call

Sd_UnassignProfile tokenSerialOrLogin

Description

Unassigns a profile from a user specified by tokenSerialOrLogin.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Note: By default, the login prefix is a minus sign (-), but a different character can be assigned through the **Sd_SetSymbols** function. If the prefix is present, the argument is parsed as a login. If not, it is parsed as a token serial number.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

DELETED_PROFILE_FROM_USER

Sd_UnassignToken

Function Prototype

int Sd_UnassignToken(char *tokenSerialNumber, char *msgBuf, int bufSize);

Tcl Function Call

Sd_UnassignToken tokenSerialNumber

Description

Unassigns a token from a user. If the user has no other tokens, this function also deletes the user record from the database, provided that

- The user is not an administrator.
- The user is not enabled on any Agent Host.
- The user does not belong to any group.
- The user record has no extension fields.

Unless all of these requirements are met, the token is not unassigned nor is the user deleted.

Parameters

tokenSerialOrLogin	When this argument has the login prefix - (minus sign), it is interpreted as the user's default login. Without the prefix, it is interpreted as a token serial number assigned to the user. The serial number must have 12 characters. Insert leading zeros to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

UNASSIGN_TOKEN and, if user record is deleted, DELETED_USER

RSA ACE/Server Administration Toolkit Executables

The four toolkit executables described in the following table are maintained for backward compatibility. RSA Security recommends that you use the functions listed in the table.

Executable	Replaced by
emergency	**Sd_EmergencyAccessOn** **Sd_EmergencyAccessOff** **Sd_IsEmergencyAccess**
repltok	**Sd_AssignNewToken** **Sd_ReplaceToken**
resync	**Sd_Resync**
setpin	**Sd_SetPin**

For the sake of compatibility with existing code, the executables are distributed as part of the Toolkit. They reside in the **oldutils** subdirectory.

The executables are described here in alphabetical order. Each description has the following parts:

Invocation. Shows how the executable is invoked in a C program statement, with the types, names, and order of parameters.

Description. More detailed indication of what the executable does.

Parameters. List of input parameters with definitions.

Return Values. All executables return 0 on successful completion or 1 if an error condition exists. This is briefly noted.

Return Text. An inventory of text information returned by the executable. When an error occurs, all executables return text that identifies the type of error.

Logged Events. Entry or entries, if any, made in the event log as a result of executable operation.

emergency

Invocation

system("emergency [-s | -o | -n | -l | -d] tokenSerialNumber");

Description

To put a token in emergency access mode, invoke emergency without optional parameters or with **-n**, **-l**, and/or **-d**. By default (when executed without optional parameters), this executable changes the status of the specified token to Lost, putting it in emergency access mode, and provides two 6-digit one-time tokencodes to be used with the user's PIN for authentication. (The second tokencode is used if the RSA ACE/Server requests a second tokencode. Otherwise, it can be used to authenticate a second time. These tokencodes apply to an artificial or phantom token created when emergency is executed.) Once a token is in emergency access mode, subsequent invocations of this program with the same **tokenSerialNumber** generate different artificial tokens and associated tokencodes.

When putting a token in emergency access mode, you can specify the following parameters as optional parameters:

- Number of one-time tokencodes to generate (default is 2)

  Note: You can specify a greater number of tokencodes to be generated, provided the maximum of 50 tokencodes on file for a single user is not exceeded. If you request a number that will raise this total above 50, the request is automatically reduced to a smaller number.

- Lifetime: number of hours from the present system time until emergency access mode expires (default is 24)
- Digits: number of digits in each generated tokencode (default 6)

To remove a token from emergency access mode (resetting its status to Not Lost), invoke emergency with **-o** as the only input argument.

To determine whether or not a token is in emergency access mode, invoke emergency with **-s** as the only input argument.

Parameters

tokenSerialNumber	Serial number of token to act on (not optional).
-s	Requests the status of the token (Lost or Not Lost). Use without other parameters.
-o	Removes the token from emergency access mode by resetting its status from Lost to Not Lost. Use without other parameters.
-n *nn*	Number of tokencodes to generate; default is 2.

-l *nn*	Number of hours from the current time until emergency access mode expires.
-d *n*	Number of digits in each tokencode (4 to 8; default is 6).

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Return Text

- When token is put into emergency access mode, tokencodes to be given to the user to authenticate during the lifetime of the mode.
- When token is removed from emergency access mode, none.
- When token's emergency access status is requested, TRUE or **FALSE.**

Logged Events

TOKEN_LOST, TOKEN_FOUND

repltok

Invocation

system("repltok oldTokenSerialNumber newTokenSerialNumber");

Description

Unassigns a token (**oldTokenSerialNumber**) currently assigned to a user and assigns a different token (**newTokenSerialNumber**) to the same user. All user information is preserved and the PIN that applied to the old token now applies to the new one.

Parameters

oldTokenSerialNumber	Serial number of a token currently assigned to the user ("the old token").
newTokenSerialNumber	Serial number of an unassigned token to assign in place of the old token.

Return Values

Values returned are **OK** (value 0) upon successful completion of function or **ERROR** (value 1) if an error condition exists.

Logged Events

UNASSIGN_TOKEN, ASSIGN_TOKEN

resync

Invocation

system(resync tokenSerialNumber);

Description

Resynchronizes the specified token with respect to the RSA ACE/Server. When resync is executed, it prompts the user to enter two sequential tokencodes. When resynchronization is successfully done, the time of last login is updated and Next Token Code status is cleared.

Parameters

tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of function or ERROR (value 1) if an error condition exists.

Logged Events

RESYNC_TOKEN

setpin

Invocation

system("setpin [-p pinValue | -c | -s] tokenSerialNumber");

Description

To set a PIN, invoke **setpin** with the **p pinValue** argument. The value should be either the new PIN or -1 to have the PIN generated by the system.

To clear a PIN, invoke **setpin** with the **c** argument. The value of **p pinValue** must be 0 or an empty string (""). The token is put into new PIN mode.

To query a PIN status, invoke setpin with the **-s** argument.

Parameters

-p pinValue	Can be any of the following: *New PIN* (Enter literally) –1 Assign randomly generated PIN 0 or "" Clear PIN, set new PIN mode -c Clear PIN and put token into new PIN mode.
tokenSerialNumber	A token serial number must have 12 characters. Insert leading zeros as needed to meet this requirement. For example, 000000123456.

Return Values

Values returned are OK (value 0) upon successful completion of setting or clearing PIN; ERROR (value 1) if an error condition exists.

Return Text

- When a new PIN is passed or generated, the new PIN value is returned.
- When the PIN is cleared, no text is returned.
- For a PIN status check, TRUE is returned if a current PIN exists, **FALSE** if there is no current PIN.

Logged Events

PIN_CLEARED, PIN_STATUS_CHECKED, PIN_SET_BY_ADM

Database Schema

This section describes the Server and Log databases. Field types are indicated by one of the prefixes listed below.

Prefix	Data Type
i, tod	Integer
ch	Character
b	Boolean
d	Date

Note: RSA Security recommends that you do not modify any field marked with an asterisk (*).

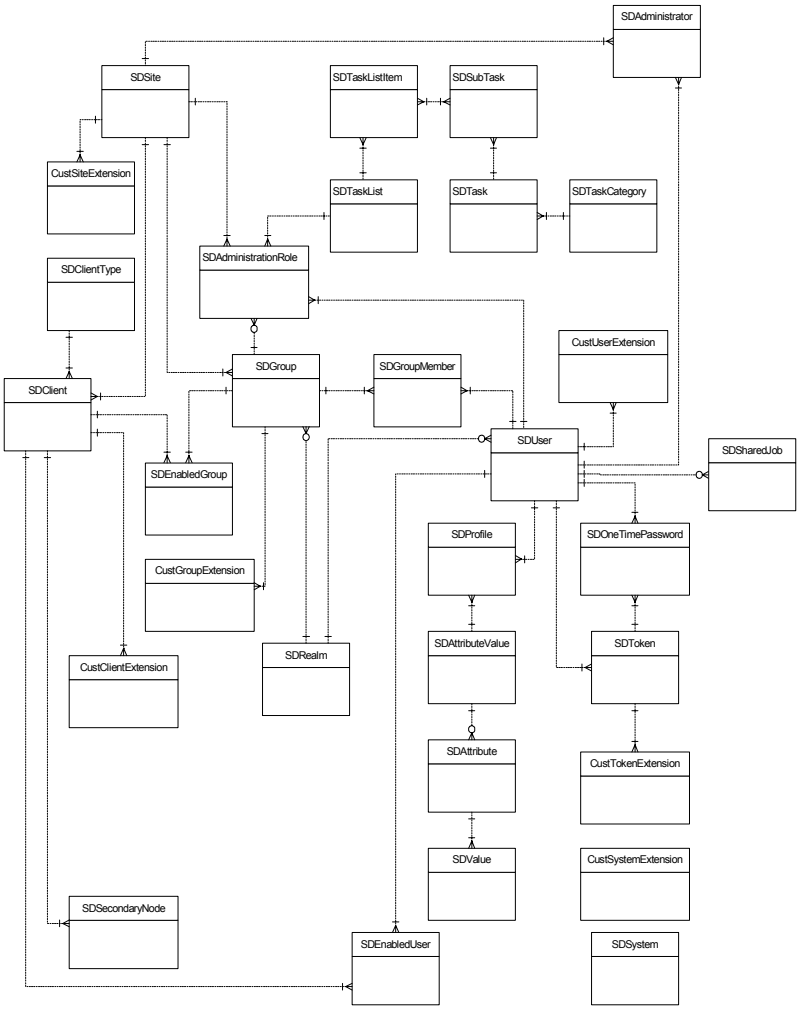
Server Tables

Server tables contain data pertaining to:

- Administrative tasks
- Users
- Groups
- Sites
- Realms
- Authenticators

The following diagram shows how Server tables are related to each other in the database schema.

Note: Tables that are *not* connected to any other table contain static information.



CustClientExtension

The customer Agent Host extension table contains Agent Host-related data defined by the application developer.

Field Name	Description
iClientNum	Agent Host to which this data is related.
iSequence	Used to order records for the same Agent Host.
chKey	Customer defined.
chData	Customer defined.

Primary unique key: iClientNum (asc), iSequence (asc)
Secondary key: chKey (asc)
Related to: SDClient By field: iClientNum Relationship: many to one

CustGroupExtension

The customer group extension table contains group-related data defined by the application developer.

Field Name	Description
iGroupNum	Group to which this data is related.
iSequence	Used to order records for the same group.
chKey	Customer defined. Limited to 48 characters.
chData	Customer defined. Limited to 80 characters.

Primary unique key: iGroupNum (asc), iSequence (asc)
Secondary key: chKey (asc)
Related to: SDGroup By field: iGroupNum Relationship: many to one

CustRealmExtension

The customer realm extension table contains customer-defined data about realms.

Field Name	Description
iSequence	Sequence number, used to order records for same realm.
iRealmNum	Realm Number.
chKey	Customer defined. Limited to 48 characters.
chData	Customer defined. Limited to 80 characters.

Primary unique key: iRealmNum(asc), iSequence(asc)
Secondary key: chKey(asc)
Related to: SDRealm By Field: iRealmNum Relationship: many to one

CustSiteExtension

The customer site extension table contains site-related data defined by the application developer.

Field Name	Description
iSiteNum	Site to which this data is related.
iSequence	Used to order records for the same site.
chKey	Customer defined. Limited to 48 characters.
chData	Customer defined. Limited to 80 characters.

Primary unique key: iSiteNum (asc), iSequence (asc)
Secondary key: chKey (asc)
Related to: SDSite By field: iSiteNum Relationship: many to one

CustSystemExtension

The customer system extension table is defined by the application developer and contains data related to the RSA ACE/Server system.

Field Name	Description
iSequence	Used to order records for the system.
chKey	Customer defined. Limited to 48 characters.
chData	Customer defined. Limited to 80 characters.

Primary unique key: iSequence (asc)
Secondary key: chKey (asc)

CustTokenExtension

The customer token extension table contains token-related data defined by the application developer.

Field Name	Description
iTokenNum	Token to which this data is related.
iSequence	Used to order records for the same token.
chKey	Customer defined. Limited to 48 characters.
chData	Customer defined. Limited to 80 characters.

Primary unique key: iTokenNum (asc), iSequence (asc)
Secondary key: chKey (asc)
Related to: SDToken By field: iTokenNum Relationship: many to one

CustUserExtension

The customer user extension table contains user-related data defined by the application developer.

Field Name	Description
iUserNum	User to whom this data is related.
iSequence	Used to order records for the same user.
chKey	Customer defined. Limited to 48 characters.
chData	Customer defined. Limited to 80 characters.

Primary unique key: iUserNum (asc), iSequence (asc)
Secondary key: chKey (asc)
Related to: SDUser By field: iUserNum Relationship: many to one

SDAdministrativeRole

This table contains information about a user's administrative role.

Field Name	Description
*iUserName	User to whom this role is associated.
iListNum	Task list related to this role.
iRealmNum	Realm number for this role. Default = 0 indicating the local realm.
iSiteNum	Site number if this user is scoped to a site; 0 if the user is not limited to a specific site.
iGroupNum	Group number if this user is scoped to a group; 0 if the user is not limited to a specific group.
bFilterData	If TRUE, the user can only list users, groups, and Agent Hosts based on the values in iSiteNum and iGroupNum.

Primary unique key: iUserNum (asc)
Related to: SDUser By field: iUserNum Relationship: many to one
Related to: SDTaskList By field: iListNum Relationship: many to one
Related to: SDSite By field: iSiteNum Relationship: many to one
Related to: SDGroup By field: iGroupNum Relationship: many to one
Related to: SDRealm By field: iRealmNum Relationship: many to one.

SDAdministrator

SDAdministrator contains RSA ACE/Server administrators.

Field Name	Description
iUserNum	User number of the administrator.
iAdmType	Type of administrator: Always = 0.
iAdmDomain	Always = 0.

Primary unique key: iUserNum (asc)
Related to: SDUser By field: iUserNum Relationship: many to one

SDAttribute

The attribute table contains information about the available attributes.

Field Name	Description
iRadiusAttributeID	RADIUS encoding of the attribute name as an integer.
chAttributeName	Name of attribute. Limited to 32 characters.
iDataType	Integer encoding of the data type. 0: String 1: Integer 2: IP address 3: Date
bMultipleInstance	If TRUE, multiple instances of this attribute are allowed. Default = FALSE.
bUserConfigurable	If TRUE, a user can assign a value to this attribute. Default = FALSE.

Primary unique key: iRadiusAttribute (asc), chAttributeName (asc)
Secondary unique key: chAttributeName(asc)
Related to: SDAttributeValue By field: iRadiusAttributeID Relationship: many to one
Related to: SDValue By field: iRadiusAttributeID Relationship: one to many

SDAttributeValue

The attribute value table contains information about available attributes.

Field Name	Description
iRadiusAttributeID	RADIUS encoding of the attribute name as an integer.
chValue	Stored profile value as set by the user. Limited to 253 characters.
iProfileNum	Index to corresponding profile in SDProfile table. If 0, none.
iSequenceNum	Order to apply multiply-defined attributes.

Primary unique key: iProfileNum (asc), iRadiusAttribute (asc), iSequenceNum (asc)
Related to: SDAttribute By field: iRadiusAttributeID Relationship: one to many
Related to: SDProfile By field: iProfileNum Relationship: many to one

SDClient

The SDClient table holds data associated with an Agent Host.

Field Name	Description
*iClientNum	Allows the Agent Host to be renamed without changing other references to it.
chName	Host name of the Agent Host. Limited to 48 characters.
bSentNodeSecret	If TRUE, the node secret has been sent to the Agent Host.
iSiteNum	Number of the site to which this Agent Host belongs. 0 if the Agent Host does not belong to any site.
iClientType	Type of Agent 1: UNIX Agent 2: Communication Server 3: Single Transaction Server 4: Net OS Agent 5: Net SP Agent
chNetAddresss	IP address of the Agent Host. Limited to 48 characters.

Field Name	Description
iProtocol	Network protocol for which the network address is used: 1: Novell 2: Internet 3: SNA
*chSecurityBlock	Encrypted security information used by the RSA ACE/Server software. Limited to 256 characters.
*dateLastModified	Date the Agent Host was last modified.
*todLastModified	Time of day in seconds the Agent Host was last modified.
iEncryptionType	Type of encryption used for Agent Host-Server communications with this Agent Host. 0: SDI 1: DES
bRemoteUserSearch	If TRUE, Agent Host is allowed to search remote realms for users. Default = FALSE.
bOpenToAllUsers	If TRUE, all locally known users are allowed on the Agent Host. Default = FALSE.
dateAutoCreated	Date Agent Host was automatically created.
todAutoCreated	Time of day Agent Host was automatically created.
iUCTAutoUpdated	Time of last automatic update.
chMasterName	The name of an Acting Master. Limited to 48 characters.
chMasterNetAddress	The IP address of an Acting Master. Limited to 48 characters.
chSlaveName	The name of an Acting Slave. Limited to 48 characters.
chSlaveNetAddress	The IP address of an Acting Slave. Limited to 48 characters.
bMasterAssigned	If TRUE, the Agent Host has been assigned to an Acting Master. Default = FALSE.
bSlaveAssigned	If TRUE, the Agent Host is assigned to an Acting slave. Default = FALSE.

Field Name	Description
iProtocol	Network protocol for which the network address is used: 1: Novell 2: Internet 3: SNA
*chSecurityBlock	Encrypted security information used by the RSA ACE/Server software. Limited to 256 characters.
*dateLastModified	Date the Agent Host was last modified.
*todLastModified	Time of day in seconds the Agent Host was last modified.
iEncryptionType	Type of encryption used for Agent Host-Server communications with this Agent Host. 0: SDI 1: DES
bRemoteUserSearch	If TRUE, Agent Host is allowed to search remote realms for users. Default = FALSE.
bOpenToAllUsers	If TRUE, all locally known users are allowed on the Agent Host. Default = FALSE.
dateAutoCreated	Date Agent Host was automatically created.
todAutoCreated	Time of day Agent Host was automatically created.
iUCTAutoUpdated	Time of last automatic update.
chMasterName	The name of an Acting Master. Limited to 48 characters.
chMasterNetAddress	The IP address of an Acting Master. Limited to 48 characters.
chSlaveName	The name of an Acting Slave. Limited to 48 characters.
chSlaveNetAddress	The IP address of an Acting Slave. Limited to 48 characters.
bMasterAssigned	If TRUE, the Agent Host has been assigned to an Acting Master. Default = FALSE.
bSlaveAssigned	If TRUE, the Agent Host is assigned to an Acting slave. Default = FALSE.

Field Name	Description
bnamelockRequired	If TRUE, the Agent Host uses two-step authentication. The Server first locks the user record for the user attempting to authenticate, then accepts the PASSCODE from the Agent Host. Only version 5.0 RSA ACE/Agent software is capable of using two-step authentication. Default = FALSE

Primary unique key: iClientNum (asc)
Secondary unique key: chName (asc), iProtocol (asc)
Secondary unique key: chNetAddress(asc), iProtocol (asc)
Related to: SDSite By field: iSiteNum Relationship: many to zero or one
Related to: SDSecondaryNode By field: iClientNum Relationship: one to zero or many
Related to: SDEnabledGroup By field: iClientNum Relationship: one to zero or many
Related to: SDEnabledUser By field: iClientNum Relationship: one to zero or many
Related to: SDClientType By field: iClientType Relationship: many to one
Related to: CustClientExtension By field: iClientNum Relationship: one to zero or many

SDClientType

The SDClient table holds data associated with an Agent type.

Field Name	Description
iClientType	Type of Agent Host.
chName	Agent type name. Limited to 48 characters.
bNextTCodeAllowed	If TRUE, Agent type supports Next Tokencode and New PIN modes. Default = FALSE.

Primary unique key: iClientType (asc)
Related to: SDClient By field: iClientType Relationship: one to zero or many

SDEnabledGroup

SDEnabledGroup is a cross-reference table for the “many-to-many” relationship between SDGroup and SDClient.

Field Name	Description
iGroupNum	Group number.
iClientNum	Agent Host on which the group is activated.

Primary unique key: iGroupNum (asc), iClientNum (asc)
Secondary key: iClientNum (asc)
Related to: SDGroup By field: iGroupNum Relationship: many to one
Related to: SDClient By field: iClientNum Relationship: many to one

SDEnabledUser

SDEnabledUser is a cross-reference table for the “many-to-many” relationship between SDUser and SDClient.

A record is inserted to activate a user directly on an Agent Host, and a record is deleted to deactivate a user from an Agent Host.

Field Name	Description
iUserNum	User number.
iClientNum	Agent Host on which the user is directly activated.
chLogin	Login ID of this user on this Agent Host. Limited to 48 characters.
chShell	Shell to use when authenticated for this Agent Host. Limited to 256 characters.

Primary unique key: iClientNum (asc), chLogin (asc)
Secondary key: iUserNum (asc)
Related to: SDUser By field: iUserNum Relationship: many to one
Related to: SDClient By field: iClientNum Relationship: many to one

SDGroup

SDGroup allows you to group users together for administration operations, such as activating a group of users on an Agent Host.

Field Name	Description
*iGroupNum	Allows the group to be renamed without changing other references to it.
chName	Group name. Limited to 48 characters.
iSiteNum	Site number of site to which this group belongs; 0 if it does not belong to a site.
chAccessTimeArray	Array of access times. Limited to 168 characters.

Primary unique key: iGroupNum (asc)
Secondary unique key: iSiteNum (asc), chName (asc)
Related to: SDSite By field: iSiteNum Relationship: many to zero or one
Related to: SDGroupMember By field: iGroupNum Relationship: one to zero or many
Related to: SDEnabledGroup By field: iGroupNum Relationship: one to zero or many
Related to: CustGroupExtension By field: iGroupNum Relationship: one to zero or many
Related to: SDRealmEnabledGroup By field: iGroupNum Relationship: one to zero or many

SDGroupMember

SDGroupMember is a cross-reference table for the “many-to-many” relationship between SDUser and SDGroup. A record is inserted to add a user to a group, and a record is deleted to remove a user from a group.

Field Name	Description
iGroupNum	Group of which the user is a member.
iUserNum	User number.
chLogin	Login ID of this user when logged in under this group’s authority. If a group login ID is not defined, the user’s regular default login is used. Limited to 48 characters.
chShell	The shell to use when logged in under this group’s authority. Limited to 256 characters.

Primary unique key: iGroupNum (asc), chLogin (asc)
Secondary key: iUserNum (asc)
Related to: SDGroup By field: iGroupNum Relationship: many to one
Related to: SDUser By field: iUserNum Relationship: many to one

SDOneTimePassword

The one-time-password table contains information about emergency access passwords.

Field Name	Description
iUserNum	User number of the user who owns the one time password.
iTokenNum	Token number associated with the one time password. If 0, one time password associated only with user.
chOTPasswordsSB	Encrypted password information used by the RSA ACE/Server software. Limited to 256 characters.
datePWExpires	Date password expires.
todPWExpires	Time of day password expires.
*iPasswordNum	Sequence in which passwords are presented to user.
iPasswordSetNum	Password set sequence number.

Primary unique key: iPasswordNum(asc)
Secondary key: iUserNum(asc), iTokenNum(asc), iPasswordNum(asc)
Related to: SDUser By Field: iUserNum Relationship: many to one
Related to: SDToken By Field: iTokenNum Relationship: many to one

SDProfile

The profile table contains information about profiles of RADIUS users.

Field Name	Description
iProfileNum	Profile index number of corresponding profile.
chName	Profile name. Limited to 56 characters.

Primary unique key: iProfileNum (asc)
Secondary key: chName (asc)
Related to: SDAttributeValue By field: iProfileNum Relationship: one to many
Related to: SDUser By field: iProfileNum Relationship: one to many

SDRealm

The realm table contains information about remote realms used in cross-realm authentication.

Field Name	Description
*iRealmNum	Realm number.
chComment	Comment. Limited to 80 characters.
chMasterName	The name of the Preferred Server in the remote realm. Limited to 48 characters.
chMasterAddr	The IP address of the Preferred Server in the remote realm. Limited to 48 characters.
chSlaveName	The name of the Failover Server in the remote realm. Limited to 48 characters.
chSlaveAddr	The IP address of the Failover Server in the remote realm. Limited to 48 characters.
iProtocol	Protocol number. This field is for internal use only.
iPeerProtocol	Protocol of peer server. This field is for internal use only.
iTrustStatust	Is realm secret established? 0: Not established 1: Established manually 2: Established automatically
chRealmSecBlk	Realm table security block. Limited to 256 characters.
chFromMasterName	The name of the Preferred Server in the local realm. Limited to 50 characters.
chFromMasterAddr	The IP address of the Preferred Server in the local realm. Limited to 16 characters.
chFromSlaveName	The name of the Failover Server in the local realm. Limited to 50 characters.

Field Name	Description
chFromSlaveAdr	The IP address of the Failover Server in the local realm. Limited to 16 characters.
chPrimaryName	The name of the Primary Server in the remote realm. Limited to 48 characters.
chPrimaryAddr	The IP address of the Primary Server in the remote realm. Limited to 16 characters.

Primary unique key: iRealmNum(asc)
Secondary unique key: chMasterAddr(asc), iProtocol(asc)
Secondary unique key: chMasterName(asc), iProtocol(asc)
Secondary unique key: chPrimaryAddr (asc), iProtocol(asc)
Secondary unique key: chPrimaryName(asc), iProtocol(asc)
Secondary key: chSlaveName(asc), iProtocol(asc)
Secondary key: chSlaveAddr(asc), iProtocol(asc)
Related to: SDRealmEnabledGroup By field: iRealmNum Relationship: one to many
Related to: SDRealmEnabledUser By field: iRealmNum Relationship: one to many
Related to: CustRealmExtension By field: iRealmNum Relationship: one to many

SDRealmEnabledGroup

The realm-enabled group table contains groups enabled on a realm.

Field Name	Description
iGroupNum	Group Number.
iRealmNum	Realm Number of realm on which group is enabled.

Primary unique key: iRealmNum(asc), iGroupNum(asc)
Secondary key: iRealmNum(asc)
Related to: SDGroup By Field: iGroupNum Relationship: many to one
Related to: SDRealm By Field: iRealmNum Relationship: many to one

SDRealmEnabledUser

The realm-enabled user table contains users enabled on a realm.

Field Name	Description
iUserNum	User Number.
iRealmNum	Realm Number of the realm in which the user is directly enabled.

Primary unique key: iRealmNum(asc), iUserNum(asc)
Secondary key: iUserNum(asc)
Related to: SDUser By Field: iUserNum Relationship: many to one
Related to: SDRealm By Field: iRealmNum Relationship: many to one

SDReplica

The Replica table contains information about the Replica Server.

Field Name	Description
iReplicaNum	Server number.
chFullyQualifedName	The fully qualified domain name of the Replica Server. Limited to 48 characters.
chNetAddress	The IP address of the Replica database. Limited to 48 characters.
iPortNumber	The port number used by this Replica to communicate with the Primary.
iStartupDelayInterval	The number of seconds after the Primary Server starts up that the replication service starts.
iReplicationInterval	How often database changes are replicated, in seconds.
benabled	If TRUE, allows deltas to be generated. Default = FALSE.
chReplicaServiceName	The service name that the Replica and the Primary use for communication.
bdbconnected	If TRUE, the Replica has established initial contact with the Primary when the Replica package is installed. Default = FALSE.
bDBPushNeeded	If TRUE, allows for dbpush at next replication pass.
iDBRepSequenceNum	The Replica Sequence Number is used to ensure that the correct database is being used by the Primary and the Replica.
chAlias1	The first alias IP address of a Replica Server.
chAlias2	The second alias IP address of a Replica Server.
chAlias3	The third alias IP address of a Replica Server.
bIsPrimary	If TRUE, one Replica record is also the Primary.

Primary unique key: iRealmNum(asc), iUserNum(asc), iReplicaNum(asc), iPortNumber, chReplicaServiceName
Secondary key: iUserNum(asc)
Related to: SDUser By Field: iUserNum Relationship: many to one
Related to: SDRealm By Field: iRealmNum Relationship: many to one

SDSchedJob

This table lists information about the license verification job, and all LDAP sychronization jobs.

Field Name	Description
iJobNum	A numeric identifier.
chJobName	The name of the job. Limited to 48 characters.
chJobClass	The job classification. Limited to 24 characters.
bEnabled	If TRUE, the job is enabled in the database. Default = TRUE.
bRunOnce	Indicates that the job is configured to run only once. Default = FALSE.
todInitialStart	The time of day (calculated in seconds) that the initial run of the job occurred.
dateInitialStart	The date that the initial run of the job occurred.
iRunIntervalSecs	The number of seconds between each job.
chExecutable	The executable used to run LDAP synchronization jobs and license check jobs. **sdldapsync** for synchronization jobs, and **rsalicutil** for license checks.
chDirectory	The working directory.
chParamData	The parameters used by the executable at the time that the job is run.
todStarted	Date the last run was started.
dateStarted	The most recent date on which the job was started.
todCompleted	Time of day the last run was completed.
dateCompleted	The date that the last run was completed.
iExitStatus	An integer representing the job's status. The status is described in the **chSummary** field.
chSummary	The summary of the last time the job was run.

Primary unique key: iJobNum
Secondary key: cJobName, chJobClass

SDSecondaryNode

SDSecondaryNode allows an Agent Host to have more than one network address.

Field Name	Description
iClientNum	Agent Host number.
iProtocol	Network protocol of the network address: 1 Novell 2 Internet 3 SNA
chName	Secondary node name. Limited to 48 characters.
chNetAddress	Secondary node network address. Limited to 48 characters.

Primary unique key: chNetAddress (asc), iProtocol(asc)
Secondary key: iClientNum (asc)
Secondary unique key: chName (asc), iProtocol (asc)
Related to: SDClient By field: iClientNum Relationship: many to one

SDSite

Field Name	Description
*iSiteNum	Allows the site to be renamed without changing other references to it.
chName	Name of the site. Limited to 48 characters.

Primary unique key: iSiteNum (asc)
Secondary unique key: chName (asc)
Related to: SDClient By field: iSiteNum Relationship: zero or one to zero or many
Related to: SDGroup By field: iSiteNum Relationship: zero or one to zero or many
Related to: SDAdministrativeRole By field: iSiteNum Relationship: one to zero or many
Related to: CustSiteExtension By field: iSiteNum Relationship: one to zero or many

SDSubTask

This table contains information about tasks that should be enabled when another task is provided. For example, if an administrator can Edit User, then he or she should be provided with appropriate subtasks, such as Assign Token and Change Administrative Role.

Field Name	Description
*iTaskNum	The task number of the task.
*iEnabledSubtaskNum	The task number of the subtask that should be enabled if the task associated with iTaskNum is provided.

Primary unique key: iTaskNum (asc)
Primary unique key: iEnabledSubtaskNum (asc)
Primary unique key: iTaskNum (asc), iEnabledSubtaskNum (asc)
Related to: SDTask By field: iTaskNum Relationship: many to one
Related to: SDTaskListItem By field: iTaskNum Relationship: many to many

SDSystem

This table contains RSA ACE/Server system status fields. You cannot modify the settings for any of these fields.

Field Name	Description
*chDBVersion	Version of this database. Limited to 8 characters.
*iServerClkCorr	The number of seconds the system clock varies from UTC. This number is added to the Server clock. Adjust this number if you cannot calibrate the Server clock to tokens or other Servers.
*iMinPINLen	Minimum length of a PIN.
*iMaxPINLen	Maximum length of a PIN.
*bUserCreatedPINAllowed	If TRUE, users can create their own PINs.
*bUserCreatedPINRequired	If TRUE, users must create their own PINs.
*bAlphanumericPINAllowed	If TRUE, users can create alphanumeric PINs.
*chSecurityBlock	Encrypted security information used by the RSA ACE/Server software. Limited to 256 characters.
*dateLastModified	Date the system record was last modified.

Field Name	Description
*todLastModified	Time of day in seconds the system record was last modified.
*dateServerStart	Date Server was last started.
*todServerStart	Time of day in seconds the Server was last started.
bSlaveNeedsPush	Does Slave Server need a push? Default = FALSE
bCriteriaChgFlag	Has system log criteria changed since the last time it was checked? Default = FALSE
bRemoteUserSearch	If TRUE, this Server is allowed to send cross-realm requests.
bAllUserRemote	If TRUE, all users are allowed to authenticate from trusted realms.
iUserAgeLimit	Maximum age in days for remote user to exist without active log in.
bAllowAutoClientRegistration	If TRUE, automatic Agent Host registration is possible on this Server. Default = FALSE.
iValidPeriod	Valid period of user password.
bAllowAutomaticRealmManagement	If TRUE, Automatic Realm Management is allowed.
bSDAuthorization	Not currently used.
bExtAuthorization	If TRUE, external authorization is allowed.
bExtAuthorizationHomeData	If TRUE, external remote authorization is allowed.
bAllowOtherRealmAdm	If TRUE, administrators of other realms are allowed access to this database when they present the proper credentials. If FALSE, no external administrators are allowed.
iAdmCertLifetime	Certificate Lifetime for Administration.
iCertsAllowed	Accept RSA Security and Custom Certificates or only Custom Certificates. This field is for internal use only.
bAllowRemoteAdm	If TRUE, remote administration is allowed.

Field Name	Description
bAllowSecurID	If TRUE, administrator may authenticate with RSA SecurID. Default = TRUE.
bAllowTBSmartCard	If TRUE, administrator may authenticate with smart cards. Default = FALSE.
bAllowSoftID	If TRUE, administrator may authenticate with an RSA SecurID Software Token. Default = FALSE.
bAllowLTPassword	If TRUE, administrator may authenticate with a lost token password. Default = FALSE.
bAllowUserPassword	If TRUE, administrator may authenticate with a user password. Default = FALSE.
bAutoDeleteReplacedToken	If TRUE, automatically deletes a token record the first time its replacement is used.
bDBPushEnabled	If TRUE, enables a database to be pushed to a disabled Replica once the Replica has been recovered. Default = TRUE.
iDBRepNextSwquenceNum	If TRUE, the time of the last login is stored in the token records.
bAllowHWMark	If TRUE, the time of last login is stored in token records.
todServerListLastUpdate	The most recent time the Server list was updated.
dateServerListLastUpdate	The most recent date on which the Server list was updated.
dateNominated	The date that a Replica is nominated to Primary.
todNominated	The total number of seconds (calculated from 12:00a.m.) since the most recent nomination of a Replica to a Primary.

Primary unique key: chDBVersion (asc)

SDTask

This table contains information about tasks that can be performed in the administrative interface.

Field Name	Description
chTaskName	Name of the task. Limited to 64 characters.
iTaskNum	Task number.
iTaskCategoryNum	Task category number.

Primary unique key: iTaskNum (asc)
Secondary unique key: chTaskName (asc)
Related to: SDTaskListItem By field: iTaskNum Relationship: one to many
Related to: SDSubTask By field: iTaskNum Relationship: one to many
Related to: SDTaskCategory By field: iTaskCategoryNum Relationship: many to one

SDTaskCategory

This table contains information about task categories and subcategories. Task categories are primarily used by the user interface.

Field Name	Description
*chCategoryName	Name of the category. Limited to 32 characters.
*iCategoryNum	Number of the category.
*iParentCategoryNum	Upper-level category number if this category is a subcategory.

Primary unique key: iCategoryNum (asc)
Secondary unique key: chCategoryName (asc)
Related to: SDTask By field: iCategoryNum Relationship: one to many
Related to: SDTaskCategory By field: iParentCategoryNum Relationship: many to one

SDTaskList

This table contains information about each available task list. User defined task lists are assigned:

- A number beginning with 101.
- A default value of FALSE in the "bSDITaskList" field.

Field Name	Description
chListName	Name of the task list. Limited to 64 characters.
iListNum	Number of the task list.
bSDITaskList	If TRUE, the task list is built in to the database and cannot be altered or deleted.

Primary unique key: iListNum (asc)
Related to: SDRole By field: iListNum Relationship: one to many
Related to: SDTaskListItem By field: iListNum Relationship: one to many

SDTaskListItem

This table contains information about the tasks available in a specific task list.

Field Name	Description
iListNum	Number of the task list.
iTaskNum	Number of the task.

Primary unique key: iListNum (asc)
Primary unique key: iTaskNum (asc)
Related to: SDTaskList By field: iListNum Relationship: many to one
Related to: SDTask By field: iTaskNum Relationship: many to one
Related to: SDSubTask By field: iTaskNum Relationship: many to many

SDToken

Field Name	Description
*iTokenNum	Saves time when comparing, since the serial number is a character field; also saves space in indexes and other tables.
*chSerialNum	Serial number with leading zeros. Limited to 12 characters.
*chSecurityBlock	Encrypted security information used by the RSA ACE/Server software. Limited to 256 characters.
*iNumDigits	Number of digits in the token display.
*iInterval	Display change interval in seconds.
*dateBirth	Date the token was activated.
*todBirth	Time of day in seconds the token was activated.
*dateDeath	Date the token will shut down.
*todDeath	Time of day in seconds the token will shut down.
*dateLastLogin	The date of the last login.
*todLastLogin	Time of day in seconds of the last login.
*iType	Token type: 0: RSA SecurID Standard Card 1: RSA SecurID PINPad Card 2: RSA SecurID Key Fob 3: reserved for future use 4: RSA SecurID Software Token 5: reserved for future us| 6: RSA SecurID Modem 7: reserved for future use
*bHex	If TRUE, the token has a hexadecimal display.
bEnabled	If TRUE, the token has been enabled.
*bNewPINMode (can be set but not cleared)	If TRUE, the user is in New PIN mode for this token.
iSeedSizeType	Stores the algorithm type. 64: SID, 128: AES, 1: Password

Field Name	Description
bkeypad	If TRUE, the token has a keypad.
bLocalPin	If TRUE, a PIN is stored locally on a user's computer. Default = FALSE.
iFormFactor	Bitmask for form factors.
iTokenVersion	The version of the token algorithm
*iNextTCodeStatus	Next tokencode status: 0: Not in next tokencode mode. 1: Waiting for one more good tokencode, no PIN. 2: Waiting for two good tokencodes, no PIN.
*iBadTokenCodes	Count of bad tokencodes.
*iBadPINs	Count of bad PINs.
*datePIN	Date the PIN was modified.
*todPIN	Time of day in seconds the PIN was modified.
iPinType	Code for a PIN type.
*dateEnabled	Date the token was enabled or disabled.
*todEnabled	Time of day the token was enabled or disabled.
iUserNum	User to whom the token has been assigned.
*dateCountsLastModified	Date token counts last modified. This field applies to fixed passwords only.
dateTokenAssigment	Date on which the token was assigned. This field applies to fixed passwords only.
*todCountsLastModified	Time of day the token counts were last modified.
todTokenAssignment	Time of day that the token is assigned.
bLost	Defines the token as lost. Default = NOT LOST.
dateLostStatusExpires	The date on which a lost token's fixed password expires.

Field Name	Description
todLostStatusExpires	Time of day token loses lost status.
iLostAuthMethod	The method by which a lost token authenticates. 0: The token is NOT lost. 1: The token is assigned a fixed password. 2: The token is assigned a one time password.
iReplacingTokenNum	Used to store the token number of the replaced token.
iReplacedByTokenNum	Used to store the token number of the replacement token.
bSoftID_CopyProtected	If TRUE, copy protection is on. If FALSE, copy protection is off.
bSoftID_Deployed	If TRUE, the software token has been issued. If FALSE, the software token has not been issued.
iSoftID_Count	The number of times the software token has been issued (maximum 5).
iSoftID_Type	The type of software token. 0: No type defined 1: PC 2: Palm computing platform 3: Other
bFirstLogin	Used for tokens that do not have PINs. If TRUE, the assigned user has not yet used the token for a successful login.

Primary unique key: iTokenNum (asc)
Secondary unique key: chSerialNum (asc)
Secondary key: iTokenNum (asc), iUserNum (asc), iType (asc), iSeedSizeType (asc)
Secondary key: iUserNum (asc)
Secondary key: dateDeath (asc)
Related to: CustTokenExtenstion By field: iTokenNum Relationship: one to zero or many
Related to: SDUser By field: iUserNum Relationship: many to one
Related to: SDOneTimePassword By field: iTokenNum Relationship: one to zero or many

SDUser

Field Name	Description
*iUserNum	Allows user to be renamed without changing other references to it.
chLastName	The user's last name. Limited to 24 characters.
chFirstName	The user's first name. Limited to 24 characters.
chDefaultLogin	Default login ID. Limited to 48 characters.
bCreatePIN	If TRUE, users can create their own PINs.
bMustCreatePIN	If TRUE, users must create their own PINs.
chDefaultShell	Default shell. Limited to 256 characters.
bTempUser	If TRUE, then the user is a temporary user and the fields dateStart, todStart, dateEnd, and todEnd apply to this user.
dateStart	If temp user, then start date for this user.
todStart	If temp user, then start time of day in seconds for this user.
dateEnd	If temp user, then expiration date for this user.
todEnd	If temp user, then expiration time of day in seconds for this user.
iRealmNum	Home realm number, 0 for local users.
iLocation	The user's location. 0=local, 1=remote, 2=unknown remote
chAccessTimeArray	Array of access times. Limited to 168 characters.
dateAutoCreated	The date on which the record was automatically created.
todAutoCreated	Time of day the record was automatically created for remote users.
chProfileName	The name of the profile assigned to the user.

Field Name	Description
iProfileNum	The profile number of the profile assigned to the user (from SDProfile). 0 = no profile.
dateLastRemoteContact	Date of last contact with user's home realm.
todLastRemoteContact	Time of day of last contact with user's home realm.
chRemoteAlias	User's login name in the home realm. Limited to 48 characters.
*iAuthReservation	The software that processes authentication sets this value to a non-zero integer when an authentication has been started.
chLDAPSource	User's location as specified in an LDAP directory. Limited to 1024 characters.
iJobNum	The number of the LDAP synchronization job used to update this user in the Server database.
dateLastLDAPSynch	The date of the most recent synchronization with the LDAP Server.
todLastLDAPSynch	The time of the most recent sychronization with the LDAP Server.

Primary unique key: iUserNum (asc)
Secondary unique key: chDefaultLogin (asc)
Secondary key: chLastName (asc), chFirstName (asc)
Secondary key: iUserNum (asc), chLastName (asc), iRealNum (asc)
Related to: SDToken By field: iUserNum Relationship: one to zero or many
Related to: SDGroupMember By field: iUserNum Relationship: one to zero or many
Related to: SDEnabledUser By field: iUserNum Relationship: one to zero or many
Related to: CustUserExtension By field: iUserNum Relationship: one to zero or many
Related to: SDProfile By field: iProfileNum Relationship: many to one
Related to: SDOneTimePassword By field: iUserNum Relationship: one to zero or many
Related to: SDRealmEnabledUser By field: iUserNum Relationship: one to zero or many

SDUserScope

The SDUserScope table is maintained with triggers and should *never* be modified using 4GL or ESQL. This table tracks a user's membership in groups, and a user's assignment to sites by their membership in groups. The information in this table is modified when the Last Name, First Name or Login of a user is modified, when a user is added to or deleted from a group, and when a group is assigned to or unassigned from a site.

A record is created for each group in which a user is a member. An additional record is created for each assignment of the group to a site.

SDValue

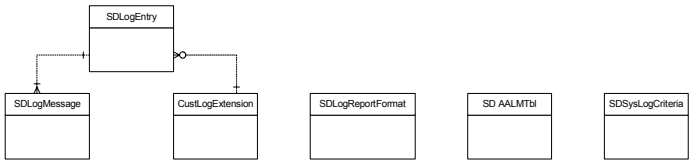
The value table contains the RADIUS dictionary.

Field Name	Description
chValue	Value name. Limited to 32 characters.
iRadiusAttributeID	RADIUS encoding of the attribute name as an integer.
iRadiusValueID	RADIUS encoding of the value name as an integer.

Primary unique key: iRadiusAttribute (asc), chValue (asc)
Related to: SDAttribute By field: iRadiusAttributeID Relationship: many to one

Log Tables

Log tables contain data about events logged in RSA ACE/Server log database.



CustLogExtension

The customer log extension table contains log-entry-related data defined by the application developer.

Field Name	Description
iLogEntryNum	Log entry to which this data is related.
iSequence	Used to order records for the same entry.
chKey	Customer defined. Limited to 48 characters.
chData	Customer defined. Limited to 80 characters.

Primary unique key: iLogEntryNum (asc), iSequence (asc)
Secondary key: chKey (asc)
Related to: SDLogEntry By field: iLogEntryNum Relationship: many to one

SDAALMTbl

SDAALMTbl contains information about Unattended Log Management.

Field	Description
bEnable	Flags whether or not the scheduled maintenance is enabled.
iDeleteActionType	Specifies Delete action type: 1: Delete All 2: Delete Selected
iArchActionType	Specifies Archive action type: 1: Archive All 2: Archive Selected 3: Do Not Archive
iArchiveMethod	Specifies Archive method: 1: Replace Files 2: Append to Files
szArchiveFileName	Name of the archive file. Limited to 256 characters.
iMaxArchiveFiles	Specifies the maximum number of archive files.
iTOD	Time of day (in seconds) of the scheduled maintenance.
iFrequency	Frequency of the maintenance: 1: Once 2: Hourly 3: Daily 4: Weekly 5: Monthly
dateOnce	Date maintenance should be performed if iFrequency is once. This date should not be earlier than the current date.
iHourlyInterval	Number of hours until the next scheduled hourly maintenance.
iDailyInterval	Number of days until the next scheduled daily maintenance.
iWeeklyInterval	Number of weeks until the next scheduled weekly maintenance.

Field	Description
iDayOfWeek	Day of the week the scheduled weekly maintenance should be performed: 1: Sunday 2: Monday 3: Tuesday 4: Wednesday 5: Thursday 6: Friday 7: Saturday
iMonthly	Specifies one of the two monthly choices: 1: Numeric day (between 1 and 31) 2: Combination of specific week (First through Fourth) and day of the week (Sunday through Saturday)
iDayOfMonth	Day of the month the scheduled monthly maintenance should be performed. If the specified number is larger than the last day of the month, maintenance is performed on the last day of the month.
iWeekOfMonth	Week of the month the scheduled monthly maintenance should be performed: 1: First 2: Second 3: Third 4: Fourth 5: Last
iDayOfWeekMonthly	Day of the specified week (in iDayOfWeekMonthly) the scheduled monthly maintenance should be performed: 1: Sunday 2: Monday 3: Tuesday 4: Wednesday 5: Thursday 6: Friday 7: Saturday
iAmountCalcMethod	Calculation method used in choosing the amount of the audit log record: 1: By Percentage (in iPercentage) 2: By Number of Records (in iNumRecords) 3: By Number of Days, Weeks, or Months (in iUnitAmount) 4: By This Date (in dateOnOrBeforeDate)

Field	Description
iPercentage	Percentage of the audit log record affected by maintenance.
iNumRecords	Number of audit log records affected by maintenance.
iUnitChoice	Unit choice among Days, Weeks, and Months: 1: Days 2: Weeks 3: Months
iUnitAmount	Amount of the unit choice. The maximum number can vary depending on the unit selected; the following values are suggested: 365 days, 52 weeks, 12 months.
dateOnOrBeforeDate	Date of the audit log records on which the action should be performed. This date should be no later than the current date.
iLastSchdMaint	Number of seconds from January 1, 1970 to the date that the last scheduled maintenance occurred.
bTestBusy	A flag to be set by the test program. When set, this flag tells you the test program is still in the verification state. This flag is checked by the Audit Log Maintenance Daemon/Service to see if it should start the next scheduled maintenance when it's due.
iTimeDelta	Amount of time, in seconds, added to the current system time in order to advance the system clock to the time when the maintenance should be carried out. During the test, this field should contain a value greater than 0. This method is used as an alternative to altering the system clock.
bDaemonBusy	A flag to be set by the Audit Log Maintenance Daemon/Service just before starting the maintenance work. This flag is monitored by the test program before it verifies the result for the correctness.

SDLogEntry

SDLogEntry holds log entries. The actual messages are stored in SDLogMessage.

Field Name	Description
*iLogEntryNum	Used to relate to customer extension records.
dtGMTDate	UTC date the event occurred.
tGMTTOD	UTC time of day in seconds of when the event occurred.
dtLocalDate	Local date when the event occurred.
tLocalTOD	Local time of day in seconds when the event occurred.
iMessageNum	What event happened.
chUserName	User name. Limited to 48 characters.
chLogin	Login ID. Limited to 48 characters.
chAffectedUserName	Affected user name. Limited to 48 characters.
chTokenSerialNum	Affected user token serial number. Limited to 12 characters.
chSiteName	Site name. Limited to 48 characters.
chGroupName	Group name. Limited to 96 characters.
chClientName	Agent Host name. Limited to 48 characters.
chServerName	Server name. Limited to 48 characters.
iPID	Process ID.
iMinorError	Minor error number.

Primary unique key: iLogEntryNum (asc)
Secondary key: dtGMTDate (asc), tGMTTOD (asc)
Secondary key: iLogEntryNum (asc), iMessageNum (asc), iSeverity (asc)
Related to: SDLogMessage By field: iMessageNum Relationship: many to one
Related to: CustLogExtension By field: iLogEntryNum Relationship: one to zero or many

SDLogMessage

SDLogMessage translates message numbers to text.

Field Name	Description
iMessageNum	Message number, messages above 10,000 customer messages.
chShortMessage	Short message text. Limited to 33 characters.
chLongMessage	Long message text. Limited to 256 characters.
iSeverity	Severity level: 0: informational 1: non-critical error 2: critical, needs immediate attention
iIncidentSearchCount	Number of records to search back.
iIncidentMatchCount	Number of records to match.
iIncidentEvent	Precursor events to search. 4 (does this mean a limit of 4?)
bSysLogFlag	Flag for system log inclusion. Default = FALSE.

Primary unique key: iMessageNum (asc)
Secondary key: iSeverity (asc), iMessageNum (asc)
Related to: SDLogEntry By field: iMessageNum Relationship: one to zero or many

SDLogReportFormat

SDLogReportFormat has only one record. It contains report format parameters.

Field Name	Description
iTermOrFile	Output report to screen or to file. 1 Screen 2 File
chRepFileName	The default filename to store report output. Limited to 256 characters.
bPrintHeader	Should the report header be included? Default: TRUE
bPageBreak	Should there be page breaks between pages? Default: TRUE
ILinesPerPage	Number of lines per page for reports that are sent to a file. Limited to 24 lines.

Field Name	Description
iTokenIdentifier	Should reports show the affected username, token serial number, or both? 1: Both 2: Token serial number 3: Username
chActRepTitle	Default Activity report title. Limited to 30 characters.
chExcRepTitle	Default Exception report title. Limited to 30 characters.
chIncRepTitle	Default Incident report title. Limited to 30 characters.
chUsageSumTitle	Default Usage Summary title. Limited to 30 characters.
chMonRepTitle	Default Monitor report title. Limited to 30 characters.
bFullNames	If TRUE, log reports will display full usernames. Default = FALSE.

SDSysLogCriteria

The system log criteria table contains information for filtering audit log messages for system log inclusion.

Field Name	Description
bSysLogFlag	If TRUE, send certain messages to the system log. Default = FALSE.
chCompareStr	Compare string (|) pipe delimited. Default = “’’. Limited to 256 characters.
chCurrentLogin	Current login. Limited to 49 characters.
chAffectedUserNum	Affected user. Limited to 49 characters.
chAffectedToken	Affected token. Limited to 13 characters.
chClientName	Agent Host name. Limited to 50 characters.
chServerName	Server name. Limited to 50 characters.

Error Messages and Codes

Toolkit functions return two types of error messages:

Function error messages. These messages indicate that a function has failed to accomplish its task in the database. These messages are not critical and do not require that you restart your C application or Tcl script.

Function errors are defined in **api_errors.h**. Each defined error is linked to a corresponding error message contained in **api_msgs.h**.

Communication error messages. These messages indicate that there is a communication problem between your C application or Tcl shell and the **apidemon**. When a communication error message is returned, RSA Security strongly recommends that you restart your application or shell.

Communication error messages are contained in the header file **message.h**, and are described in the following list.

Parameter *xxx*... is too long.

Severity: High. Causes the **apidemon** to shutdown.

Meaning: You are passing a string parameter that exceeds the internal buffer which is limited to 1024 bytes.

Solution: Shorten the string to 1024 bytes or less before calling the function again.

Output buffer is too short.

Severity: High. Causes the **apidemon** to shutdown.

Meaning: The output of the function is larger than the output buffer. Because no function has an output greater than **4097** bytes, the message might appear for either of two reasons:

- You decreased the size of **MAX_RESULT_MSG_SIZE** in the header file **apiuser.h**.
- Your memory has become corrupted.

Solution: Redefine **MAX_RESULT_MSG_SIZE**. For memory problems, verify your source code and reboot your system.

Database connection is not established!

Severity: Medium. Does not cause the **apidemon** to shutdown.

Meaning: The function you are calling attempts to work with the database, but the **apidemon** either is not running or is not connected to the database.

Solution: Start the **apidemon** (if necessary) and restore the connection to the database by calling **Sd_ApiInit**.

Interprocess communication failed!

Severity: Medium. Causes the **apidemon** to shutdown.

Meaning: The application or script has lost its connection with the **apidemon**.

Solution: Restart the application or Tcl shell and restart the **apidemon** with a call to one of the following functions: **Sd_ApiInit**, **Sd_ApiRev**, or **Sd_HexMD5**. Check resources such as memory and disk space.

Interprocess communication timeout!

Severity: Medium. Causes the **apidemon** to shutdown.

Meaning: There was no response from the **apidemon** within five minutes after it began the specified task. Five minutes is a default time limit. This setting can be changed in the **apidemon.ini** file.

Solution: Restart the application or Tcl shell and restart the **apidemon** with a call to one of the following functions: **Sd_ApiInit**, **Sd_ApiRev**, or **Sd_HexMD5**. Avoid having several processes that simultaneously perform an API task.

Failed to init demon!

Severity: Medium. This message indicates that the **apidemon** is not running.

Meaning: The **apidemon** cannot be found in the toolkit directory.

Solution: Check the environment variables listed in **admenv**.

Function can't be called in this state!

Severity: Low. Does not cause the **apidemon** to shutdown.

Meaning: The function that you just called cannot be executed due to the current state of the **apidemon**. For example, if **Sd_ApiInit** is called twice, the second call returns this message because the **apidemon** is already running.

Solution: This message can be ignored until execution of the program or script is completed. To avoid future occurrences, debug the program or script to eliminate redundant function calls.

Demon/Library or Demon/TCL mismatch!

Severity: High. Causes the **apidemon** to shutdown.

Meaning: The version of the **apidemon** you are running does not match the library version.

Solution: Download matching versions of the **apidemon** and the library (available upon request from RSA Security). If you are using C, rebuild your application with the new version of the library.