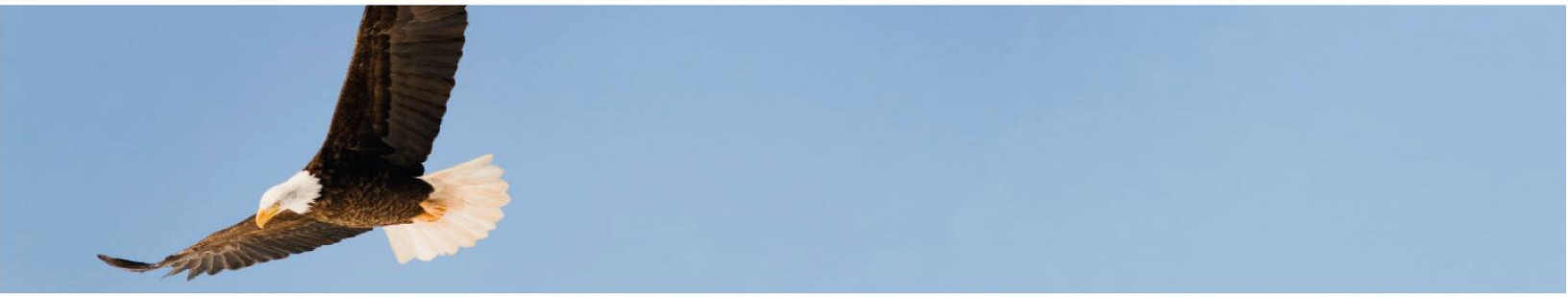
Salt Group



Replacing RSA Authentication Manager with Echidna

Version 15.1

May 2015

Contents

Chapter 1: Before starting 5

1.1 About Echidna 5

1.2 Using Echidna to replace RSA Authentication Manager 6

1.3 Example: Echidna brokers authentication requests from two sources 7

Chapter 2: Prepare to configure brokering 8

2.1 Install Echidna 8

2.2 Check the protocol used by the RSA clients 8

2.3 Identify existing RADIUS configuration details 9

2.4 Identify test users 9

2.5 Configure the firewall 9

Chapter 3: Configure communication between Echidna and RSA Authentication Manager 10

3.1 Configure RSA Authentication Manager to allow Echidna as a Client 10

3.2 Configure Echidna to call RSA Authentication Manager 11

Chapter 4: Configure communication between the RSA clients and Echidna 12

4.1 Add RSA clients to the Echidna access list 13

4.2 Configure each RSA client to call Echidna instead of RSA 13

4.3 Update native RSA components 14

4.4 Migrate RSA administrator accounts 14

4.5 Migrate or eliminate shared accounts 14

Chapter 5: Migrate any RADIUS profiles 15

5.1 Overview of steps to migrate RADIUS profiles 16

5.2 Create a resolution process to manage the RADIUS profiles 17

5.3 Create a new database for the RADIUS profiles 18

5.4 Link the user to the RADIUS profile (for a directory user store only) 19

5.5 Link the user to the RADIUS profile (for a database user store only) 20

5.6 Populate the new RADIUS_PROFILES table 21

5.7 Include the RADIUS_PROFILES attributes in the authentication response 22

5.8 Allow the RADIUS profiles to be managed with the User Support console 24
5.9 Verify the RADIUS profile 25

Chapter 6: Verify the new configuration 26

Chapter 1: Before starting

This guide describes how to replace RSA Authentication Manager with Echidna.

Echidna is an authentication server that supports OATH hardware tokens, Salt mCodeXpress mobile tokens, and SMS OTP, plus many other authentication methods. Echidna provides a fully integrated and scalable authentication service, which supports high-assurance implementations in finance and government organisations.

1.1 About Echidna

Echidna is a multi-factor authentication server, designed to let staff and users not only access applications securely, but also add other security features to existing applications. For example, Echidna can add transaction-signing capabilities to an Internet banking application.

Echidna supports a range of authentication devices, allowing fit-for-purpose and no lock-in, irrespective of legacy systems.

Echidna is suitable for financial institutions, government departments, and enterprises.

Echidna accepts both RADIUS and HTTP/S requests, and it connects to your existing user store.

Echidna allows users to gradually migrate from the legacy solution, without requiring a massive single change. Echidna can continue to support legacy hardware tokens, by brokering authentication requests to the legacy server. As the legacy tokens expire, users can be given newer, more cost-effective tokens.

1.2 Using Echidna to replace RSA Authentication Manager

Echidna can broker authentication requests to a legacy authentication server such as RSA Authentication Manager. Echidna proxies the authentication requests to RSA, allowing it to validate legacy or proprietary tokens.

The organisation retains its existing authentication mechanisms and at the same time seamlessly introduces new lower-cost and convenient tokens.

Users with legacy tokens continue to use the legacy server. As the legacy tokens expire, users receive their new lower-cost standards-based tokens. Because the legacy RSA server continues to respond to authentication requests, there is minimal user or technology impact. User with the new tokens are unaware that their tokens are now validated by the Echidna server, as brokering for the user will no longer be required.

With this model, after the users have all been transparently migrated, the legacy authentication server can be either re-purposed or retired.

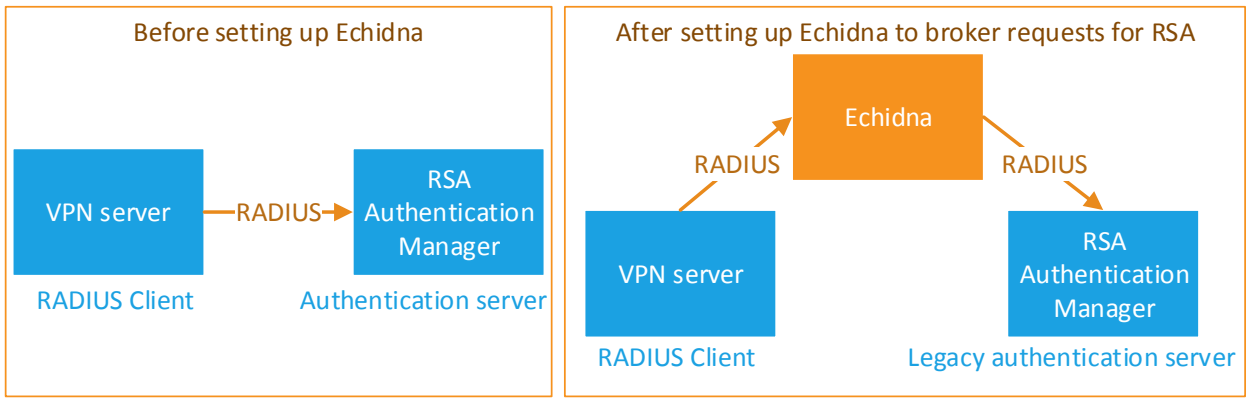
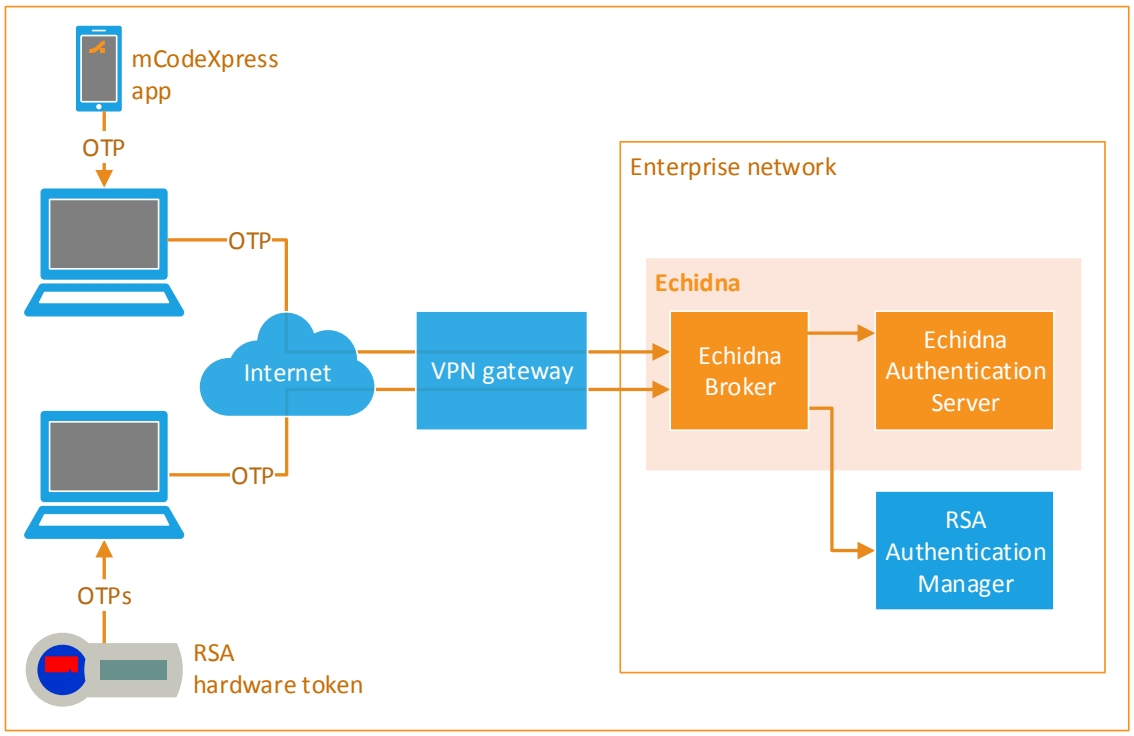


Figure 1: Echidna can be placed in front of a legacy authentication server

This diagram shows how Echidna provides a single target for all authentication traffic for the enterprise. Echidna handles some authentication requests itself, and it forwards other requests to RSA Authentication Manager and then waits for the response.

1.3 Example: Echidna brokers authentication requests from two sources

In the following example, the organisation uses both Salt mCodeXpress and legacy RSA hardware tokens. Echidna receives authentication requests from both sources. It handles the mCodeXpress requests itself, and it delegates the RSA requests to RSA Authentication Manager.



The Echidna broker works like this:

1. Echidna receives a RADIUS authentication request from the VPN gateway.
2. Echidna looks up the user in the user store to identify their credential type.
3. Echidna uses the credential type to decide where to send the request:
   - If the user has an mCodeXpress token, Echidna services the request.
   - If the user has a SecurID token, Echidna delegates the request to RSA Authentication Manager.
4. Echidna returns the authentication state to the VPN gateway.

Chapter 2: Prepare to configure brokering

To set up communication between Echidna and the legacy RSA Authentication Manager, follow these steps:

1. Install Echidna (see page 8)
2. Check the protocol used by the RSA clients (see page 8)
3. Identify existing RADIUS configuration details (see page 9)
4. Identify test users (see page 9)
5. Configure the firewall (see page 9)

2.1 Install Echidna

The steps in this book assume that Echidna has been installed.

Echidna is supplied in two formats: as a virtual appliance, and as an installable application. The instructions in this book work with both situations.

To install Echidna as a virtual appliance, follow the instructions in the Echidna *Installation Guide*. Follow every chapter except Chapter 7, which briefly describes brokering. This book gives much more detail.

2.2 Check the protocol used by the RSA clients

Check that existing RSA clients communicate with RSA via RADIUS, not via RSA's proprietary methods. This book does not cover how to handle clients that use the SecurID proprietary protocol to communicate with RSA.

Most products that support the proprietary RSA protocol also support RADIUS. Before configuring brokering through Echidna, migrate the clients to the standard RADIUS protocol first.

2.3 Identify existing RADIUS configuration details

Check the configuration of the existing RADIUS clients and take note of the following information:

- The hostnames (or IP addresses) and ports of the RADIUS servers.
- Whether there are multiple servers in a failover arrangement, and what timeouts are being used.
- Any RADIUS profile attributes being used from the RADIUS responses.
- Management of the RSA user identities. Are the users maintained only in RSA Authentication Manager's internal database, or has integration to AD been configured to find and authenticate users, and to resolve group memberships. If AD integration is used, note the LDAP directory URLs being used and whether they are Global Catalog addresses or regular AD nodes.

2.4 Identify test users

Identify the test users in the user store.

To validate the behaviour of the existing systems that are relying on RSA Authentication Manager and to ensure that the Echidna configuration enables the same behaviour, RSA Authentication Manager users should be created or identified who can be involved in this validation process.

After completing the setup, these users can be used to issue authentication requests in order to verify whether the authentication is brokered correctly or not.

2.5 Configure the firewall

Echidna and RSA Authentication Manager both communicate using RADIUS over a UDP connection.

2.5.1 Allow the RSA clients to communicate with Echidna

Identify the ports Echidna uses (for example, 1812 or 1645) and configure the firewall to allow connections to those ports.

2.5.2 Allow Echidna to communicate with RSA Authentication Manager

Identify the ports that RSA Authentication Manager uses and configure the appliance firewall to allow connections to those ports.

Chapter 3: Configure communication between Echidna and RSA Authentication Manager

To set up communication between Echidna and the legacy RSA Authentication Manager, follow these steps:

1. Configure RSA Authentication Manager to allow Echidna as a Client (see page 10)
2. Configure Echidna to call RSA Authentication Manager (see page 11)

3.1 Configure RSA Authentication Manager to allow Echidna as a Client

Use RSA Authentication Manager **Security Console** to add Echidna as a new RADIUS client, using these details:

- The hostname and IP address of the Echidna server(s)
- A shared secret value (a random password recommended to be at least 22 characters long). The same shared secret will be needed in configuring Echidna in the next section.

See the "Add a RADIUS Client" section in the RSA Authentication Manager *Administrator's Guide*.

3.2 Configure Echidna to call RSA Authentication Manager

Create a new authenticator to talk to RSA Authentication Manager, and ensure that it gets added to the existing authentication process.

1. Log in to the Echidna Administration console.
2. Click **CONFIG**, then click **Authenticators**.
3. On the left under **Authenticators**, click **new**, then enter details about the new authenticator:
   a. Type a name for this authenticator.
   b. Select **Remote RADIUS** as the authenticator type.
   c. In the **Authentication Processes** section at the bottom, leave **authenticationProc** selected. For more information, see "How processes work" in the Echidna *Administration Reference*.
4. Enter the following information:
   - **User ID Link Attribute**: The user attribute(s) that will be used for the remote RADIUS request. For example, ${storeName}/${uid}.
   - **Remote Request Retry Count**: The number of times to attempt to send a request to RSA Authentication before giving up.
   - **Remote Server Routing Algorithm**: The method for sending requests to alternative RSA servers if a request times out.
5. In the Remote Server List section, define each RSA Authentication Manager server that Echidna can delegate to.
   - **Remote RADIUS Server Host Name**: The hostname of RSA Authentication Manager.
   - **Remote RADIUS Server Auth Port**: The RADIUS authentication port, which is usually 1812 or 1645.
   - **Response Timeout**: The timeout for requests to this RSA server.

     Ensure that this value makes sense when used with the **Remote Request Retry Count** value in Step 4. The Response Timeout value applies for each retry, plus any timeouts apply when a request is failed over to another RSA server. Long timeouts can add up to very long response times.

     Also ensure that the timeout value does not conflict with the time validity of the OTP in use. For example, if the OTPs are valid for only 30 seconds, ensure that related authentication requests cannot be retried for longer than 30 seconds.
   - **RADIUS Client Shared Secret**: The shared secret from Configure RSA Authentication Manager to allow Echidna as a Client on page 10.
6. Click **Update**, then save the Echidna configuration.

Chapter 4: Configure communication between the RSA clients and Echidna

To set up communication between the RSA clients and Echidna, follow these steps:

1. Add RSA clients to the Echidna access list (see page 13)
2. Configure each RSA client to call Echidna instead of RSA (see page 13)
3. Update native RSA components (see page 13)
4. Migrate RSA administrator accounts (see page 14)
5. Migrate or eliminate shared accounts (see page 14)

(Optional)

6. Migrate any RADIUS profiles (see page 15)

4.1 Add RSA clients to the Echidna access list

1. Log in to the Echidna Administration console.
2. Click the **Connectors** tab.
3. In the **Clients** section on the left, click **new**.
4. Enter the host name of the RSA client, then click **Next**.
5. Enter the shared secret that will be used by the RSA clients to connect to Echidna.
6. Click **Update** to save the client details.

4.2 Configure each RSA client to call Echidna instead of RSA

When an existing client already authenticates to RSA Authentication Manager, it must be updated to authenticate to Echidna instead. Echidna will then forward authentication requests to RSA.

4.2.1 Update a web services client

To update a client that currently uses web services to communicate with RSA Authentication Manager, use the Echidna web services API.

To access the Echidna web services API:

1. Create a client in Echidna:
   a. Log in to the Echidna Administration console.
   b. Click the **Connectors** tab.
   c. In the **Clients** section on the left, click **new**.
   d. In the **New RADIUS Client Name or IP Address** box, enter a username, then click **Next**.
   e. Create a new password by entering it into the **New Password** and **Confirm Password** boxes.
   f. Click **Update**.
2. In a web browser, go to the address for the API.
   Use the address of the Echidna server and the port number of the HTTP web services.
   For example, if the Administration console is at https://198.51.100.62:8443 and the HTTP access point port is 1888, use http://198.51.100.62:1888 to access the web services API.
3. Log in using the username and password that were created for the new client.

4.2.2 Update a RADIUS client

To update a client that currently uses RADIUS to communicate with RSA Authentication Manager, update the following information:

- **IP address**: The IP address of Echidna
- **Shared secret**: The shared secret identified in Configure RSA Authentication Manager to allow Echidna as a Client on page 10.

4.3 Update native RSA components

If the RSA database contains any local users that are not present in the LDAP or AD user store, review these and decide what to do with them.

4.4 Migrate RSA administrator accounts

1. Create a list of all RSA administrators, and record the following information for each one:
   - User ID
   - First Name
   - Last Name
   - Last Login
   - Administrative Role Name
2. Use this information to understand which accounts are no longer required.
3. Delete any accounts that are no longer required.
4. Migrate other accounts to the user store. Use the RSA Authentication Manager's Security Console "Export Users with Tokens" task to assist with this migration.

4.5 Migrate or eliminate shared accounts

RSA Authentication Manager supports the use of accounts that are shared by many people. Shared accounts weaken both the security and the two-factor nature of the authentication, because the same PIN is used by more than one user. This means that the PIN is often written down or stored somewhere, and is not something that is uniquely known by one user.

Shared accounts cannot be audited effectively, because there is no certain way to detect who logged in at any particular time.

Shared accounts are often used in environments where hardware tokens are expensive. mCodeXpress mobile tokens are free to install, and do not include any additional costs outside of the Echidna license.

Salt Group strongly recommends that each user has their own individual software token, and that each user then is granted access only to those systems and applications that they need.

Although they can reduce security, Echidna supports accounts for entities that are not present in the Active Directory user store. These accounts can be created in the database used by Echidna as a user store. Use the database to store accounts for machines, applications, or shared accounts.

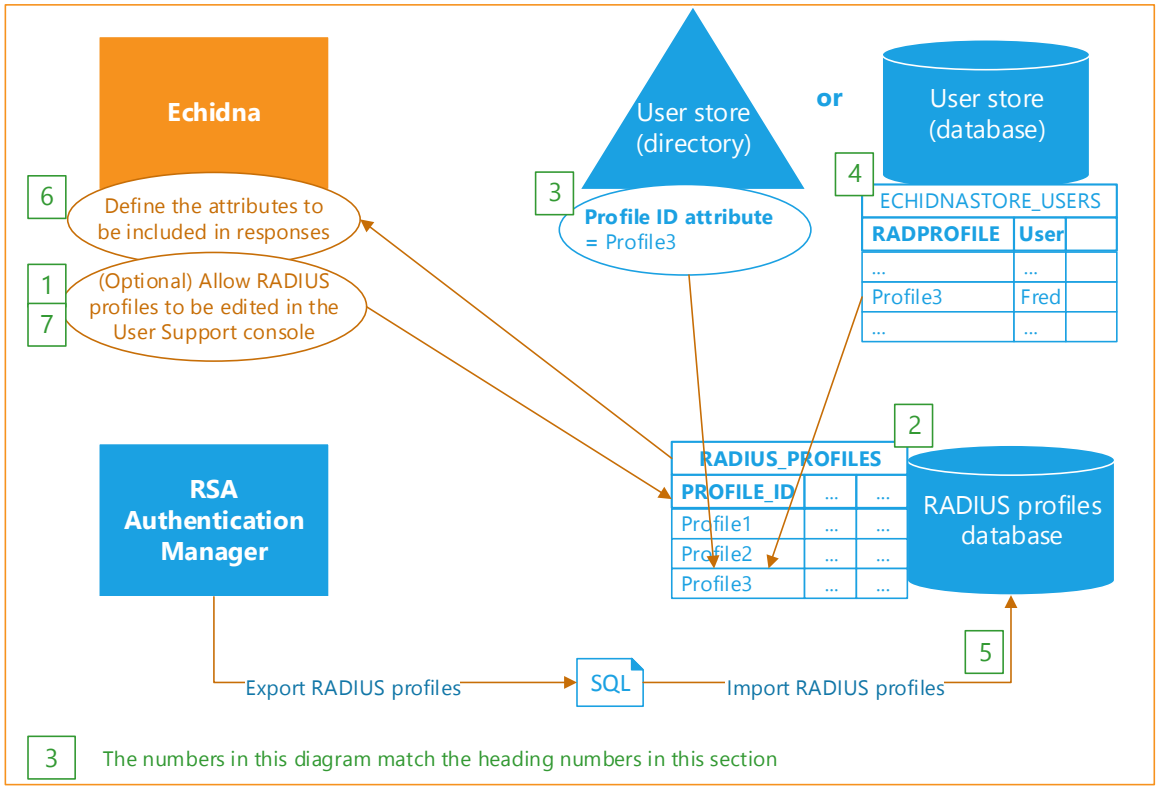
Chapter 5: Migrate any RADIUS profiles

A RADIUS profile lets a RADIUS server include authorisation information in its responses to clients. This allows clients to rely on the server to provide authorisation information, which can be simpler than having each client work this out.

When migrating from a legacy RADIUS server, RADIUS profiles may already have been in use. Echidna can continue to support them. Echidna stores RADIUS profiles in a database, independent of which type of user store it uses.

5.1 Overview of steps to migrate RADIUS profiles

The following diagram shows the components and connections described in this section:



To set up RADIUS profiles in Echidna, follow these steps:

1. Create a resolution process to manage the RADIUS profiles (see page 17)
2. Create a new database for the RADIUS profiles (see page 18)
3. Link the user to the RADIUS profile (for a directory user store only) (see page 19)
4. Link the user to the RADIUS profile (for a database user store only) (see page 20)
5. Populate the new RADIUS_PROFILES table (see page 21)
6. Include the RADIUS_PROFILES attributes in the authentication response (see page 22)
7. Allow the RADIUS profiles to be managed with the User Support console (see page 24)
8. Verify the RADIUS profile (see page 25)

5.2 Create a resolution process to manage the RADIUS profiles

A resolution process defines the user stores that can be managed in the User Support console. This is defined here so that it can be modified to omit the RADIUS profiles database in the next section. This prevents the RADIUS profiles database being misused as a user store.

1. Log in to the Echidna Administration console.
2. Click **CONFIG**, then click **User Stores**.
3. On the left, click **new** to create a new user resolution process.
4. Enter the following details:
   - **New user resolution process name**: support-manage-resolve, or a similar name that makes it clear that this resolve process controls which user stores can have content managed by the User Support console.
   - **User/Domain Patterns to Recognize**: Leave as is.
   - **Select user stores to resolve against**:
     - **Directory user stores**: Uncheck all directory user stores. These user stores cannot be managed in the User Support console.
     - **Database user stores**: Check the database user stores that will be manageable in the User Support console.
5. Click **Next**. Make no further changes to this process at this point.

5.3 Create a new database for the RADIUS profiles

1. Log in to the Echidna Administration console.
2. Click **CONFIG**, then click **User Stores**.
3. On the left, click **new** to create a new user store.
4. Enter the following details:
   - **New user store name**: db-profiles, or another name that makes it clear that this database contains only RADIUS profiles
   - **User store type**: JDBC Database with Existing Table
   - **Select the existing User-Resolve processes that should reference the new store**: Check only the resolve process that was defined in Create a resolution process to manage the RADIUS profiles on page 17. All other resolve processes should be unchecked.
5. Click **Next**, then set the following attributes of the database:
   - **Username Attribute**: PROFILE_ID
   - **Display Name Attribute**: PROFILE_LABEL. This attribute is used as the display name for RADIUS profiles appear for editing in the User Support console.
   - **Datastore Connection Name**: Use the same one used by the existing database user store if present. This enables foreign key constraints, which are used in a later section. By default, this is LocalStoreUnit.
6. Add a new table named **RADIUS_PROFILES** to the database:
   a. Find the **Tables** section, use the **Add** button  to add a table, then enter the following details:
      - **Schema Name**: RADIUS
      - **Table name**: RADIUS_PROFILES
   b. Set **Data Integrity Protector** to **utilMac**.
   c. Add the columns that are listed in this table:

Column Name	Column Type	Column Size	NOT NULL	Primary Key Order	Default Value
PROFILE_ID	VARCHAR	40	Checked	1	
PROFILE_LABEL	VARCHAR	140			
FRAMED_COMPRESSION	VARCHAR	40			
FRAMED_IP_ADDRESS	VARCHAR	40			
FRAMED_NETMASK	VARCHAR	40			
FRAMED_MTU	INTEGER				1500
FRAMED_PROTOCOL	VARCHAR	40			
FRAMED_ROUTING	VARCHAR	40			
FRAMED_ROUTE	VARCHAR	40			
SERVICE_TYPE	VARCHAR	40			

   d. At the bottom of the page, click **Update**. Echidna immediately uses the new configuration.
   e. Click **Create Persistence Table** to create the table.

5.4 Link the user to the RADIUS profile (for a directory user store only)

Define an attribute in the user store to hold the RADIUS profile ID.

If the directory already has an unused attribute with a string syntax (such as **extensionAttribute1** for Active Directory), this can be used. If there are no appropriate attributes, define a new one.

1. Log in to the Echidna Administration console.
2. Click **CONFIG**, then click **User Stores**.
3. In the **User Stores** list on the left, click the name of the directory.
4. Find the **Attribute Aliases** section, then add the following alias:
   - **Key**: profileID
   - **Value**: *attributename*, where this is the attribute that stores the RADIUS profile. For example, **extensionAttribute1**.
5. Click **Update**, then save the configuration changes.
6. Ensure that the user records contain the appropriate RADIUS profile identifier in the nominated attribute.

5.5 Link the user to the RADIUS profile (for a database user store only)

This section describes how to define a RADPROFILE column in the user store database and then link it to the PROFILE_ID column of the RADIUS_PROFILES table.

1. Log in to the Echidna Administration console.
2. Click **CONFIG**, then click **User Stores**.
3. In the **User Stores** list on the left, click on the user store name.
4. Find the **ECHIDNASTORE_USERS** table in the user store database.
5. Add a column to this table, with the following details:
   - **Name**: RADPROFILE
   - **Column Type**: VARCHAR(40),
   - **Constraint**: FOREIGN KEY REFERENCES RADIUS.RADIUS_PROFILES(PROFILE_ID). This foreign key constrains the RADPROFILE value to contain only a value that is a valid PROFILE_ID from the RADIUS profiles table.

     A foreign key constraint can be set only if the RADIUS profile store and the user store use the same database connection. This was set in Step 5 in Create a new database for the RADIUS profiles on page 18.
6. Find the **Updates** section, then modify the definitions of the **users.create Update** and **users.update Update** to include the **RADPROFILE** column.
7. At the bottom of the page, click **Update**. Echidna immediately uses the new configuration.
8. Click **Create Persistence Table** to create the table.
9. Find the **Attribute Aliases** section, then add the following alias:
   - **Key**: profileID
   - **Value**: RADPROFILE
10. Click **Update**, then save the configuration changes.

5.6 Populate the new RADIUS_PROFILES table

This section describes how to populate the RADIUS_PROFILES table directly, in bulk. Alternatively, the User Support console can be used to set up the RADIUS profiles one-by-one. See Allow the RADIUS profiles to be managed with the User Support console on page 24.

To populate the RADIUS_PROFILES table by executing a script directly against the database:

1. Export the current RADIUS profiles from RSA Authentication Manager to a text file.
2. Convert the exported text file into a script that can update the RADIUS_PROFILES database.

   For example, adapt the following example SQL script:

```
INSERT INTO RADIUS_PROFILES
  (PROFILE_ID, FRAMED_COMPRESSION, FRAMED_IP_ADDRESS, FRAMED_NETMASK, FRAMED_MTU,
FRAMED_PROTOCOL, FRAMED_ROUTING, FRAMED_ROUTE, SERVICE_TYPE)
VALUES
  ('PRIMUS','VJ-TCP-IP-header-
compression','10.20.15.228','255.255.255.255',1500,'PPP','None',NULL,'Framed'),
  (SECUNDUS','VJ-TCP-IP-header-
compression','255.255.255.1','255.255.255.255',1500,'PPP','None',NULL,'Framed'),
  ('TERTIUS','VJ-TCP-IP-header-
compression','10.20.15.254','255.255.255.255',1500,'PPP','Send-and-
Listen','10.20.200.0/24','Framed');
```

3. (Optional) Add a human-readable name for each profile in the PROFILE_LABEL column.
4. Run the script.

5.7 Include the RADIUS_PROFILES attributes in the authentication response

Configure the authentication process to include the RADIUS_PROFILES attributes in the authentication response.

1. Log in to the Echidna Administration console.
2. Click **CONFIG**, then click **Authenticators**.
3. In the **Authentication Procs** list on the left, click **authenticationProc**.
4. Use the **Add** button  next to **Process Conditions** to create a new condition of the type **AND**.
5. Scroll to the bottom of the page to see the newly added condition. For the remaining steps, this new condition is referred to as **condition.x**.
6. Add a sub-condition under condition.x, with the following details:
   - **New Condition Type**: EXACTMATCH
   - **Reference**: ${result.type}
   - **Match**: ACCEPT
7. Add another sub-condition under condition.x, with the following details:
   - **New Condition Type**: PRESENT
   - **Reference**: ${userContext.profileID}. This is the user attribute that is used to search for the RADIUS profile.
8. Add a sub-action under condition.x, with the following details:
   - **New Action Type**: ExpandUserConStep
   - **Context Variable Name**: profile
   - **Datastore Connection Name**: Enter the same connection name that was entered in Step 5 of Create a new database for the RADIUS profiles on page 18.
   - **Data Confidentiality Protector**: utilProt
9. Add a query with the following details:
   - **Query Name**: profile.resolve
   - **Query SQL**: SELECT * FROM RADIUS.RADIUS_PROFILES WHERE (PROFILE_ID = ${userContext.profileID})

10. For each context variable to be set, add a sub-action under condition.x, using the following details:
    - **New Action Type**: SetVariableStep
    - **Var Name**: response.Framed-Compression
    - **Expression**: ${profile.FRAMED_COMPRESSION}

    Repeat this step for each profile attribute that needs to be included in the response:

Var Name	Expression
response.Framed-Compression	${profile.FRAMED_COMPRESSION}
response.Framed-IP-Address	${profile.FRAMED_IP_ADDRESS}
response.Framed-IP-Netmask	${profile.FRAMED_NETMASK}
response.Framed-MTU	${profile.FRAMED_MTU}
response.Framed-Protocol	${profile.FRAMED_PROTOCOL}
response.Framed-Routing	${profile.FRAMED_ROUTING}
response.Framed-Route	${profile.FRAMED_ROUTE}
response.Service-Type	${profile.SERVICE_TYPE}

11. Click **Update**, then save the configuration changes.

5.8 Allow the RADIUS profiles to be managed with the User Support console

The User Support console is a web application that comes with Echidna. It lets customer support staff edit user and token details. It can also be used to update RADIUS profiles.

If the RADIUS profiles can be managed by the User Support console, service staff can create, edit, and delete RADIUS profiles using the User Support console.

If the RADIUS profiles cannot be managed in the User Support console, the only way to edit them is by editing the database directly.

To allow RADIUS profiles to be managed in the User Support console, expose a **user registration** web service so that the table can be managed through the User Support web application. A user registration service allows the entries in any database-backed user stores referenced in a given **user resolution** process to be managed.

1. Log in to the Echidna Administration console.
2. Click **CONFIG**, then click C**onnectors**.
3. In the **Services** list on the left, click **webServices**.
4. Find the **User Reg Services** section, use the **Add** button  to add a service, then enter the following information:
   - **Name**: userReg
   - **User Resolve Process Ref**: support-manage-resolve
5. In the **Published Services** list, find the new **userReg** row, then check the **Service Enabled** box.
6. Click **Update**, then save the configuration changes.

The RADIUS profiles can now be edited in the User Support console. To edit a RADIUS profile in the User Support console, follow these steps:

1. Log in to the User Support console.
2. Click **Manage Users**, then select **Users** under the **db-profiles** user store.
3. Click **New** to create a new profile. Each field of the RADIUS_PROFILES table is shown.
4. Enter values for each field that is required, then click **Create**.

5.9 Verify the RADIUS profile

Use a RADIUS profile testing tool to verify the configuration changes. RADIUS testing tools are available online.

To test the RADIUS profile changes, send an authenticate request and check the response. The response should include the correct RADIUS_PROFILES attributes based on the RADPROFILE value assigned to the user.

The following is a sample authentication response:


```
"authenticateResponse":{
"method":"authenticate",
"result":{
"@name":"SUCCESS",
"@type":"ACCEPT",
"@packetIdentifier":276,
"userIdent":{
"@userID":"User101",
"@domain":"dbMySQL"},
"extra":[
{
"@key":"Framed-MTU",
"@value":"1500"},
{
"@key":"Framed-IP-Netmask",
"@value":"/255.255.255.255"},
{
"@key":"Salt-User-Domain",
"@value":"dbMySQL"},
{
"@key":"Service-Type",
"@value":"Framed(2)"},
{
"@key":"Salt-User-Name",
"@value":"User101"},
{
"@key":"Framed-IP-Address",
"@value":"/255.255.255.1"},
{
"@key":"Framed-Protocol",
"@value":"PPP(1)"},
{
"@key":"Framed-Compression",
"@value":"VJ-TCP-IP-header-compression(1)"},
{
"@key":"Framed-Routing",
"@value":"None(0)"}]}
```

Chapter 6: Verify the new configuration

After setting up Echidna to broker requests to RSA Authentication Manager, test that the end-to-end communication works correctly, including the following:

1. As a user, log in to the Self Service console and set **Remote RADIUS** as the preferred login option.
2. As the same user, attempt to log in to the RADIUS client.
3. Check that the user is successfully authenticated by the authentication mechanism configured in the remote server.

Note: If the only client is currently localhost, test Echidna by sending a request from the local computer only. Echidna will not accept authentication requests from other clients. If the configuration includes the details of actual clients, Echidna will accept authentication requests from them.