Open System Consultants
Radiator RADIUS Server

RSA SecurID Ready Implementation Guide

Last Modified: August 3, 2015

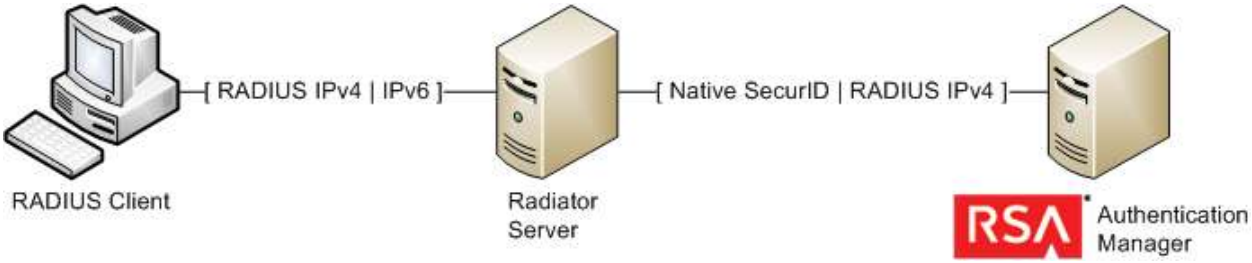
Partner Information

Product Information	
Partner Name	Open System Consultants
Web Site	www.open.com.au
Product Name	Radiator RADIUS Server with AuthBy ACE
Version & Platform	Version 4.15
Product Description	A full featured, flexible, full source RADIUS server with native RSA SecurID support.

Radiator
www.open.com.au

Solution Summary

Open System Consultants Radiator is a highly flexible, multi-platform RADIUS server that is easy to customize. The server can be configured as an RSA Authentication Manager agent or a RADIUS client to enable RSA SecurID two-factor authentication. In both cases, Radiator can extend or enhance RSA authentication or add RSA authentication to existing RADIUS, TACACS+ or Diameter-based user management or billing systems.



The Radiator *AuthBy ACE* module can be configured as a native RSA Authentication Manager agent. The module uses the *Authen-ACE4* Perl module and the RSA Authentication API to allow RSA Authentication Manager to authenticate user's RSA SecurID credentials.

The Radiator *AuthBy RADIUS* module can be configured as a proxy that forwards authentication requests to an RSA Authentication Manager RADIUS server.

RSA SecurID supported features	
Open System Consultants Radiator 4.15	
RSA SecurID Authentication via Native RSA SecurID Protocol	Yes
RSA SecurID Authentication via RADIUS Protocol	Yes
On-Demand Authentication via Native SecurID Protocol	Yes
On-Demand Authentication via RADIUS Protocol	Yes
Risk-Based Authentication	No
RSA Authentication Manager Replica Support	Yes
Secondary RADIUS Server Support	Yes
RSA SecurID Software Token Automation	No
RSA SecurID SD800 Token Automation	No
RSA SecurID Protection of Administrative Interface	No

Agent Host Configuration

RSA Authentication Agents are custom or ready-made software applications that securely pass user authentication requests to RSA Authentication Manager. RSA provides the RSA Authentication Agent API for building custom agents, as well as a variety of out-of-the-box agents for protecting access to various operating systems and web resources.

Note: The Radiator native RSA Authentication Manager integration uses the AuthBy ACE custom RSA Authentication Agent API agent.

All agents must be registered with RSA Authentication Manager in order for the server to establish secure communication channels with them. Use the RSA Security Console to register an agent for each Radiator server in your environment.

You need the following information to do so:

- the hostname of the Radiator server
- IP addresses for all of the Radiator server's network interfaces

When you register an Authentication Agent, set its agent type to *Standard Agent*.

Note: Each agent hostname must resolve to one or more valid IP addresses on the local network.

If you are configuring Radiator as a proxy to RSA Authentication Manager RADIUS server, you must use the RSA Security Console create a RADIUS client for each of your RSA Authentication agents.

You will need the following information to create a RADIUS client for a Radiator server:

- the hostname of the Radiator server
- IP addresses for all of the Radiator server's network interfaces
- the RSA RADIUS server's shared secret

Consult the *RSA Authentication Manager Administrator Guide* for more information about authentication agents and RADIUS clients.

RSA SecurID files

The RSA Authentication Agent configuration files that Radiator uses are located in the *C:\Windows\System32* directory on Windows and in the */var/ace* directory on UNIX. If you're configuring Radiator on a UNIX system and you want to store the files in a different location, you can use the Radiator configuration file's *ConfigDirectory* variable or the *VAR_ACE* environment variable.

RSA SecurID Authentication Files	
Files	**Location**
sdconf.rec	**UNIX**: */var/ace/*; **Windows**: *C:\Windows\System32*
Node Secret	**UNIX**: */var/ace/*; **Windows**: *C:\Windows\System32*
sdstatus.12	**UNIX**: */var/ace/;* **Windows**: *C:\Windows\System32*
sdopts.rec	**UNIX**: */var/ace/;* **Windows**: *C:\Windows\System32*

Note: The appendix of this document contains more detailed information regarding these files.

Partner Product Configuration

Before You Begin

This document provides instructions for enabling RSA SecurID two-factor authentication for Radiator users. You should have working knowledge of RSA Authentication Manager and Radiator, as well as access to the appropriate administrative documentation. Ensure that that both products are running properly prior to configuring the integration.

Note: This document is not intended to suggest optimal installations or configurations.

Configuration Overview

Decide if you want to configure the RSA Authentication Manager native API integration or the RADIUS proxy integration, and follow the corresponding instructions below.

- Configure Radiator as a native RSA Authentication Agent (*AuthBy ACE module)*
- Configure Radiator as an RSA Authentication Manager RADIUS client (*AuthBy RADIUS)*

Configure Radiator as a native RSA Authentication Agent (*AuthBy ACE module)*

This section contains the basic steps to install the *AuthBy ACE* module on a Radiator host and configure it to communicate with an RSA Authentication Manager server pool. For detailed configuration instructions, see the *%RADIATOR_HOME%/goodies/ace.txt* file

AuthBy ACE Prerequisites

Before you can configure the *AuthBy Ace* module, you must install Perl, the *Authen::ACE4* Perl module and RSA Authentication Manager Agent API on your Radiator host(s). See radiator host operation system-specific instructions below.

- Complete AuthBy ACE Prerequisites for Windows Systems
- Complete AuthBy ACE Prerequisites for UNIX Systems

Complete AuthBy ACE Prerequisites for Windows Systems

1. Configure each Radiator host as an RSA Authentication Agent.
2. Download the RSA Authentication Agent C SDK version 8.1.3 (*AuthSDK_C_8.1.3.zip)* or later from https://knowledge.rsasecurity.com, unzip the SDK file in the *C:* drive root directory and rename the unpacked directory *ACEAgentSDK*. (The unzipped directory should be C:\ACEAgentSDK.)
3. Download an *sdconf.rec* file from the RSA Security Console and copy it to *%windir%\system32*.
4. Download ActivePerl 5.8 or later from www.activestate.com, install it on your Radiator host and modify the *PATH* environmental to include the location of its *bin* directory.
5. Install Microsoft's *nmake* build utility and modify the *PATH* environmental to include the utility's location. (Note: *nmake* is packaged with Microsoft Visual Studio. See http://msdn.microsoft.com for details. Consult the *Authen-ACE4* module's documentation for alternatives to *nmake*).
6. Download the *Authen::ACE4* Perl module from www.cpan.org or www.open.com.au/radiator/free-downloads and unzip/untar the archive to a local directory. Refer to the module's README file for build and installation instructions

Complete AuthBy ACE Prerequisites for UNIX Systems

1. Configure each Radiator host as an RSA Authentication Agent.
2. Download the RSA Authentication Agent C SDK version 8.1.3 (*AuthSDK_C_8.1.3.zip)* or later from https://knowledge.rsasecurity.com, unzip the SDK file in the */opt/ace* directory and rename the unpacked directory *ACEAgentSDK* (i.e. The unzipped directory should be */opt/ace/ACEAgentSDK*.)
3. Download an *sdconf.rec* file from the RSA Security Console.
4. If you set your Radiator configuration file's *ConfigDirectory* variable or your *VAR_ACE* environment variable to a directory path, copy the *sdconf.rec* file to that path. Otherwise, copy the file to the */var/ace* directory
5. Download and install gcc if it is not already installed.
6. Download ActivePerl 5.8 or later from www.activestate.com and install it.
7. Download, compile and install the *Authen::ACE4* Perl module from www.cpan.org or www.open.com.au/radiator/free-downloads. Refer to the module's README file for build and installation instructions.

Configure the AuthBy ACE Module

Once you have completed the prerequisites on your Radiator host(s), follow the instructions below to configure the *AuthBy Ace* module:

1. Create a Radiator configuration file with an <*AuthBy ACE*> clause. Use the *%RADIATOR_HOME%/goodies/ace.cfg* sample configuration file as a starting point.
2. Start Radiator and use the *config_file* command line flag to specify your configuration file.
3. Test basic Radiator authentication. Use the *radpwtst* program to send sample RADIUS authentication requests to Radiator, which will use the RSA Authentication Agent API to forward the requests to one of your RSA Authentication Manager servers for validation.
4. Complete your Radiator configuration based on your requirements. Refer to the Radiator documentation for full details on the options available to you.

Configure Radiator as a RADIUS Client (*AuthBy RADIUS module)*

Follow the instructions below if you want to configure your Radiator server as a proxy to your RSA Authentication Manager RADIUS server(s).

1. Use the RSA Security Console to create a RADIUS client for each of your Radiator hosts. Remember the shared secret that you set, as you will need it to configure Radiator.
2. Create a Radiator configuration file with an <*AuthBy RADIUS*> clause. Use the *%RADIATOR_HOME%/goodies/proxy.cfg* sample configuration file as a starting point.
3. Set the *Host* variable in your configuration file to your primary RSA Authentication Manager RADIUS server's host name or IP address.
4. For each additional RSA Authentication Manager RADIUS server you want to configure, add another *Host* variable on a new line and set its value to the server's host name or IP address.
5. Set the *AuthPort* variable to your RSA Authentication Manager RADIUS authentication port.
6. Set the *Secret* variable to the shared secret from step 1.
7. Complete your Radiator configuration based on your requirements. Refer to the Radiator documentation for full details on the options available to you
8. Start Radiator and use the *config_file* command line flag to specify your configuration file.

Certification Checklist for RSA Authentication Manager 8.1

Date Tested: July 29, 2015

Certification Environment		
Product Name	**Version**	**Operating System**
RSA Authentication Manager	8.1.1	Virtual Appliance
RSA Authentication Agent API	8.1.3	Windows 2008
Radiator	4.15	Windows 2008

Mandatory Functionality			
RSA Native Protocol		**RADIUS Protocol**	
New PIN Mode			
Force Authentication After New PIN	✓	Force Authentication After New PIN	✓
System Generated PIN	✓	System Generated PIN	✓
User Defined (4-8 Alphanumeric)	✓	User Defined (4-8 Alphanumeric)	✓
User Defined (5-7 Numeric)	✓	User Defined (5-7 Numeric)	✓
Deny 4 and 8 Digit PIN	✓	Deny 4 and 8 Digit PIN	✓
Deny Alphanumeric PIN	✓	Deny Alphanumeric PIN	✓
Deny Numeric PIN	✓	Deny Numeric PIN	✓
Deny PIN Reuse	✓	Deny PIN Reuse	✓
Passcode			
16 Digit Passcode	✓	14 Digit Passcode	✓
4 Digit Fixed Passcode	✓	4 Digit Fixed Passcode	✓
Next Tokencode Mode			
Next Tokencode Mode	✓	Next Tokencode Mode	✓
On-Demand Authentication			
On-Demand Authentication	✓	On-Demand Authentication	✓
On-Demand New PIN	✓	On-Demand New PIN	✓
Load Balancing / Reliability Testing			
Failover (3-10 Replicas)	✓	Failover	✓
No RSA Authentication Manager	✓	No RSA Authentication Manager	✓

JGS

✓ = Pass ✗ = Fail N/A = Not Applicable to Integration

Appendix

Partner Integration Details	
RSA SecurID API	8.1.3
RSA Authentication Agent Type	Standard Agent
RSA SecurID User Specification	Designated Users
Display RSA Server Info	No
Perform Test Authentication	Yes
Agent Tracing	Yes

API Details:

The version of the API shared libraries depends on the version of the RSA Authentication Agent or API that you supply. The integration was last tested with the RSA Authentication API version 8.1.3.

RSA Authentication Manager Configuration Files

Node Secret:

RSA Authentication Manger will create a node secret for your agent on your Radiator Server in response to the first successful authentication. It will store the node secret in the *C:\Windows\System32* folder on Windows and the */var/ace* folder on Unix*.

sdconf.rec:

Copy the *sdconf.rec* file to the *C:\Windows\System32* folder on Windows and the */var/ace* folder on Unix*.

sdopts.rec:

If you want to use the *sdopts.rec* configuration file to set up manual load balancing, copy the file to the *C:\Windows\System32* folder on Windows and the */var/ace* folder on Unix*.

sdstatus.12:

The agent keeps the *sdstatus.12* file in the *C:\Windows\System32* folder on Windows and the */var/ace* folder on Unix*.

*__Note__: If you're configuring Radiator on a UNIX system and you want to store the files in a different location, you can use the Radiator configuration file's *ConfigDirectory* variable or the *VAR_ACE* environment variable.

Agent Tracing (Windows):

Use *regedit* to locate the *HKEY_LOCAL_MACHINE\Software\SDTI\ACECLIENT* key and create two *DWORD* values: *tracelevel* and *tracedest***.**

The value *tracelevel* specifies the verbosity and the categories of messages produced by the code. The value *tracedest* controls the output destination of the trace messages.

tracedest values:

```
SDITRACE_EVENT_LOG  0x00000001   // messages to event log
SDITRACE_CONSOLE    0x00000002   // messages to console
SDITRACE_LOGFILE    0x00000004   // messages to logfile (aceclient.log)
SDITRACE_DEBUGGER   0x00000008   // messages to debugger output
SDITRACE_NOFILELINE 0x80000000   // no file and line information
```

The *SDITRACE_NOFILELINE* value can be combined with any of the other values to stop the display of file and line number information. The logfile is *%SystemRoot%\ACECLIENT.LOG*, but you can change its location by creating a *REG_SZ:tracefile* value and specifying the file pathname.

tracelevel values:

```
SDITRACEING_OFF     0x000000000  // All messages off
SDITRACEING_ON      0x000000001  // All messages marked with this level on
SDITRACEING_ENTRY   0x000000002  // All entrypoints use this
SDITRACEING_EXIT    0x000000004  // All function returns use this
SDITRACEING_FLOW    0x000000008  // All logic flow control use this (ifs)
SDITRACEING_GRP1    0x000000010  // Old SDITRACE macros use this (see dbglib.h)
```

The hex value *0xF* gives the complete set of tracing. The values can be combined to produce multiple sets of trace messages.

Note: Using the *SDITRACE_CONSOLE* value can cause the service applications to access violate during log off. Only use this setting for real time debugging situations.

Agent Tracing (Linux / Solaris):

In order to configure agent tracing on UNIX systems, you must configure Radiator's initialization script to set and export environment variables. Add a variable called *RSATRACELEVEL* and give it an integer value to configure the tracing level. The list of values are integer representations of the Windows hexadecimal values listed above, with a value of 0 disabling tracing and a value of 15 enabling all trace options.

To configure a log file path, add a variable called *RSATRACEDEST* and set to the location you choose.