Amigopod

Auto Create MAC Auth Account (Authentication Based)

ARUBA networks

Technical Note

Copyright

© 2011 Aruba Networks, Inc. Aruba Networks trademarks include Airwave, Aruba Networks®, Aruba Wireless Networks®, the registered Aruba the Mobile Edge Company logo, Aruba Mobility Management System®, Mobile Edge Architecture®, People Move. Networks Must Follow®, RFProtect®, Green Island®. All rights reserved. All other trademarks are the property of their respective owners.

Open Source Code

Certain Aruba products include Open Source software code developed by third parties, including software code subject to the GNU General Public License (GPL), GNU Lesser General Public License (LGPL), or other Open Source Licenses. The Open Source code used can be found at this site:

http://www.arubanetworks.com/open_source

Legal Notice

The use of Aruba Networks, Inc. switching platforms and software, by all individuals or corporations, to terminate other vendors' VPN client devices constitutes complete acceptance of liability by that individual or corporation for this action and indemnifies, in full, Aruba Networks, Inc. from any and all legal actions that might be taken against it with respect to infringement of copyright on behalf of those vendors.

Warranty

This hardware product is protected by the standard Aruba warranty of one year parts/labor. For more information, refer to the ARUBACARE SERVICE AND SUPPORT TERMS AND CONDITIONS.

Altering this device (such as painting it) voids the warranty.



www.arubanetworks.com
1344 Crossman Avenue
Sunnyvale, California 94089
Phone: 408.227.4500
Fax 408.227.4550

Table of Contents

1 Introduction 4

Audience 4

Document Overview 4

2 Solution Summary 5

Prerequisites 5

Test Environment 5

Aruba Controller Configuration 5

RADIUS Server Definition 5

MAC Authentication Profile 6

AAA Profile 6

Captive Portal Profile 6

Netdestination Alias for Amigopod 6

Access Policy to allow redirect to Amigopod 6

Initial Role with Captive Portal enabled 6

Post Authentication Role for MAC Authentication 6

Virtual AP 6

SSID Profile 7

Amigopod Configuration 8

RADIUS Role for MAC Accounts 8

RADIUS Role to trigger MAC Address Account Creation 10

Testing the Workflow 15

Create Test Account 15

Initial Connection Attempt 16

Subsequent Connection Attempt 18

3 Summary 19

4 Appendix – Browser Detection Extension 20

Pre-requisites 20

Create Browser Redirect Page 21

Create Laptop Detected Error Page 22

Update the Captive Portal Profile 23

5 Appendix – Limiting Paired Devices 24

1 Introduction

This technical note explains the configuration process for enabling the automatic creation of a MAC authentication user account within the Amigopod database based on a successful Web Login Authentication.

The use case for this design is to allow a device once initially authenticated via Captive Portal to have transparent network access on any subsequent connection to the network.

This design relies on the Aruba Controller being configured with an SSID that has MAC Authentication enabled with Captive Portal failover. This will allow any wireless devices that have not passed through the initial Captive Portal authentication process to fail MAC authentication and be presented with the Amigopod Web Login page.

Once authenticated the first time, a MAC authentication account will be created and all subsequent wireless connections will be transparent authenticated by RADIUS MAC authentication.

This design can be extended to include the initial authentication source being an external Active Directory or LDAP server. For more information of integrating Amigopod's authentication with Active Directory, please refer to the Amigopod deployment guide or External Authentication Tech Note.

Audience

This document is intended for network administrators and system integrators deploying an Amigopod-based visitor management solution.

Basic familiarity with the Amigopod Visitor Management Appliance is assumed. For in-depth information about the features and functions of the Amigopod appliance, refer to the Amigopod Deployment Guide.

Document Overview

The first section of the document explains the configuration steps required to configure the automatic creation of MAC authentication accounts on Amigopod. The Aruba Controller configuration items are reviewed but detailed explanation of the procedure to configure ArubaOS is assumed knowledge

The Appendix contains a reference guide for implementing browser detection of the WiFi client as an extension to this design. The goal is to only allow smartphones and tablets to be able to register their MAC address and not laptops that should be using the existing corporate SSID.

2 Solution Summary

Prerequisites

Support for the automatic creation of MAC authentication accounts requires the following plugin versions:

- Amigopod Guest Manager 3.5.x or later
- Amigopod RADIUS 3.0.x or later
- Amigopod MAC Authentication 3.5.x or later
- ArubaOS 6.0.x or later

To verify you have the correct plugin versions installed and enabled, navigate to **Administrator** > **Plugin Manager** > **List Available Plugins** and check the version number in the list.

Use the **Check for Plugin Updates** link to download and install updated plugins.

Test Environment

- Amigopod Virtual Appliance running on VMWare Fusion (3.1.13 Kernel, 3.1.10)
- Aruba 651 Controller with inbuilt WiFi Access Point (6.1.0.0-beta with PEF License)
- iPhone, iPad, Mac Book Pro MAC OS X, Dell Windows XP

Aruba Controller Configuration

The configuration discussed in this Tech Note assumes that the Aruba Controller has been configured with an SSID that has RADIUS MAC Address Authentication enabled with Failover to Captive Portal.

This Tech Note focuses on the customization aspects of the Amigopod solution and not the detailed configuration of the Aruba controllers. This is assumed knowledge and if more information is required on the correct configuration of the ArubaOS environment, please refer to the ArubaOS documentation or various VRD documents available from arubanetworks.com

The following configuration extracts provides an example of this setup:

RADIUS Server Definition

```
aaa authentication-server radius "amigopod"
   host "172.16.0.20"
   key f0e40f33109cd5f863a77327072720aaa4785eff2ca57800
   nas-identifier "Aruba651"
   nas-ip 172.16.0.254
!
aaa server-group "amigopod-srv"
 auth-server amigopod
!
aaa rfc-3576-server "172.16.0.20"
```

```
   key 10795ff19c00465dd0b0824e562103bee537be631e5bc876
```

MAC Authentication Profile

```
aaa authentication mac "amigopod-mac"
   case upper
   delimiter dash
```

AAA Profile

```
aaa profile "amigopod-aaa"
   authentication-mac "amigopod-mac"
   mac-default-role "authenticated"
   mac-server-group "amigopod-srv"
   radius-accounting "amigopod-srv"
   rfc-3576-server "172.16.0.20"
```

Captive Portal Profile

```
aaa authentication captive-portal "amigopod-cp"
   server-group "amigopod-srv"
   redirect-pause 3
   no logout-popup-window
   protocol-http
   login-page "http://172.16.0.20/aruba_login.php"
```

Netdestination Alias for Amigopod

```
netdestination amigopod
  host 172.16.0.20
```

Access Policy to allow redirect to Amigopod

```
ip access-list session allow-amigopod
  user   alias amigopod svc-http  permit
  user   alias amigopod svc-https  permit
```

Initial Role with Captive Portal enabled

```
user-role logon
 captive-portal "amigopod-cp"
 access-list session logon-control
 access-list session allow-amigopod
 access-list session captiveportal
```

Post Authentication Role for MAC Authentication

```
user-role MAC-Guest
 access-list session allowall
```

Virtual AP

```
wlan virtual-ap "MAC-Auth-CP"
```

```
   aaa-profile "amigopod-aaa"
   ssid-profile "MAC-Auth-CP"
```

SSID Profile

```
wlan ssid-profile "MAC-Auth-CP"
   essid "amigo-MAC-CP"
```

Amigopod Configuration

RADIUS Role for MAC Accounts

Create new RADIUS role to hold the logic for the automatic creation of the MAC account.

This role can contain any standard RADIUS or Aruba specific attributes that make sense within your deployment. In the example shown below, the **Aruba-User-Role** VSA is being used to signal to the Aruba controller to place all MAC authenticated clients into an ArubaOS Role of **MAC-Guest.**

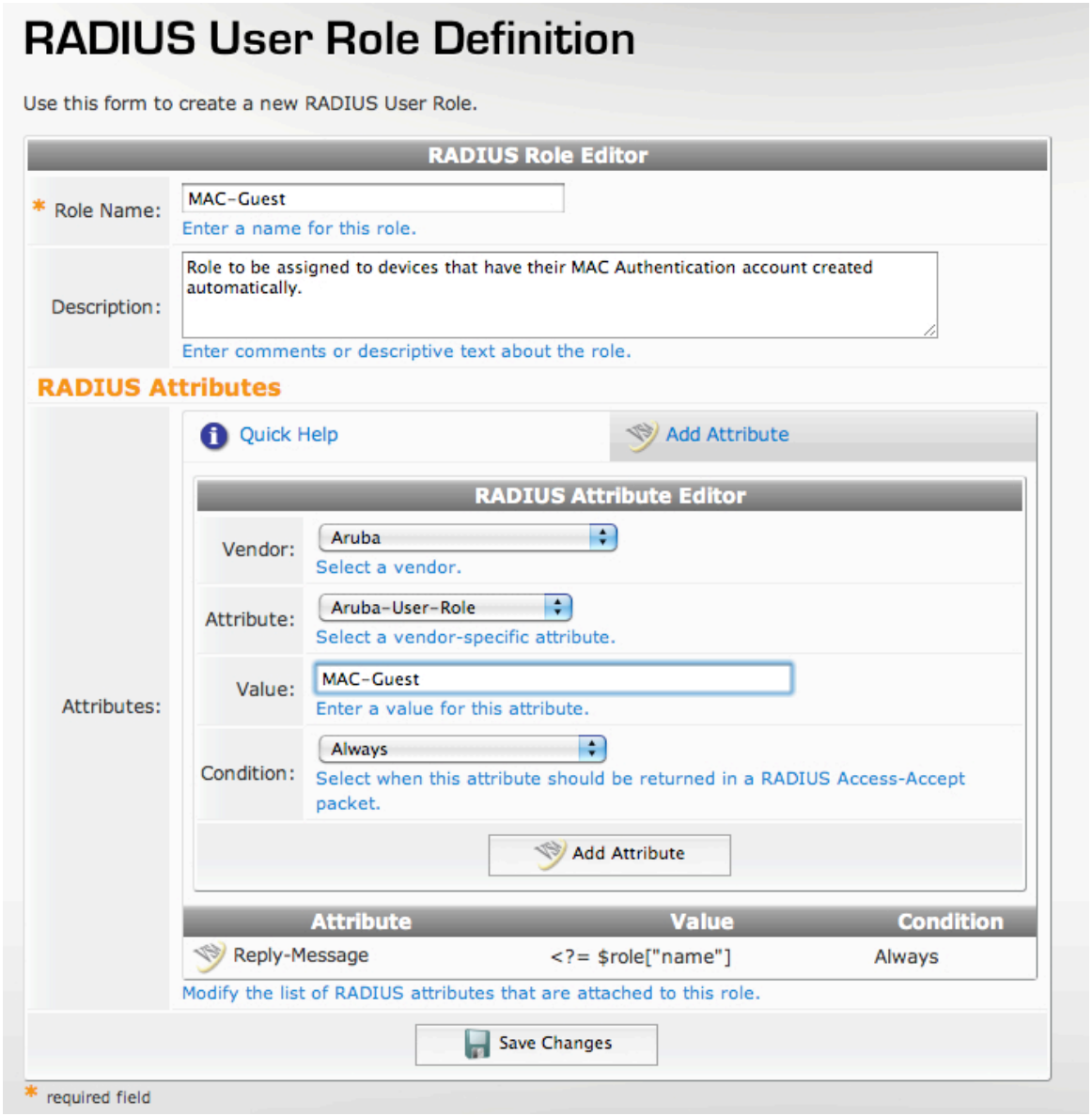


Figure 1. Sample RADIUS Role for MAC authenticated devices.

Once the changes to the new RADIUS Role have been saved, the current list of available Role will be listed as shown in Figure 2 below:

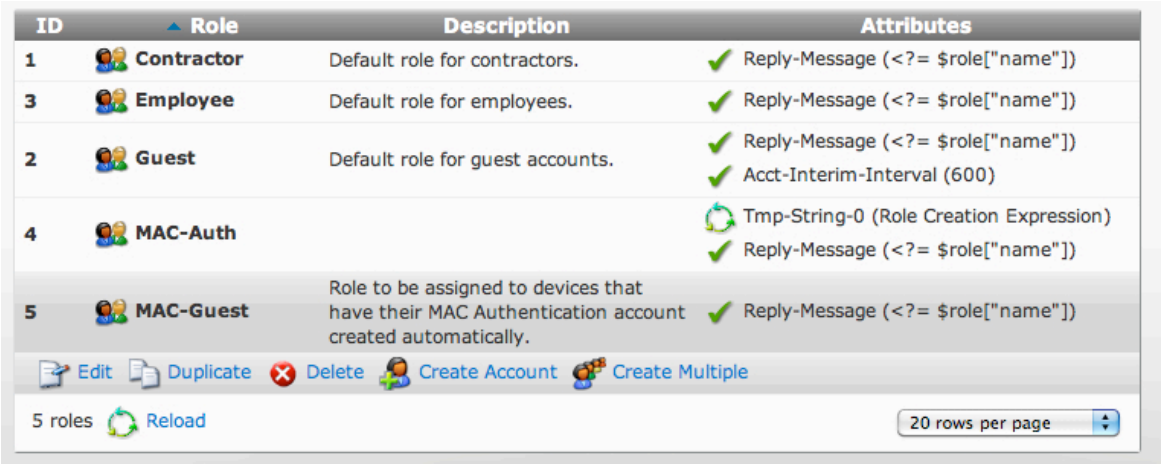


Figure 2. List of available RADIUS Roles.

Note the Role ID listed in the left most column of this table (in this case **role_id** 5) as this will be required for the next step and will be referenced in the condition expression configured to automatically create the MAC authentication account.

RADIUS Role to trigger MAC Address Account Creation

This Role should be assigned to all users that you wish to allow the automatic creation of MAC Authentication accounts. Depending on your deployment model, this maybe accounts that have been created through the Amigopod standard Guest Manager interface or potentially authenticated via external Authentication server such as Active Directory or LDAP. Nonetheless for the automatic creation of the MAC account to happen, the user account needs to below to or be mapped to this RADIUS role described below.

Create new RADIUS Role for MAC Account Creation (Optional)

Create new RADIUS role to hold the logic for the automatic creation of the MAC account.

NOTE The logic to create the new account is held within a single attribute and can be either added to a brand new RADIUS Role or to an existing RADIUS Role that is in use within the existing deployment.



Figure 3. Sample RADIUS Role for MAC Account creation logic.

Add MAC Account Creation attribute

To automatically create the new MAC authentication account, a condition expression is used within a Null attribute. This conditional expression will call internal Amigopod libraries to create the MAC authentication account based on the received **Calling-Station-ID** in the RADIUS Authentication Request packet.

Attribute: Tmp-String-0

Value: Role creation expression (or some other descriptive string, the attribute won't be returned for authentication)

Condition: Enter condition expression...

Expression:

```
return empty($user['mac_auth'])
    && NwaDynamicLoad('NwaCreateUser')
    && NwaDynamicLoad('NwaNormalizeMacAddress')
    && ($mac=NwaNormalizeMacAddress(GetAttr('Calling-Station-Id')))
    && ((!empty($user['id']) &&  NwaCreateUser(array(
        'creator_accept_terms'=>1,
        'mac'=>$mac,
        'mac_auth'=>1,
        'role_id'=>5,
        'visitor_name'=>$user['username'],
        'mac_auth_pair'=>$user['id'],
        'auto_update_account'=>1)))
       || (empty($user['id']) && NwaCreateUser(array(
        'creator_accept_terms'=>1,
        'role_id'=>5,
        'mac'=>$mac,
        'mac_auth'=>1,
        'visitor_name'=>$user['username'],
        'sponsor_name'=>$user['username'],
        'modify_expire_time'=>'today 17:00',
        'do_expire'=>4,
        'auto_update_account'=>1)))
       )
    && 0;
```

Annotated Expression:

```
return
    // Not already a MAC device...
    empty($user['mac_auth'])
    // Required call to load a function.
    && NwaDynamicLoad('NwaCreateUser')
    // Required call to load a function.
    && NwaDynamicLoad('NwaNormalizeMacAddress'
    // All MACs need to be normalized.
    && ($mac=NwaNormalizeMacAddress(GetAttr('Calling-Station-Id')))
```

```
// We are caching the MAC for a local user account.
//  'id' only exists for local accounts
&& ((!empty($user['id']) &&  NwaCreateUser(array(
    // Required field to act as the confirmation.
    'creator_accept_terms'=>1,
    // The normalized MAC.
    'mac'=>$mac,
    // Flag as a MAC so it shows in List Devices.
    'mac_auth'=>1,
    // The role ID.  Use $user['role_id'] to use this same role.
    'role_id'=>5,
    // The original username
    'visitor_name'=>$user['username'],
    // Formally pair the two accounts.
    // Cross links between accounts in the GUI.  A number of data items synched
    'mac_auth_pair'=>$user['id'],
    // Expire on the current day at 5pm.  Exclude to match the base $user value.
    'modify_expire_time'=>'today 17:00',
    // Overwrite the account if it already exists.
    'auto_update_account'=>1)))
   // Authentication is via an external server
   || (empty($user['id']) && NwaCreateUser(array
    // Required field to act as the confirmation.
    'creator_accept_terms'=>1,
    // The normalized MAC.
    'mac'=>$mac,
    // Flag as a MAC so it shows in List Devices.
    'mac_auth'=>1,
    // The role ID.  Use $user['role_id'] to use this same role.
    'role_id'=>5,
    // The original username
    // Could use $user['displayname'] here or some other attribute returned from the server.
    'visitor_name'=>$user['username'],
    // Setup a sponsor name.
    // Sponsor names can be used for filtering accounts within operating logins.
    'sponsor_name'=>$user['username'],
    // Expire on the current day at 5pm
    'modify_expire_time'=>'today 17:00',
    // Set the account to be disconnected via RFC-3576 if necessary.
    'do_expire'=>4,
    // Overwrite the account if it already exists.
    'auto_update_account'=>1)))
   )
// Make sure the expression is never returned
&& 0;
```

NOTE The **role_id** value in this expression will need to match the Role ID of the RADIUS Role created in the previous step (**role_id** of 5 in this example). In cases where you want the role to be the same as the original, you can use **`$user['role_id']`** in lieu of the numeric value.

NOTE This conditional expression assumes that the MAC Authentication Profile configured in the Aruba Controller has the **Delimiter** set to **dash** and the **Case** to **upper**.

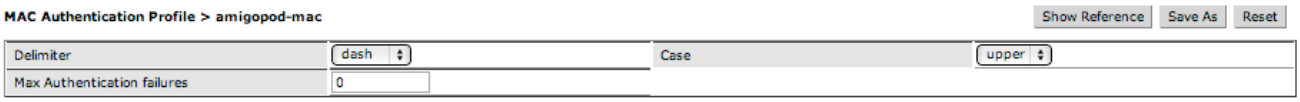


Figure 4. Suggested MAC Authentication Profile in ArubaOS configuration.

NOTE The logic is setup to populate the **visitor_name** field of the MAC Authentication account with the name of the user that authenticated during the authorization phase of this process. Depending on whether you are using the local Amigopod Guest Manager database or an external Active Directory database the **$user** attribute will have to be modified. For local Amigopod database accounts the value should be **$user['username']** and for Active Directory it should be **$user['displayname'].**

NOTE The sample code used is designed to create the MAC Authentication account and have it expire automatically at 5pm of the same day the user first authenticates via the Captive Portal process. The expiry time is set via the **modify_expire_time** attribute and should be customized to suit the deployment requirements at each site. Other common values include '**24h**' for 24 hours from the current time, '**this Friday 18:00**' for the end of the current week at 6pm. On expiry of the account, the RADIUS MAC authenticated session will be disconnected using RFC3576 and the account will be deleted. This configuration is triggered through the attribute **do_expire**. For more information on these Guest Manager attributes please refer to the Amigopod Deployment Guide.

The diagram in shows an example of the configuration of this Null Attribute.

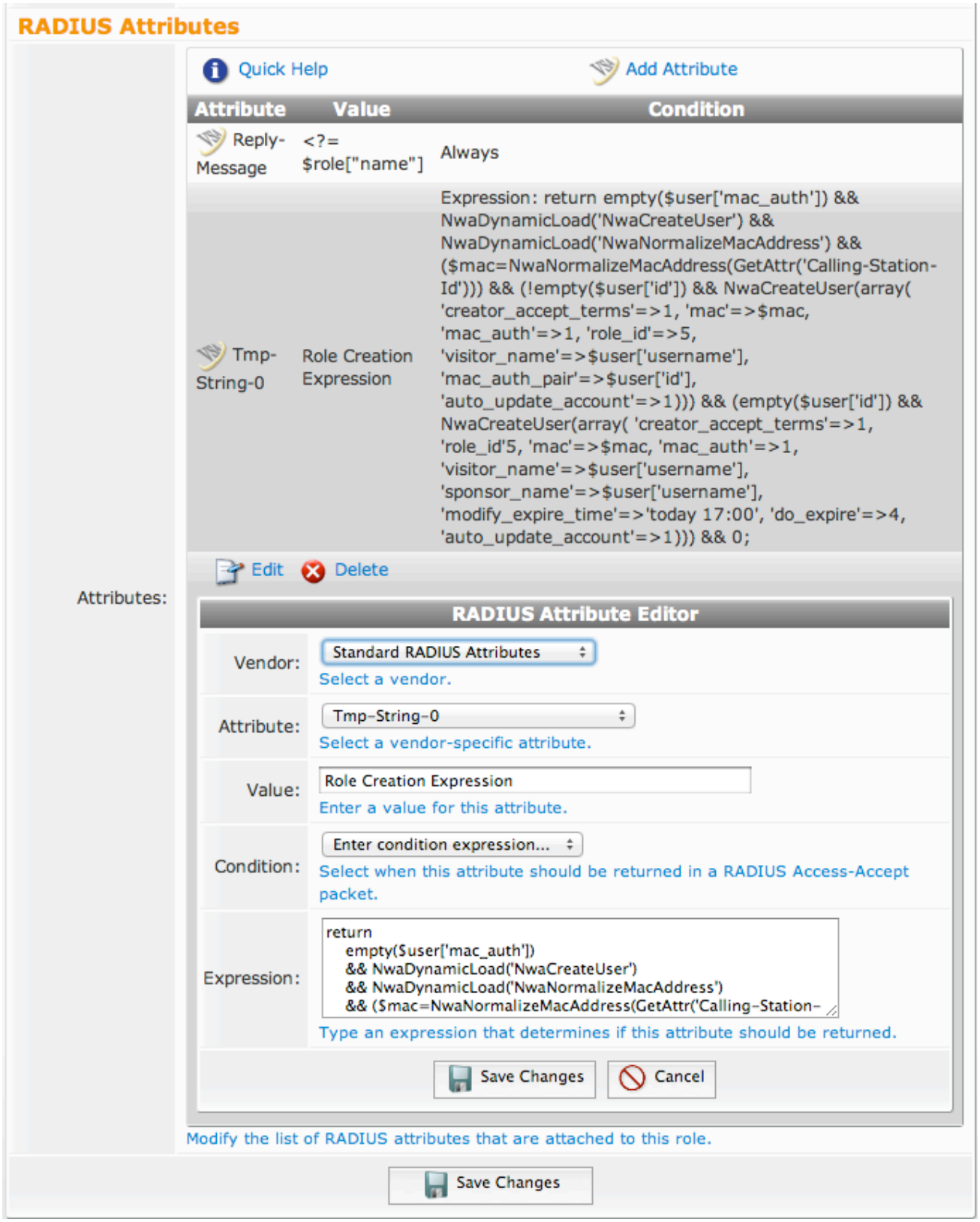


Figure 5. Sample of conditional expression used to create the MAC Authentication account.

Now that these two RADIUS roles have been configured the underlying logic is in place to support the authentication and automatic creation of MAC Authentication accounts.

Testing the Workflow

In order to test the workflow of the proposed design, there first needs to be an account in the local Amigopod Guest Manager database that is assigned to the **MAC-Auth** RADIUS Role created in the previous section.

Create Test Account

Navigate to the **Guests > Create Account** option to quickly create a test account and ensure that the **Role** selected is **MAC-Auth**.

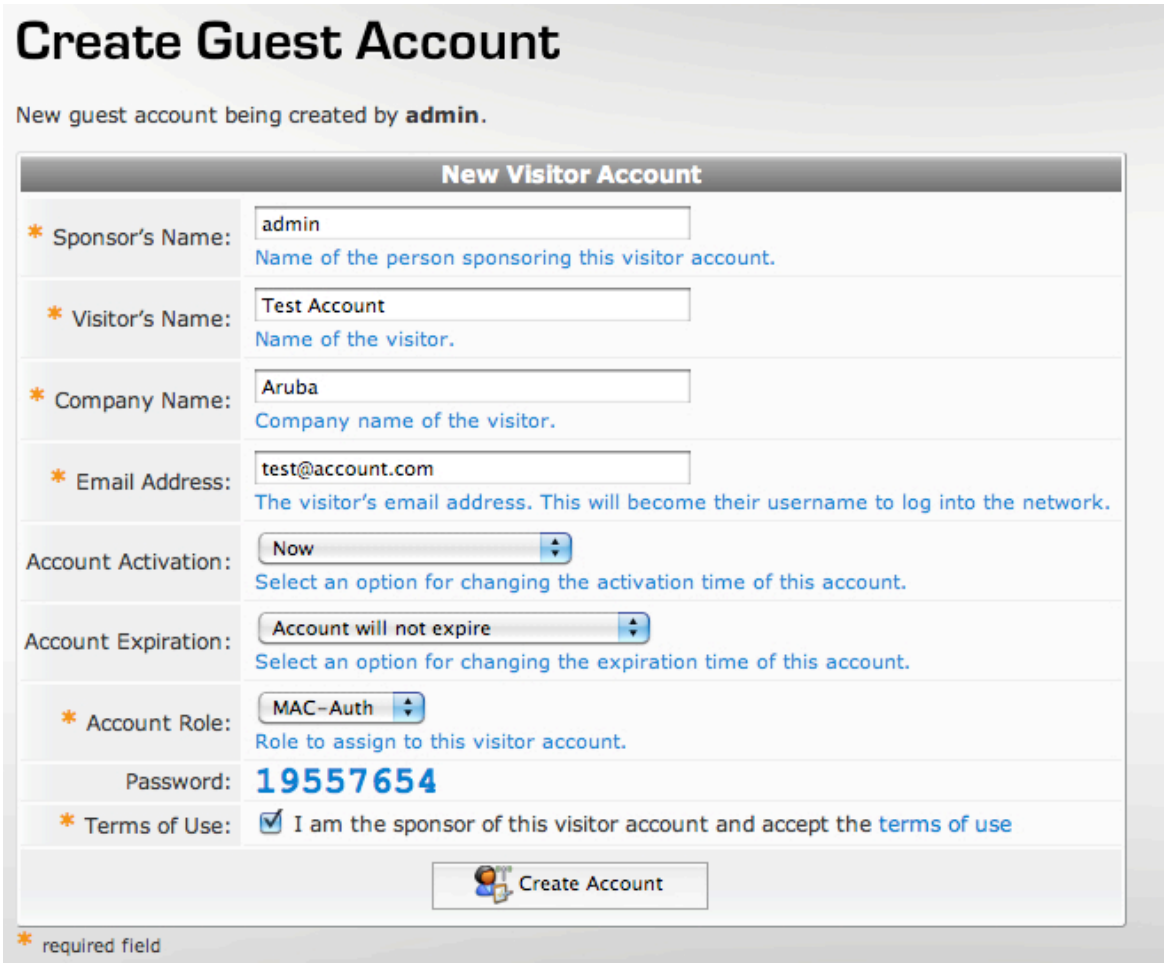


Figure 6. Sample Create Account for test user.

Returning to the List Accounts view, the newly created account is now visible and the role assignment to **MAC-Auth** can be verified as shown in Figure 7.

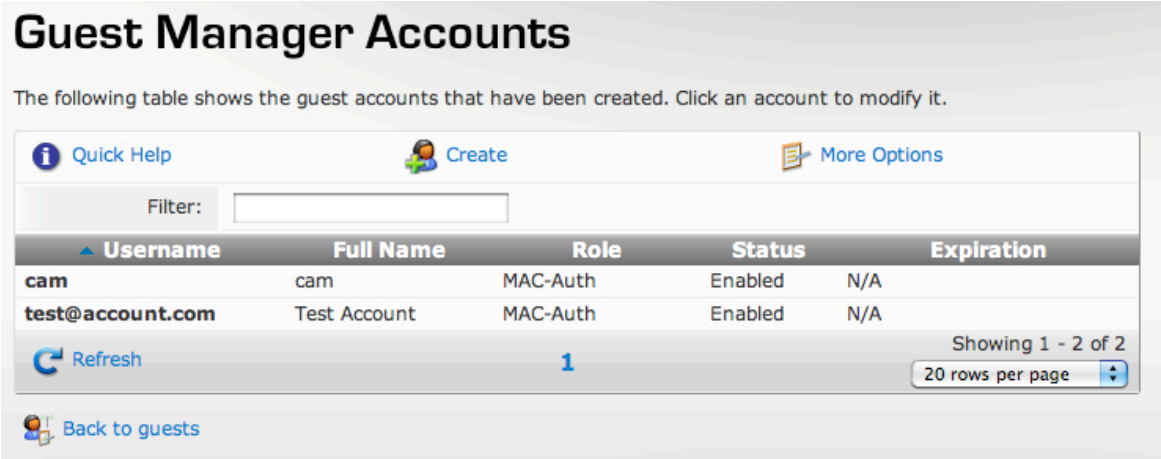


Figure 7. List Accounts view with new test account.

Initial Connection Attempt

From a test WiFi device, connect to the MAC Auth SSID. In this example the SSID is **Amigo-MAC-CP** to represent the MAC Authentication with Captive Portal failover configuration.

Navigate to **RADIUS** > **Server Control** page and confirm the initial failed attempt at RADIUS MAC Authentication in the tail of the RADIUS log displayed at the bottom of the page.

```
The most recent entries in the RADIUS server log file are shown below.

Thu Apr 7 17:39:03 2011 : Auth: Login incorrect: [00-26-BB-0C-42-75] (from client 651 port 0 cli 0026BB0C4275)
Thu Apr 7 16:39:00 2011 : Info: Ready to process requests.
Thu Apr 7 16:39:00 2011 : Info: rlm_extautz: compiled Jan 17 2011 17:20:34
Thu Apr 7 16:39:00 2011 : Info: rlm_sql (sql): Attempting to connect to amigopod@localhost:5432/amigopod
Thu Apr 7 16:39:00 2011 : Info: rlm_sql (sql): Driver rlm_sql_postgresql (module rlm_sql_postgresql) loaded and linked
Thu Apr 7 16:39:00 2011 : Info: rlm_exec: Wait=yes but no output defined. Did you mean output=none?
Thu Apr 7 16:39:00 2011 : Info: Using deprecated naslist file. Support for this will go away soon.
```

Now that RADIUS MAC Authentication has failed, the configuration will Failover to the underlying configuration of the **AAA Profile**.

Given the configuration of the **Initial Role** in the **AAA Profile**, the Captive Portal profile will redirect any attempt to browse the internet to the amigopod hosted Web Login page.

By opening the test device web browser, the Internet session should be redirected to the Web Login page as shown in Figure 8 below.



Figure 8. Sample Web Login page hosted on Amigopod

Logging in with the Test Account created previously, the user will be granted access to the network initially based on the configuration of the **Captive Portal Profile** on the Aruba controller.

This RADIUS transaction can be confirmed by again reviewing the RADIUS log under the **RADIUS > Server Control** page as shown below:

```
The most recent entries in the RADIUS server log file are shown below.

Thu Apr 7 16:39:32 2011 : Auth: Login OK: [test@account.com] (from client 651 port 0 cli
0026BB0C4275)
Thu Apr 7 17:39:03 2011 : Auth: Login incorrect: [00-26-BB-0C-42-75] (from client 651 port
0 cli 0026BB0C4275)
Thu Apr 7 16:39:00 2011 : Info: Ready to process requests.
Thu Apr 7 16:39:00 2011 : Info: rlm_extautz: compiled Jan 17 2011 17:20:34
Thu Apr 7 16:39:00 2011 : Info: rlm_sql (sql): Attempting to connect to
amigopod@localhost:5432/amigopod
Thu Apr 7 16:39:00 2011 : Info: rlm_sql (sql): Driver rlm_sql_postgresql (module
rlm_sql_postgresql) loaded and linked
Thu Apr 7 16:39:00 2011 : Info: rlm_exec: Wait=yes but no output defined. Did you mean
output=none?
Thu Apr 7 16:39:00 2011 : Info: Using deprecated naslist file. Support for this will go
away soon.
```

Subsequent Connection Attempt

Assuming the test device has now logged out of the wireless network and attempts to reconnect to the same WiFi SSID of **Amigo-MAC-CP**, the user experience should now be transparent login based on the RADIUS MAC Authentication.

This can be confirmed by again navigating to the RADIUS > Server Control page to review the log entries as shown below.

The most recent entries in the RADIUS server log file are shown below.

The most recent entries in the RADIUS server log file are shown below.

```
Thu Apr 7 16:40:18 2011 : Auth: Login OK: [00-26-BB-0C-42-75] (from client 651 port 0 cli 0026BB0C4275)
Thu Apr 7 16:39:32 2011 : Auth: Login OK: [test@account.com] (from client 651 port 0 cli 0026BB0C4275)
Thu Apr 7 17:39:03 2011 : Auth: Login incorrect: [00-26-BB-0C-42-75] (from client 651 port 0 cli 0026BB0C4275)
Thu Apr 7 16:39:00 2011 : Info: Ready to process requests.
Thu Apr 7 16:39:00 2011 : Info: rlm_extautz: compiled Jan 17 2011 17:20:34
Thu Apr 7 16:39:00 2011 : Info: rlm_sql (sql): Attempting to connect to amigopod@localhost:5432/amigopod
Thu Apr 7 16:39:00 2011 : Info: rlm_sql (sql): Driver rlm_sql_postgresql (module rlm_sql_postgresql) loaded and linked
Thu Apr 7 16:39:00 2011 : Info: rlm_exec: Wait=yes but no output defined. Did you mean output=none?
Thu Apr 7 16:39:00 2011 : Info: Using deprecated naslist file. Support for this will go away soon.
```

This can also be confirmed by navigating to the Monitoring > Controller > Clients page within the ArubaOS GUI of the controller as shown in Figure 9 below.

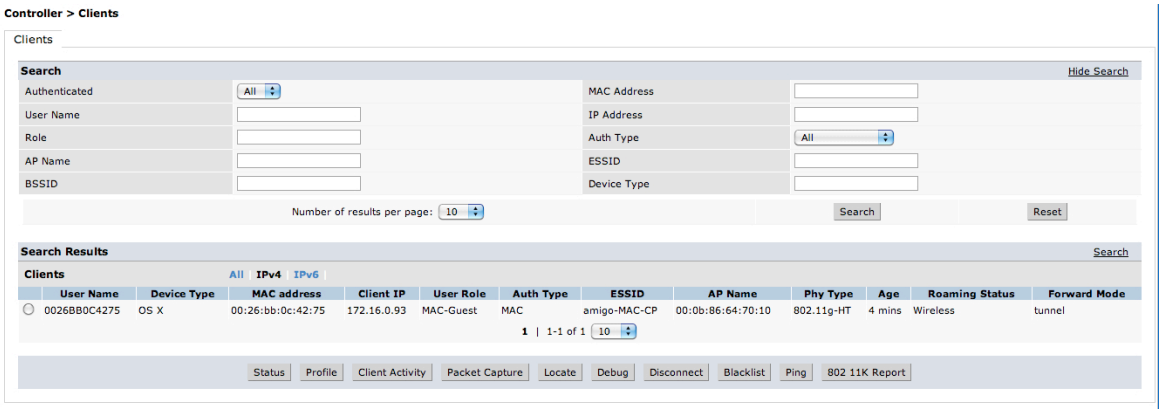


Figure 9. ArubaOS Controller Clients page confirming MAC Authentication & role assignment

3 Summary

In review this solution provides a businesses with a zero touch method of registering web enabled devices for transparent authentication moving forward. Once the device is first authenticated via the Captive Portal process, all subsequent authentications are transparent via the background RADIUS MAC Authentication.

The registration process is protected from open access by enforcing a authorization step where valid credentials must be provided during the initial Captive Portal authentication phase.

These credentials can be either a simple Guest Account created by a sponsor within the business or potentially leverage Active Directory integration in the authentication process to verify the existence of a corporate AD account before permitting the registration of the device. An extension to this design would be to leverage a Group Membership with AD to determine whether particular users are able to register their device for transparent access moving forward.

Although MAC based authentication is certainly not considered a robust security solution, this proposed design does provide a stepping-stone towards a more sophisticated Bring Your own Device management solution such as Aruba's MDAC strategy.

The Appendix provides an optional advanced extension to this solution where browser detection on the Amigopod can be leveraged to prevent employees attempting to register their regular laptop devices and only make the MAC registration process available for SmartPhones and Mobile Tablets alike.

4 Appendix – Browser Detection Extension

The following discussion provides an overview to how some advanced Amigopod configurations can be leveraged to prevent a non-mobile device from attempting to authenticate to this MAC registration SSID. The use case for this might be a business that wishes to provide its employees a method of self-provisioning their mobile device for transparent wireless access. That being said they still wish to ensure all existing laptops issued to employees continue to use the preferred 802.1x or other wireless security mechanism in place.

As the design stands so far the Aruba Controller is configured to redirect any unregistered devices through to an Amigopod standard Web Login page as shown in the configuration extract below:

```
aaa authentication captive-portal "amigopod-cp"
   server-group "amigopod-srv"
   redirect-pause 3
   no logout-popup-window
   protocol-http
   login-page "http://172.16.0.20/aruba_login.php"
```

This redirect URL can be changed to point towards a new Amigopod hosted Web Login that will perform browser detection and then determine whether the associated device should proceed to the Captive Portal authentication page for MAC registration or be redirected to an Error page informing the Laptop user to continue using the corporate sanctioned secured wireless network.

Pre-requisites

There is a need for the Amigopod deployment to have a **Blank Skin** provisioning. This can be confirmed by navigating to **Administrator > Plugin Manager > List Plugins** and scrolling to the bottom of the page to review the available Skin plugins.

If the Blank Skin plugin is not available contact your Aruba representative or channel partner for assistance.

Create Browser Redirect Page

Navigate to **Customization > Web Logins** and click on the **Create New** button.

Give the **Web Login** page a meaningful such as Browser Detection Redirect and then specify a **Page Name** that will form the basis of the URL hosted on Amigopod. For example, a **Page Name** of login_redirect would result in the URL of `http://<Amigopod IP or FQDN>/login_redirect.php`

This **Page Name** will then be referenced in the **Captive Portal Profile** on the Aruba Controller.

Ensure the **Vendor Settings** is set to Aruba Networks and the **Custom Form** option is checked.

Scrolling down the Web Login configuration, ensure the **Skin** entry is set to the **Blank Skin** discussed in the pre requisites section above.

In the **Header HTML**, remove the default contents and copy the following code to enable the browser detection and subsequent redirection to either the MAC Registration for mobile devices or Error Page for Laptops.

```
{if !$_wpl.browser.is_mobile}
  {php}
    header('Location: laptop_detect.php');
  {/php}
{else}
  {php}
    header('Location: aruba_login.php?' . $_SERVER['QUERY_STRING']);
  {/php}
{/if}
```

Create Laptop Detected Error Page

Navigate to **Customization > Web Logins** and click on the **Create New** button.

Give the **Web Login** page a meaningful such as Detected Laptop Error Page and then specify a **Page Name** that will form the basis of the URL hosted on Amigopod. For example, a **Page Name** of laptop_detect would result in the URL of `http://<Amigopod IP or FQDN>/laptop_detect.php`

This **Page Name** is referenced in the logic inserted in the **HTML Header** section of the previous **Web Login** page.

Ensure the **Vendor Settings** is set to Aruba Networks and the **Custom Login Form** option is checked.

Scrolling down the Web Login configuration, ensure the **Skin** entry is set to the Default Skin or the skin preferred for the presentation of the non-mobile device detected error messaging.

Modify the **Title** to something meaningful such as WiFi Policy.

In the **Header HTML** section, use regular HTML syntax to craft a message to users that have attempted to connect their laptop device to the MAC registration SSID. An example include syntax such as:

```
<H2>
Unauthorized Device detected
</H2>
<p>
Acme Corporation Web Usage Policy dictates that only employee
<strong>SmartPhone and Tablets</strong> can be registered for
transparent WiFi Access.
</p>
<p>
This Laptop must continue to use the secure corporate WiFi network known
as <strong>Corp-Secure</strong>. Please disconnect from this WiFi
network and reconnect to <strong>Corp-Secure</strong>.
</p>
<H4>
Your device details have been logged – do not attempt to connect to this
network again.
</H4>
```

Update the Captive Portal Profile

Back on the Aruba Controller, the Captive Portal Profile needs to be updated to reflect the new termination point for the initial redirect. Given the Page Name defined in the above step the new login page needs to be defined as `http://<Amigopod IP or FQDN>/login_redirect.php` as shown in the configuration extract below:

```
aaa authentication captive-portal "amigopod-cp"
   server-group "amigopod-srv"
   redirect-pause 3
   no logout-popup-window
   protocol-http
   login-page "http://172.16.0.20/login_redirect.php"
```

5 Appendix – Limiting Paired Devices

It may be desired to limit the number of MAC devices that are created and tied to a single user account. This can be accomplished with a change to the role expression.

```
return
    (
        ($MAX_MAC_ACCOUNTS = 2)
        && (NwaRadiusLocalServer()->GetUserCount(array(
            'sponsor_name' => strtolower(GetAttr('User-Name')),
            'delete_time' => 0,
            'mac_auth' => 1)
           ) >= $MAX_MAC_ACCOUNTS)
           ? (AccessReject() && 0) : 1
    )
    && empty($user['mac_auth'])
    && NwaDynamicLoad('NwaCreateUser')
    && NwaDynamicLoad('NwaNormalizeMacAddress')
    && ($mac=NwaNormalizeMacAddress(GetAttr('Calling-Station-Id')))
    && ((!empty($user['id']) &&  NwaCreateUser(array(
        'creator_accept_terms'=>1,
        'mac'=>$mac,
        'mac_auth'=>1,
        'role_id'=>5,
        'visitor_name'=>$user['username'],
        'sponsor_name'=>strtolower($user['username']),
        'mac_auth_pair'=>$user['id'],
        'auto_update_account'=>1)))
       || (empty($user['id']) && NwaCreateUser(array(
        'creator_accept_terms'=>1,
        'role_id'=>5,
        'mac'=>$mac,
        'mac_auth'=>1,
        'visitor_name'=>$user['username'],
        'sponsor_name'=>strtolower(GetAttr('User-Name')),
        'modify_expire_time'=>'today 17:00',
        'do_expire'=>4,
        'auto_update_account'=>1)))
       )
    && 0;
```

Annotated:

```
return
    // A logical block to put our expression.
    (
        // Number of devices per user allowed.
        ($MAX_MAC_ACCOUNTS = 2)
        // Search for existing accounts.
```

```
    && (NwaRadiusLocalServer()->GetUserCount(array(
        // sponsor_name is set to the username on create.
        'sponsor_name' => strtolower(GetAttr('User-Name')),
        // delete_time is 0 for valid accounts.
        'delete_time' => 0,
        // Only search for devices.
        'mac_auth' => 1)
        // Check that the returned count is greater than the allowed.
       ) >= $MAX_MAC_ACCOUNTS)
        // If it is, then the AccessReject()* will stop the rest of the expression.
        // If it is not, then the 1 will allow the expression to continue.
       ? (AccessReject() && 0) : 1
)
// The start of the original expression…
&& empty($user['mac_auth'])
&& …
```

* Note, the above example denies logins passed the limit. If you prefer to allow the credentials based login to continue, and simply limit the MAC accounts created, swap

```
? (AccessReject() && 0) : 1
```

for

```
? 0 : 1
```