Implement SiteMinder Authentication for Mobile Apps

Introduction

Product: CA SiteMinder®

This scenario describes how to implement SiteMinder authentication for mobile apps using the SiteMinder mobile authentication solution.

This Knowledge Base Article is subject to the following notices (see page 37), terms, and conditions.

How to Implement SiteMinder Authentication for Mobile Apps

The SiteMinder mobile application authentication solution enables you to leverage your existing SiteMinder infrastructure to secure mobile apps. This solution works by configuring a Web Agent or Secure Proxy Server to expose SiteMinder authentication operations as REST web services. Mobile apps can send requests to these services to do the following operations:

- Log in a user using Basic (username and password) authentication.
- Verify the status of a SiteMinder session.
- Log out a user.

The SiteMinder mobile application authentication solution includes the following kits:

Mobile Authentication iOS Source Kit

Provides a complete sample source code project that you can use to build a working sample native iOS app that demonstrates the use of SiteMinder for authentication. Mobile application developers can also use the source code as the basis for developing their own native mobile apps that use SiteMinder authentication.

Mobile Authentication Android Source Kit

Provides a complete sample source code project that you can use to build a working sample native Android app that demonstrates the use of SiteMinder for authentication. Mobile application developers can also use the source code as the basis for developing their own native mobile apps that use SiteMinder authentication.

Mobile Authentication Agent Kit

Provides resources to deploy on a Web Agent or Secure Proxy Server (SPS) to expose SiteMinder authentication operations as web services.

Note: The <stmdr> mobile application authentication solution is available for download from the Scripts and Tools section of the CA SiteMinder General Discussion message board in the Security Global User Community (Distributed).

How to implement SiteMinder authentication for mobile apps

To implement SiteMinder authentication for mobile apps, the application programmer, agent owner, and Policy Server administrator must collaborate to perform the following required processes:

- Application programmer does one of the following:
  - Application programmer uses the native iOS sample app to demonstrate or develop a mobile app that uses SiteMinder authentication (see page 3)
  - Application programmer uses the native Android sample app to demonstrate or develop a mobile app that uses SiteMinder authentication (see page 12)
- Agent owner configures an agent or Secure Proxy Server to support mobile authentication (see page 22)
- Policy administrator configures a security policy to support mobile authentication (see page 24)

Note: For background information, see the following multimedia resources:

- View a streaming video demonstrating the sample iOS app.
- View a streaming video http://caqrcode.atgte.am/SMAuthDemo.mov demonstrating the sample Android app.
- View a webcast about enabling SiteMinder authentication for mobile applications. (Free BrightTalk registration required.)

Application Programmer Uses the Sample App to Demonstrate or Develop a Native iOS App That Uses SiteMinder Authentication

The Mobile Authentication iOS Source Kit contains complete source code for a sample native iOS app that uses SiteMinder authentication. The sample app (AuthN) is written in Objective-C and supplied as a complete Xcode project. You can run AuthN to demonstrate SiteMinder mobile authentication functionality or use the source code to develop your own mobile app.

Note: You require at least Xcode 4 running on a supported MacOS system to complete the procedures in this process.

To demonstrate or develop your own mobile app that uses SiteMinder authentication, complete the following process:

1. Build and run the AuthN app using one of the following procedures:
   - Run the app using the integrated iOS Simulator (see page 4).
   - Run the app on a physical iOS device (see page 4) (requires an iOS Developer Program license).
2. Demo the AuthN app (see page 5).
3. Develop your own mobile app that uses SiteMinder authentication (see page 9).

Run the AuthN App Using the Xcode iOS Simulator

Run the Sample Mobile Authentication App (AuthN) from within Xcode using the iOS Simulator.

Note: To open the Xcode project to test the app, you require a MacOS system with the Xcode developer tools package installed.

Follow these steps:

1. Start Xcode.
2. Open the AuthnN app project, AuthN.xcodeproj.
3. If necessary, configure the iOS Simulator as the run destination in the project scheme using the following steps:
   a. Click Product, Edit Scheme.
   b. Select iOS Simulator in the Destination pop-up menu.
4. Click Run.

   The AuthN app is built and launched in an iOS Simulator window.

Run the AuthN App on a Physical iOS Device

Run the AuthN app on a physical iOS (5.0 or later) device.

Note: To open the Xcode project and test the app on a physical iOS device, you require a MacOS system with the Xcode developer tools package installed. You also require an iOS Developer Program license.

Follow these steps:

1. Start Xcode.
2. Open the AuthN app project, AuthN.xcodeproj.
3. In the AuthNproject Build Settings, specify your iOS Developer identity in the Code Signing Identity fields.
4. Provision your iOS device using the following steps:
   a. Connect your device to your Mac.

      The Devices Organizer pane opens.
   b. Select your device and click Use for Development.

5. Configure your iOS device as the run destination in the project scheme using the following steps:
   a. Click Product, Edit Scheme.
   b. Select your device in the Destination pop-up menu.
6. Verify that your iOS device is unlocked.
7. Click Run.

   The AuthN app is built and launched on your iOS device.

Demo the AuthN Native iOS App

After you start AuthN, use the app to demo SiteMinder mobile authentication functionality.

Note: The AuthN app is configured with default values that allow you to authenticate a valid sample user (jsmith) using a test server that is hosted by CA.

Follow these steps:

1. Start the Authn app.

   The app starts, showing the Login page.

2. To open the Configuration page, click the **Config** button.

By default, all fields are populated with default values to use AuthN with the SiteMinder mobile authentication solution test server that is hosted by CA.

Note: If the fields are not populated the first time that you run the application, click **Reset** and then **Save**.

To configure AuthN to use another SiteMinder-protected server (for instance, if you configure your own SiteMinder environment to support mobile authentication), change these values.

Show BizPage

Specifies what AuthN displays upon successful authentication. If set ON, AuthN displays the Business Logic page. If set to OFF, detailed trace information is displayed in the status area at the bottom of the Login page.

Default: ON

Target

Specifies the web page that AuthN displays in the Business Logic page upon successful authentication.

Default: authn:http://www.ca.com

Note: authn is a custom URI handler which allows the target to be displayed in the mobile browser (Safari).

LoginID

Specifies the default user identity for authentication.

Default: jsmith

Server

Specifies the address of the web server that hosts the SiteMinder mobile authentication services. The default server address accesses the CA-hosted mobile authentication services using the Mini-cookies session scheme.

Default: http://minicookie.forwardinc.mobi

To access the CA-hosted mobile authentication services using the SSL_ID session scheme, change this value to "http://spstest.forwardinc.mobi."

Login

Specifies the pathname of the SiteMinder mobile authentication Login service on the web server specified in the Server field.

Default: sma/loginonget.fcc

Logout

Specifies the pathname of the SiteMinder mobile authentication Logout service on the web server specified in the Server field.

Default: sma/logout.fcc

Session

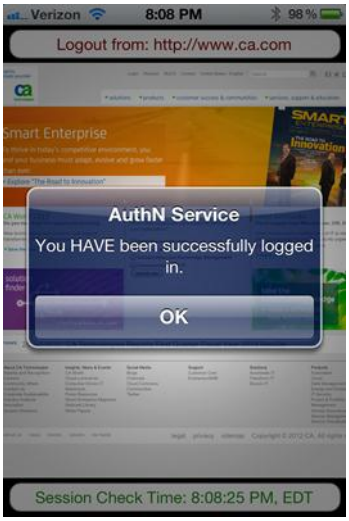
Specifies the pathname of the SiteMinder mobile authentication Session Status service on the web server specified in the Server field.

Default: /status.asp

Click **Clear** to clear all fields. Click **Reset** to revert all fields to their default values. Click **Save** to commit changes.

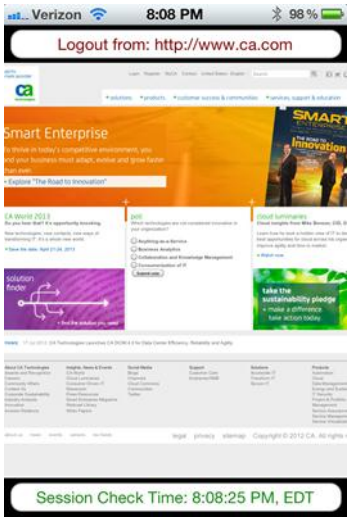
3. To return to the Login page, click the **Login** button.
4. Accept the default UserID and Password for jsmith and click **Login**.

   A login success dialog appears.



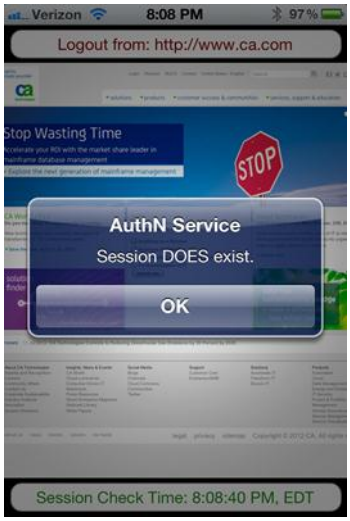
5. Click **OK**.

   The Business Logic page, which represents the app payload, is revealed. By default, AuthN displays the CA home page in an internal browser as an example. If you develop your own app, your business logic—for example, a static set of fields whose contents are driven by backend web services—would be displayed here.



6. To confirm that a valid SiteMinder session exists, click the **Session Check** field at the bottom of the page.

   A Session DOES exist dialog appears.

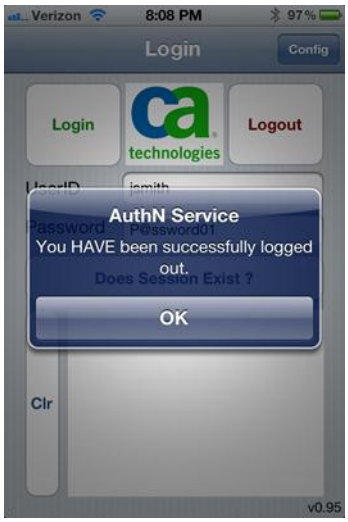


   Note: If a valid session does not exist, the Login page is displayed.

7. Click **OK**.

8. Click the **Logout** field at the top of the page.

   The user is logged out and a logout success dialog appears.



Get Started Developing Your Own Mobile iOS App That Uses SiteMinder Authentication

The AuthN iOS sample source code includes two classes (AuthNService and AuthNContext) that comprise a simple SiteMinder mobile authentication SDK. The storyboard and other classes demonstrate how to use AuthNService and AuthNContext to authenticate users in a simple mobile app. Use the AuthN source as the starting point to create your own app that uses SiteMinder authentication.

Note: This section is not intended to provide a detailed procedure for developing your native iOS app. It is intended only to point you in the right direction.

To start developing your own mobile app that uses SiteMinder authentication, follow this process:

1. Review the AuthN iOS app source code project in Xcode (see page 9).
2. Review the classes in the AuthN iOS source code (see page 11).

Review the AuthN iOS App Source Code Project in Xcode

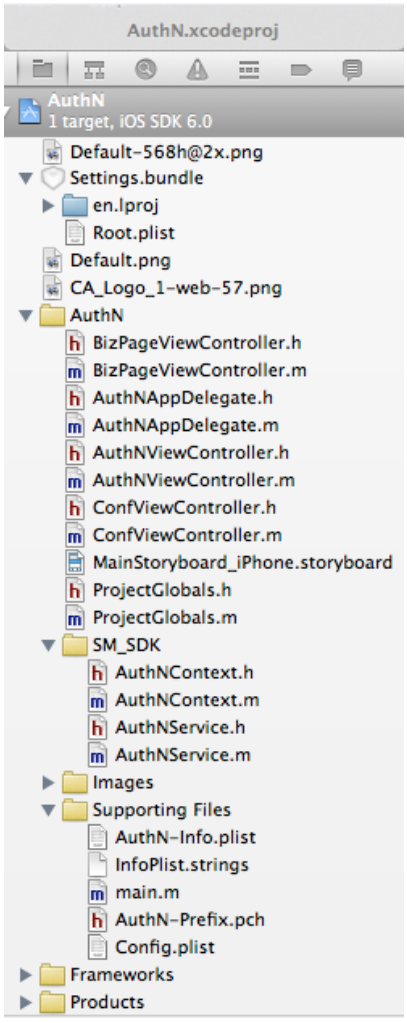
The AuthN app is a native iOS app written in Objective-C using the Xcode IDE. Review the source to see how it uses the AuthNService and AuthNContext classes to implement SiteMinder authentication requests.

Note: To use Xcode to manipulate the AuthN source or develop your own app, you require an iOS Developer Program license.

Follow these steps:

1. Start Xcode.
2. Open the Sample Mobile Authentication App project, AuthN.xcodeproj.

   The project opens and its contents are displayed in the Xcode project navigator as shown in the following screenshot.



3. Review the source and storyboard files.

Review the Classes in the AuthN iOS Source Code

This section describes the classes that implement the AuthN app. Review these classes in the AuthN source code to see how the app interacts with the SiteMinder authentication services.

main

Instantiates AuthNAppDelegate to start the app.

AuthNAppDelegate

Instantiates the app window, does any up-front setup, and then displays the Login page. AuthNAppDelegate calls AuthNViewController after it is done setting up the environment.

AuthNViewController

Displays the credential collector panel (username/password) on the Login page. Executes calls to the SiteMinder authentication web services using AuthNService when the Login, Logout, and Session Check buttons are pressed.

Note: In particular, see the button click methods, loginPressed(), logoutPressed(), and isSessionPressed, to see how to use AuthNService to send requests to the Login, Logout, and Status services.

AuthNService

Generates correctly formatted HTTP REST requests to the SiteMinder authentication services. Takes information that is provided to its AuthNContext instance to determine which service to call and the values to pass.

AuthNContext

Provides the contextual information that AuthNService requires to make an authentication service call. Information such as service URL, query parameters, headers, and how to process cookies are set on the AuthNContext instance. The AuthNContext instance is obtained by asking the AuthNService for it. Once you have it, you can set the values that are requires to define how to execute the REST call to the authentication service. When the call is complete, the results are returned in the AuthNContext instance.

BizPageViewController

If the Show Biz Page configuration setting is set to ON in the Configuration page, displays the Business Logic page after a successful authentication. The web page that is displayed to represent business logic is defined by the value of the Target configuration setting. In your custom app, BizPageViewController would display your business logic, for example, standard iOS fields using backend web services to obtain the content.

ConfViewController

Displays the Configuration page within the app. Configuration values that are entered on the page are stored in settings.bundle/Root.plist and the iOS NSUserDefaults storage location.

Note: The configuration values can also be seen and manipulated in the traditional iOS settings page for the AuthN app.

ProjectGlobals

A convenience class that provides access to the various configuration settings (server url, login page, and so on).

Application Programmer Uses the Sample App to Demonstrate or Develop a Native Android App That Uses SiteMinder Authentication

The Mobile Authentication Android Source Kit contains complete source code for a sample native Android app that uses SiteMinder authentication. The sample app (AuthN) is written in Java and supplied as a complete Eclipse project. You can run AuthN to demonstrate SiteMinder mobile authentication functionality or use the source code to develop your own mobile app.

Note: The sample source code is intended for Android 4.x. You require the latest versions of Eclipse IDE for Java Developers and the Android SDK, and JDK 6 to complete the procedures in this process. If you install the ADT bundle from Google, verify that it contains the requisite Eclipse IDE and Android SDK.

To demonstrate or develop your own mobile app that uses SiteMinder authentication, complete the following process:

1. Build and run the AuthN app using one of the following procedures:
   - Run the app using the Eclipse Android Virtual Device (see page 12).
   - Run the app on a physical Android device (see page 13).
2. Demo the AuthN Android app (see page 14).
3. Develop your own mobile Android app that uses SiteMinder authentication (see page 19).

Run the AuthN App Using the Eclipse Android Virtual Device Simulator

Run the Sample Mobile Authentication App (AuthN) from within Eclipse using the Android Virtual Device.

Note: To open the Android project to test the app, you require a system on which Eclipse and the Android SDK are installed

Follow these steps:

1. Start Eclipse.
2. Import a project from the File menu.
3. Browse to and select the AuthN project folder.
4. If necessary, create a new Android Virtual Device using the following steps:
   a. Select Android Virtual Device Manager from the window drop-down list.
   b. Click the New button if no Android Virtual Device exists that is at least 2.3.3.
   c. In the AVD Name field, enter a custom name (any name you want).
   d. From the Device drop-down list, select "Galaxy Nexus" or any other screen size.
   e. In the target drop-down list, ensure that a version higher than 2.3.3 is selected.
   f. In the SD Card size text field enter "512" and ensure MiB is selected.
5. Click Run.

The AuthN app is built and launched in an Android Virtual Device window.

Note: If this is your first time creating an Android Virtual Device, this step can take several minutes.

Run the AuthN App on a Physical Android Device

Run the AuthN app on a physical Android (2.3.3 or later) device.

Note: To open the Android project and test the app on a physical Android device, you require a system with Eclipse and the Android SDK installed.

Follow these steps:

1. Start Eclipse.
2. Import a project from the File menu.
3. Browse to and select the AuthN project folder.
4. Connect your Android to a USB port.
5. Verify that your Android device is unlocked.
6. Place your android device in "Developer" mode.
7. Go to the Settings.

8. (Android versions below 4.0 ONLY) Go to Applications.

   Note: To run the AuthN app on Android versions below 4.0, make code changes n the AuthN source code to enable certificates. For more information, see Edit the AuthN App Source Code to Run on Android Versions Below 4.0 (see page 21)

9. Select Development.
10. Verify that the **USB Debugging** option is set.
11. If you do not have a Nexus brand Android device, verify that the USB Drivers for your device are installed (Samsung Kies Agent, HTC Android USB Drivers, Motorola Android USB Drivers, and so on).
12. From the Run menu, select **Run Configurations**.
13. Select the **Android Application** item from the list.
14. Click the **New Launch Configuration** icon in the upper left corner.
15. Enter a name for the configuration (for example, “SMAndroidAuthLaunch”) in the **Name** field.
16. Click the **Browse** button and select the **LoginActivity** for the AuthN project.
17. Select the **Target** tab.
18. Verify that the **Always prompt to pick device** option is set.
19. Click **Apply**.
20. Click **Run**.
21. In the popup window, select your physical Android device and click **OK.**

The AuthN app is built and launched on your Android device.

Demo the AuthN Android App

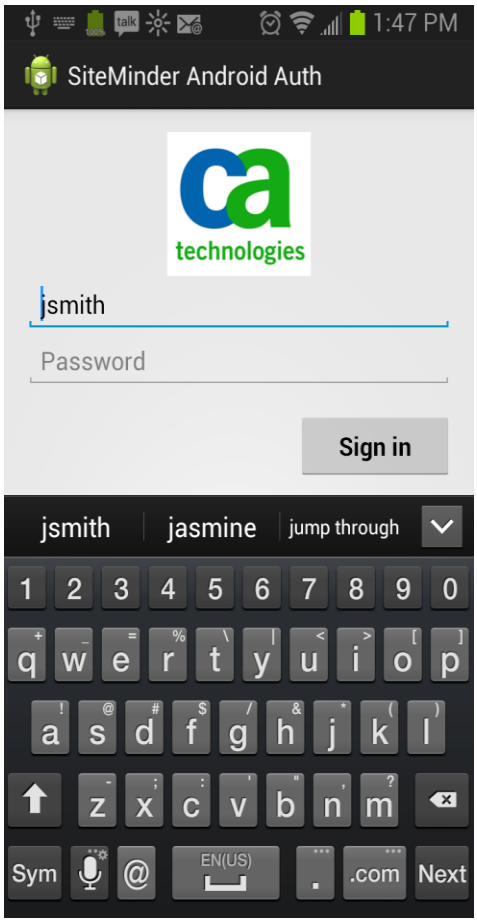
After you start AuthN, use the app to demo SiteMinder mobile authentication functionality.

Note: The AuthNn app is configured with default values that allow you to authenticate a valid sample user (jsmith) using a test server that is hosted by CA.

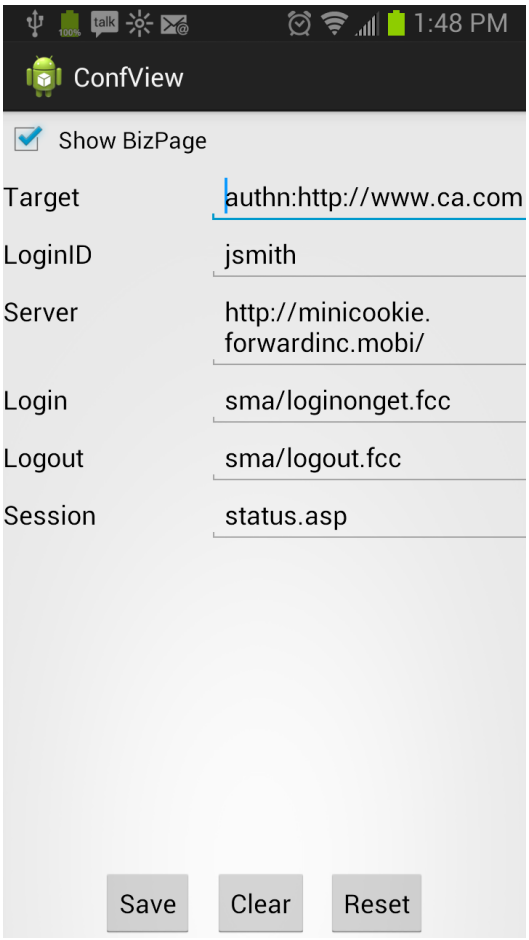
Follow these steps:

1. Start the Authn app.

   The app starts, showing the Login page.



2. To open the Configuration page, press the Android menu button and then press Config.



By default, the fields are not populated. Press the reset button to populate all the fields with default values to use AuthN with the SiteMinder mobile authentication solution test server that is hosted by CA.

Note: If the fields are not populated the first time that you run the application, click Reset and then Save.

To configure AuthN to use another SiteMinder-protected server (for example, if you configure your own SiteMinder environment to support mobile authentication), change these values.

Show BizPage

Specifies what AuthN displays upon successful authentication. If set ON, AuthN displays the Business Logic page. If set to OFF, detailed trace information is displayed in the status area at the bottom of the Login page.

Default: ON

Target

Specifies the web page that AuthN displays in the Business Logic page upon successful authentication.

Default: authn:http://www.ca.com

Note: authn is a custom URI handler which allows the target to be displayed in the default browser (Chrome or Android Browser).

LoginID

Specifies the default user identity for authentication.

Default: jsmith

Server

Specifies the address of the web server that hosts the SiteMinder mobile authentication services. The default server address accesses the CA-hosted mobile authentication services using the Mini-cookies session scheme.

Default: http://minicookie.forwardinc.mobi

To access the CA-hosted mobile authentication services using the SSL_ID session scheme, change this value to "http://spstest.forwardinc.mobi."

Login

Specifies the pathname of the SiteMinder mobile authentication Login service on the web server that is specified in the Server field.

Default: sma/loginonget.fcc

Logout

Specifies the pathname of the SiteMinder mobile authentication Logout service on the web server that is specified in the Server field.

Default: sma/logout.fcc

Session

Specifies the pathname of the SiteMinder mobile authentication Session Status service on the web server that is specified in the Server field.

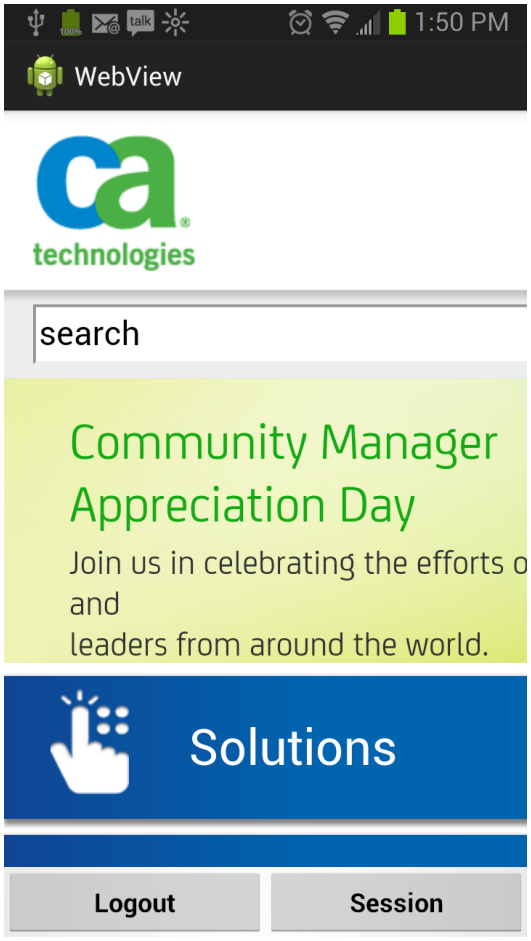
Default: /status.asp

Click **Clear** to clear all fields. Click **Reset** to revert all fields to their default values. Click **Save** to commit changes.

3. To return to the Login page, click the **Sign In** button.

4. Enter "P@ssword01" in the **Password** field.
5. Press the **Login** button.

   A login success dialog appears.



   The Business Logic page, which represents the app payload, is revealed. By default, AuthN displays the CA home page in an internal browser as an example. If you develop your own app, your business logic—for example, a static set of fields whose contents are driven by backend web services—would be displayed here.

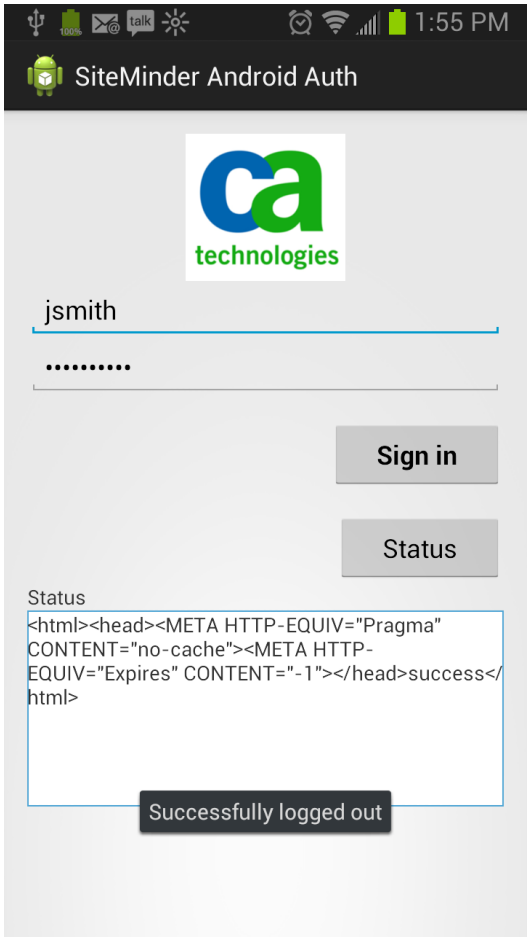
6. To confirm that a valid SiteMinder session exists, click the Session field at the bottom of the page.

   An "Authenticated" message appears and disappears on the screen

   Note: If a valid session does not exist, the message "Not Authenticated" will appear.

7. Click the **Logout** button at the bottom left of the page.

   The user is logged out and a logout success message appears.



Get Started Developing Your Own Mobile Android App That Uses SiteMinder Authentication

The AuthN sample source code includes four classes (AuthComm, ConfView, LoginActivity, and WvActivity) that comprise a simple SiteMinder mobile authentication SDK. The storyboard and other classes demonstrate how to use AuthComm and WvActivity to authenticate users in a simple mobile app. Use the AuthN source as the starting point to create your own app that uses SiteMinder authentication.

Note: This section is not intended to provide a detailed procedure for developing your native iOS app. It is intended only to point you in the right direction.

To start developing your own mobile app that uses SiteMinder authentication, follow this process:

1. Review the AuthN app source code project in Eclipse (see page 20).
2. Review the classes in the AuthN source code (see page 21).

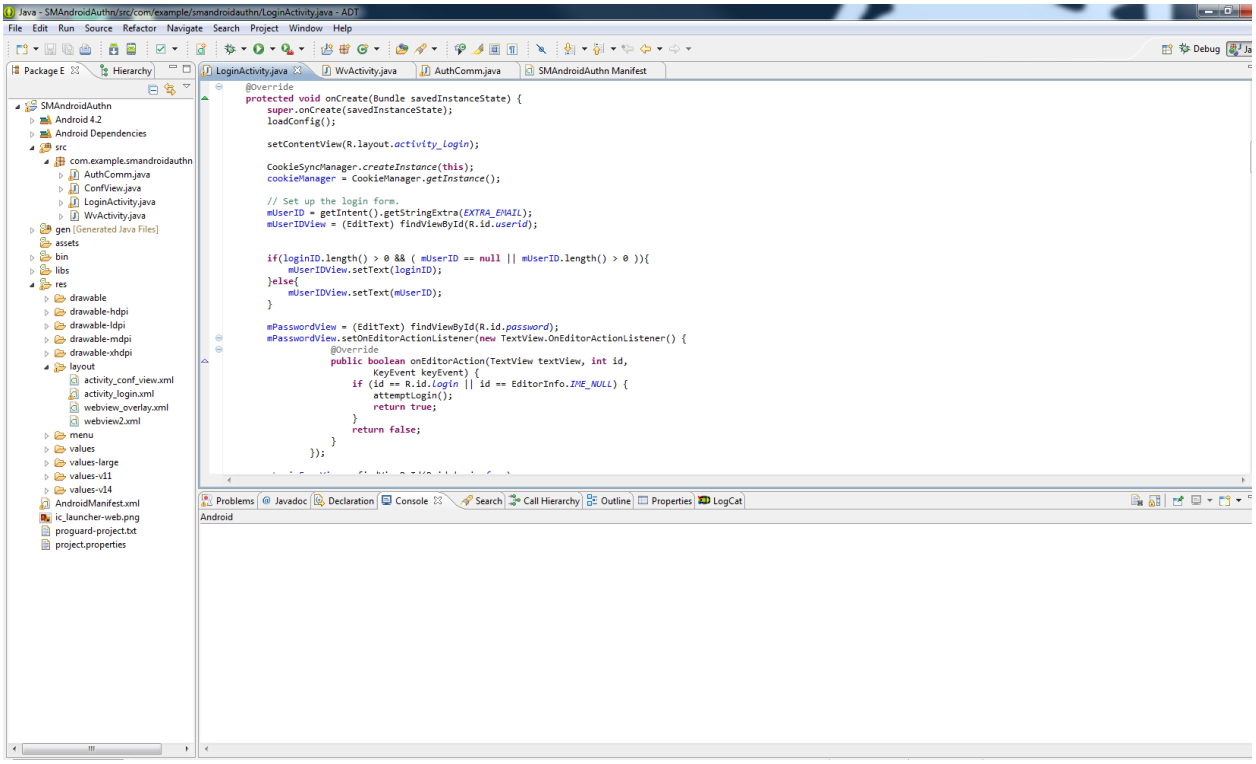
Review the AuthN Android App Source Code Project in Eclipse

The AuthN App is a native Android app that is written in Java using the Eclipse IDE. Review the source to see how AuthNuses the AutheComm and WvActivity classes to implement SiteMinder authentication requests.

Follow these steps:

1. Start Eclipse.
2. Open the Sample Mobile Authentication App project or import the project from the folder.

   The project opens and its contents are displayed in the Eclipse project navigator as shown in the following screenshot.



3. Review the source and storyboard files.

Review the Classes in the AuthN Android Source Code

This section describes the classes that implement the AuthN app. Review these classes in the AuthN source code to see how the app interacts with the SiteMinder authentication services.

LoginActivity

Launches and initializes the app and displays the main login page.

Displays the credential collector panel (username/password) on the Login page. Executes calls to the SiteMinder authentication web services using AuthNService when the Login, and Status buttons are pressed.

AuthComm

Generates correctly formatted HTTP REST requests to the SiteMinder authentication services. Takes information that is provided to it to determine which service to call and the values to pass.

WvActivity

If the Show Biz Page configuration setting is set to ON in the Configuration page, displays the Business Logic page after a successful authentication. The web page that is displayed to represent business logic is defined by the value of the Target configuration setting. In your custom app, BizPageViewController would display your business logic, for example, standard Android fields using backend web services to obtain the content.

ConfView

Displays the Configuration page within the app. Configuration values that are entered on the page are stored in the App config bundle.

Run the AuthN App on Android Versions Below 4.0

To run the AuthN app on Android versions below 4.0, edit the AuthN source code to enable certificates (certificates are enabled automatically in Android 4.0 and above).

Follow these steps:

1. Open AuthComm.java in the Eclipse Java editor.
2. Locate and uncomment the code that encapsulates the InsecureSSLSocketFactory class (near the top of the file).

   Important! The InsecureSSLSocketFactory class in AuthComm.java is not secure — it is intended for testing purposes only. If you develop your own app, replace InsecureSSLSocketFactory with a certificate checking function that uses your own generated certificates.
3. Uncomment the code in the constructor for AuthComm.
4. Compile and run as normal.

Agent Owner Configures the Agent to Support Mobile Authentication

To configure a Web Agent to support mobile authentication for your own mobile app, the agent owner deploys the SiteMinder Mobile Authentication Agent Kit on a web server that supports Active Server Pages (ASP).

To reduce the bandwidth requirements of <stmdnr> session cookies, you can also deploy some of the files on a Secure Proxy Server and configure the Mini-cookies or SSL_ID session scheme.

The Mobile Authentication Agent Kit contains the following files:

- **loginonget.fcc**—Provides a login web service that a mobile app can access to make login requests.
- **loginonget.unauth**—Defines a required login failure message.
- **logout.fcc**—Provides a logout web service that a mobile app can access to make logout requests.
- **status.asp**—Provides a status service that a mobile app can access to obtain the status of a user session.
- **success.html**—Login success page. The agent redirects to this page if login is successful.
- **fail.html**—Login failure page. The agent redirects to this page if login is unsuccessful.

To configure an agent or Secure Proxy Server to support mobile authentication, do one of the following procedures:

- Configure a Web Agent to support mobile authentication (see page 22)
- Configure a Secure Proxy Server to support mobile authentication (see page 23)

Configure a Web Agent to Support Mobile Authentication

To configure a Web Agent to support mobile authentication, copy the files from the Mobile Authentication Agent Support Kit into the web server directory structure.

Note: The web server must support Active Server Pages (ASP).

Follow these steps:

1. Extract the Mobile Authentication Agent Support Kit archive into a temporary location.
2. Copy the following files from the temporary location to *web_server_root*/sma:
   - loginonget.fcc
   - loginonget.unauth
   - logout.fcc

   web_server_root

   Specifies the web server root directory.
3. Copy the following files from the temporary location to *web_server_root*:
   - status.asp
   - success.html
   - fail.html
4. Configure Secure Sockets Layer (SSL) security on the website that hosts the Mobile Authentication Agent Kit. Otherwise, user credentials are transmitted in clear text over an unsecured connection.
5. Restart the SiteMinder Agent.
6. Supply the host address for mobile authentication requests to the application programmer developing your mobile app.

Configure a Secure Proxy Server to Support Mobile Authentication

With tiered bandwidth structures for mobile data services being the norm, conserving every bit of available bandwidth is paramount to controlling overall costs. Traditional SiteMinder session (SMSESSION) cookies add approximately 2 KB to the HTTP payload each time a user requests a protected resource.

The SiteMinder Secure Proxy Server provides the following alternative session schemes that require less bandwidth than traditional SiteMinder session cookies:

Mini-cookie

The Mini-cookie session scheme reduces the HTTP payload by using only a 10-byte cookie. The original full sized cookie is stored in the Secure Proxy Server internal session store. This mini-cookie serves as an artifact to link the mini-cookie to the full sized cookie.

SSL_ID

The SSL_ID session scheme places no cookie on the client device at all. The unique identifier that is generated at the time of the SSL handshake serves as the identifier for the session. SSL_ID is the most secure option, but also has the most limitations.

Follow these steps:

1. Do the following steps on a web server:
   a. Extract the Mobile Authentication Agent Support Kit archive into a temporary location.
   b. Copy the following files from the temporary location to *web_server_root:*
      - status.asp
      - success.html
      - fail.html

      web_server_root

      Specifies the web server root directory.
2. Do the following steps on the Secure Proxy Server:
   a. Configure a virtual host for mobile authentication requests that specifies the Mini-cookie or SSL_ID session scheme.
   b. Extract the Mobile Authentication Agent Support Kit archive into a temporary location on the SPS.
   c. Copy the following files from the temporary location to *sps_home*/sma:
      - loginonget.fcc
      - loginonget.unauth
      - logout.fcc

      sps_home

      Specifies the Secure Proxy Server home directory.
      - Configure a proxy rule that forwards requests that are sent to the virtual host that you configured in step 2a to the web server that hosts status.asp, success.html, and fail.html.
3. Supply the virtual host address for mobile authentication requests to the application programmer developing your mobile app.

Policy Administrator Configures a Security Policy to Support Mobile Authentication

The policy administrator configures a security policy to support mobile authentication by defining an application object.

Note: You can also configure the required security policy using domain objects (realms, rules, and so on).

To configure the security policy, use the following process:

1. Create an application object to define the security policy for the mobile app (see page 25)
2. Configure resources for mobile authentication (see page 27)
3. Create roles that identify the users that can access the mobile app (see page 32)
4. Configure responses to associate with authentication success and failure events (see page 33)
5. Configure the policy for mobile authentication (see page 35)

Create an Application Object to Define the Security Policy for the Mobile Authentication Solution

Create an application object and configure the following general properties of the security policy to support the mobile authentication solution.

Follow these steps:

1. Log in to the SiteMinder Administrative UI.
2. Click Policies, Application.
3. Click Applications.
4. Click Create Application.

   The Create Application page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

5. Enter the name and description of the application using the following values:

   Name

   Mobile Authentication Security Policy

   Description

   Allows mobile applications to access the login service using REST calls.

6. In the Components section, define general security requirements for the mobile authentication resources. Follow these steps:

   a. Click Create Component.

   b. Enter "Mobile Authentication" in the Name field.

   c. Click Lookup Agent/Agent Group.

   d. Select the agent that protects that web server on which the authentication web services are exposed.

e. Enter the following values:

Resource Filter

/login

Default Resource Protection

protected

Authentication Scheme

Basic

7. In the User Directories section, perform the following steps to select the directory (or directories) of users who are authorized to use the mobile application:
   a. Click Add/Remove.
   b. Select one or more user directories from the list of Available Members, and click the right-facing arrows.

      The user directories are removed from the list of Available Members and added to the list of Selected Members.

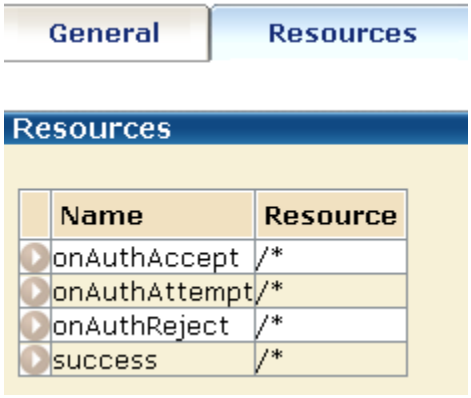
      Note: To select more than one member at one time, hold down the Ctrl key while you click the additional members. To select a block of members, click the first member and then hold down the Shift key while you click the last member in the block.
   c. Click OK.

      The selected user directories are listed under User Directories on the Create Application page.
8. Click OK.

Configure Resources for Mobile Authentication

Configure application resources to protect your mobile app and to configure the policy to handle authentication attempt, acceptance, and failure events. Configure the following resources:

- onAuthAccept (see page 27)
- onAuthAttempt (see page 28)
- onAuthReject (see page 30)
- success (see page 31)



Designate a Resource for Mobile Authentication Accept Events

To specify that the security policy handles mobile app authentication acceptance events, configure a resource. The policy binds a response to this resource to determine how to handle the event.

Follow these steps:

1. On the Create Application page, click the Resources tab.
2. Click Create.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Specify the following resource properties:

   Name

   onAuthAccept

   Resource

   /*

   Action

   Authentication events: OnAuthAccept

4. Click OK

   The resource is created.

Designate a Resource for Mobile Authentication Attempt Events

To specify that the security policy handles mobile app authentication attempt events, configure a resource. The policy binds a response to this resource to determine how to handle the event.

Follow these steps:

1. On the Create Application page, click the Resources tab.
2. Click Create.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Specify the following resource properties:

   Name

   onAuthAttempt

   Resource

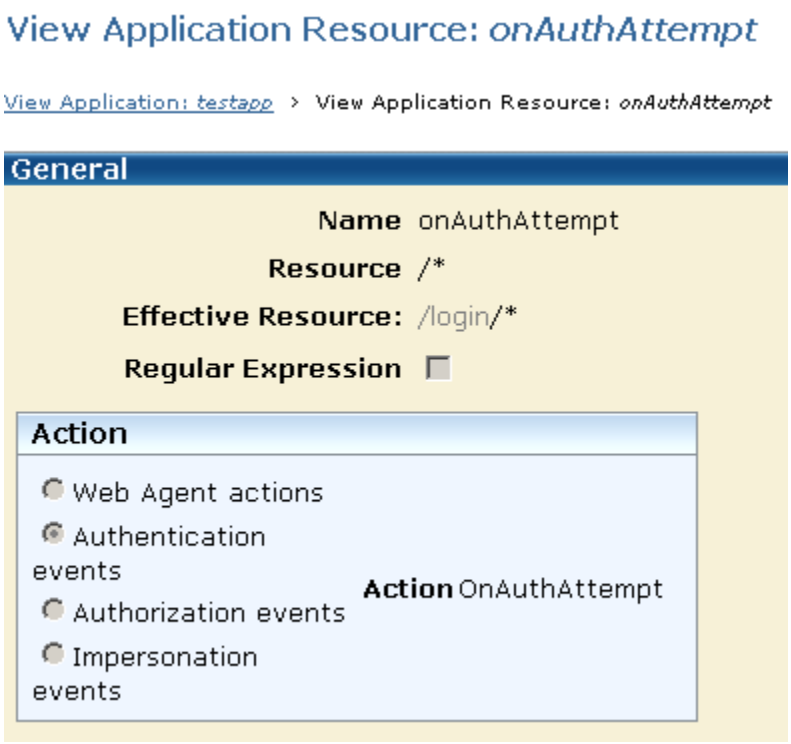
   /*

   Action

   Authentication events: OnAuthAttempt

4. Click OK

   The Resource is created.



Designate a Resource for Mobile Authentication Accept Reject Events

To specify that the security policy handles mobile app authentication reject events, configure a resource. The policy binds a response to this resource to determine how to handle the event.

Follow these steps:

1. On the Create Application page, click the Resources tab.
2. Click Create.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Specify the following resource properties:

   Name

   onAuthReject

   Resource

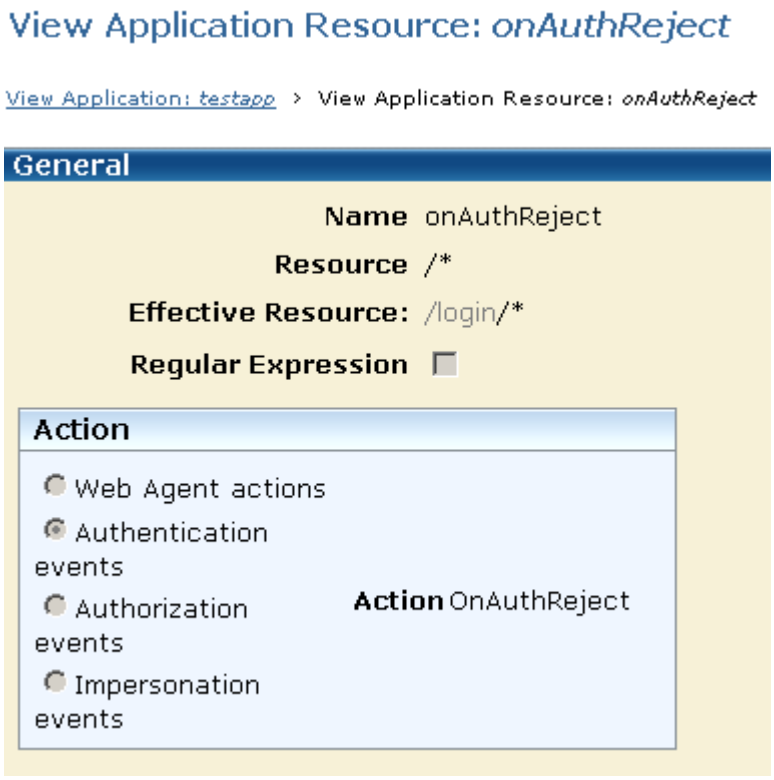
   /*

   Action

   Authentication events: OnAuthReject

4. Click OK

   The resource is created.



Designate a Resource for Mobile Authentication Requests

Configure a resource for successful mobile authentication requests.

Follow these steps:

1. On the Create Application page, click the Resources tab.
2. Click Create.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Specify the following resource properties:

   Name

   success

   Resource

   /*

   Action

   Web Agent Actions: Get

4. Click OK

   The Resource is created.

Create Roles That Identify the Users That Can Access the Mobile App

Specify roles that define the set of users who have access to the mobile app.

Follow these steps:

1. On the Create Application page, click the Roles tab.
2. Click Create Role.
3. Verify that the Create new object of type Role option is selected, and click OK.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Enter a name and optionally, a description for the role.
5. Specify whether the role applies to All Users or Selected Users in the configured user directories.

   Note: The Users Setup and Advanced sections do not apply when the All Users option is set and are no longer displayed.

6. Define the groups, organizations, and user attribute expressions that define the members of the role by making selections in the Users Setup group box.
7. Click OK.
8. Repeat steps 2 through 8 for each additional required role.

Configure Responses to Associate with Authentication Success and Failure Events

Configure responses to associate with resources to define how the policy responds when a resource is accessed or an event occurs. In this case, configure responses to redirect the mobile application to resources to inform it of the success or failure of an authentication attempt.

Configure the following responses:

- success (see page 33)
- fail (see page 34)

Configure a Response to Associate with Authentication Success Events

Configure a success response to associate with the onAuthAccept resource in the policy.

Follow these steps:

1. On the Create Application page, click the Response tab.
2. Click Create Response.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Enter "success" in the Name field.
4. Create a response attribute:
   a. Click Create Response Attribute.
   b. Select WebAgent-HTTP-OnAccept-Redirect attribute type.
   c. Select Static as the Attribute Kind.

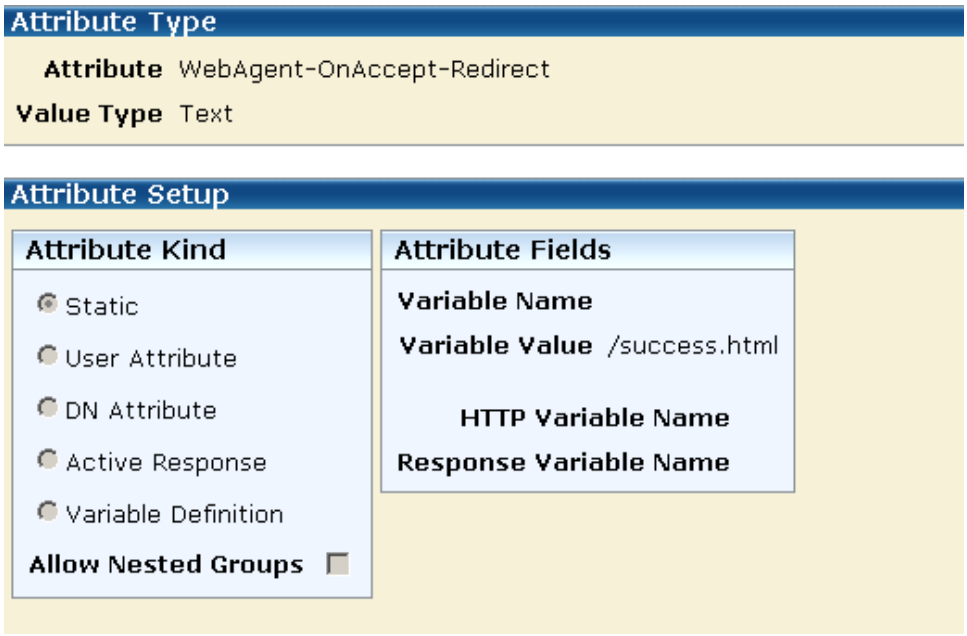
      The details in the Attribute Fields are updated to match the specified attribute kind.
   d. Enter "/success.html" in the Variable Value field.

e. Click OK.

The response attribute is added to the Attribute List.

5. Click OK.

The success response is created.



Configure a Response to Associate with Authentication Failure Events

Configure a failure response to associate with the onAuthAttempt and onAuthFailure resources in the policy.

Follow these steps:

1. On the Create Application page, click the Response tab.
2. Click Create Response.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Enter "failure" in the Name field.
4. Create a response attribute:

   a. Click Create Response Attribute.

   b. Select WebAgent-HTTP-OnReject-Redirect attribute type.

   c. Select Static as the Attribute Kind.

   The details in the Attribute Fields are updated to match the specified attribute kind.

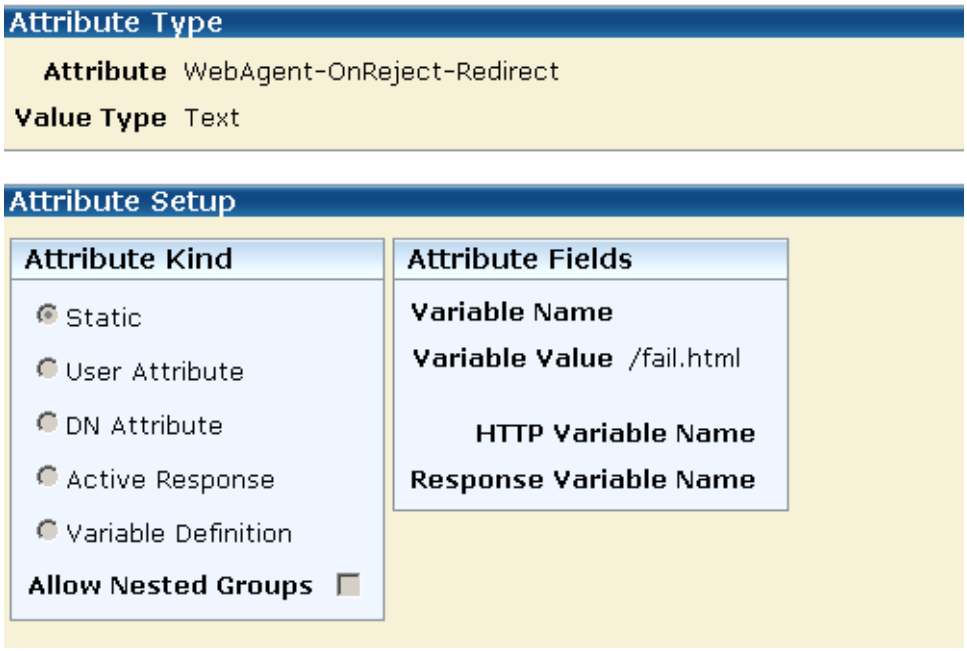
d. Enter "/fail.html" in the Variable Value field.

e. Click OK.

The response attribute is added to the Attribute List.

5. Click OK.

The failure response is created.



Configure the Policy for Mobile Authentication

Configure a policy that allows all defined user roles to access all configured resources and to associate specific resources with appropriate responses. Associate the **onAuthAccept** resource with the **success** response so that success.html is returned to the app when authentication is successful. Also, associate the **onAuthAttempt** and **onAuthReject** resources with the **fail** response so that fail.html is returned to the app when authentication is unsuccessful.

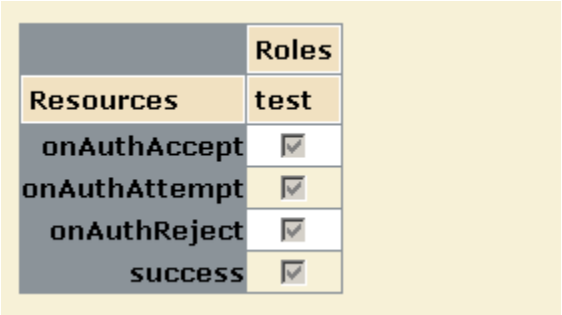
Follow these steps:

1. On the Create Application page, click the Policies tab.

   The Policies tab displays two tables, one that lists resources and roles, the other that lists resources and responses.

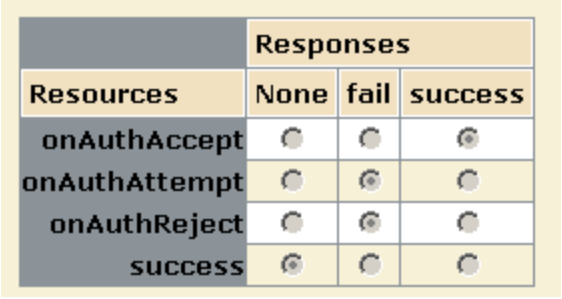
   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. In the resources table, associate all roles with all resources.



3. In the responses table, associate resources with responses as shown in the following table:

Resource	Response
onAuthAccept	success
onAuthAttempt	fail
onAuthReject	fail
success	None



4. Click Submit.

   A confirmation screen appears. The mobile authentication security policy is created.

This policy allows all users defined in defined roles to use the mobile application.

Copyright