Enhanced Authentication Use Cases

Everyday scenarios for implementing the SG 5.2 Guest Authentication, Permit Authentication Error, and User Management Features

Enhanced Authentication in SG 5.2

With the release of SG 5.2, enhanced authentication capabilities are now available. The enhanced authentication includes several features – Guest Authentication, Permit Authentication Error, and User Management – which provide administrators with the ability to more effectively manage and control users on the network. These features can be used independently, or together for more flexibility in customizing a solution that is best designed for your corporate policies.

Enhanced Authentication Use Cases

The following provides common use cases for the Enhanced Authentication features in SG 5.2. Each use case contains detailed instructions on how to configure the scenario using the VPM (Visual Policy Manager) as well as the resulting CPL (Content Policy Language). Each use case assumes that the ProxySG has already been configured as forward proxy (unless otherwise noted), that the administrator has a default policy of allow, and that no other policy exists. When policy does exist, administrators should understand how configuring these examples will interact with existing rules. Many of these use cases also require specific authentication modes to be implemented. If you are unfamiliar with authentication modes and how they work, please refer to the Configuration and Management Guide chapter "Controlling Access to the Internet and Intranet".

Table of Contents

USER MANAGEMENT USE CASES .............................................................................................................. 2
Inactivity Timeout [Forward Proxy] ................................................................................................................. 2
Inactivity Timeout [Reverse Proxy] .................................................................................................................. 3
Manual User Logout ........................................................................................................................................ 4
One Login per User [Log out old session] ....................................................................................................... 5
One Login per User [Deny new sessions] ......................................................................................................... 7
One Login per IP Address [Log out old sessions] ............................................................................................ 8
One Login per IP Address [Deny new sessions] .............................................................................................. 10
Limit Login Session Time [Force user to reauthenticate every N seconds/minutes/hours] ............................. 11

PERMIT AUTHENTICATION ERROR AND GUEST AUTHENTICATION USE CASES ............ 13
Windows Single Sign-On Failure – User not logged into domain ................................................................. 14
Communication Failure – Server is down/unreachable .................................................................................. 15
Automatic Guest Login After Multiple Authentication Failures .................................................................. 17
Guest Login Based on Source IP/Subnet ......................................................................................................... 17
Guest Login Based on URL [Transparent Authentication Deployments] ...................................................... 19
Guest Login Based on URL [Explicit Authentication Deployments] ............................................................ 20
Password Expired [Redirect user to password change page] ........................................................................ 21
Authorization Failure – Authorization data cannot be obtained .................................................................... 23
User Management Use Cases

Inactivity Timeout (Forward Proxy)

This use case provides automatic logging out of users after 5 minutes of inactivity.

1. Create an authentication realm that challenges for credentials (Local, RADIUS, LDAP, etc).
2. Configure the realm to have a five minute inactivity timeout.
   - Navigate to Configuration → Authentication → {Realm Type} → {Realm Type} General.
   - Set the inactivity timeout to 300 seconds.
   - Click "Apply".
3. Authenticate users
   - Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   - From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   - Name the Web Authentication Layer (optional) and click "OK".
   - In the action column right-click and select "Set".
   - Click "New" and select "Authenticate".
   - Name the action created (optional).
   - Choose the realm to authenticate users from the list provided.
   - Choose either “Form Cookie Redirect” or “Origin Cookie Redirect” mode depending on how you would like to challenge the user. A cookie-based surrogate is required for this use case to provide the ProxySG with a mechanism of tracking the login session.
   - Click "OK".
   - Click "OK".
   - Select “Install Policy” in the upper right-hand corner of the VPM.

Resulting CPL

```
<proxy>
authenticate(local_realm) authenticate.force(no) authenticate.mode(origin-cookie-redirect)
```

Caveats

- The inactivity logout currently depends upon the browser dropping its connections. This is done at different rates by different browsers: Firefox [5 minutes], IE [1 minute], Opera [2 minutes].
• If a site has AJAX features which are actively querying the backend, the connection may never be inactive and the user will not get logged out until 5 minutes after they close their browser.

• Setting a short inactivity timeout will log the user out quickly after they close their browser, but it will not be honored if the browser is left open.

• If a cookie-based authentication mode is not used, the user will still correctly be logged out of the ProxySG appliance, but if they do not close their browser they will not see a challenge. This is because the browser will automatically resubmit the user’s credentials. The user will therefore seem not to have been logged out.

**Inactivity Timeout (Reverse Proxy)**

This use case provides automatic logging out of users after 5 minutes of inactivity. This use case also works with SSL Reverse Proxy.

1. Create an authentication realm that challenges for credentials (Local, RADIUS, LDAP, etc).

2. Configure the realm to have a five minute inactivity timeout.
   - Navigate to Configuration → Authentication → {Realm Type} → {Realm Type} General.
   - Set the inactivity timeout to 300 seconds.
   - Click “Apply”.

3. Authenticate users.
   - Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   - From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   - Name the Web Authentication Layer (optional) and click “OK”.
   - In the action column right-click and select “Set”.
   - Click “New” and select “Authenticate”.
   - Name the action created (optional).
   - Choose the realm to authenticate users from the list provided.
   - Choose either “Form Cookie” or “Origin Cookie” mode depending on how you would like to challenge the user. A cookie-based surrogate is required for this use case to provide the ProxySG with a mechanism of tracking the login session.
   - Click “OK”.
   - Click “OK”.
   - Select “Install Policy” in the upper right-hand corner of the VPM.
Resulting CPL

\[ \text{authenticate(local\_realm) authenticate.force(no) authenticate.mode(origin-cookie)} \]

Caveats

- The inactivity logout currently depends upon the browser dropping its connections. This is done at different rates by different browsers: Firefox [5 minutes], IE [1 minute], Opera [2 minutes].
- If a site has AJAX features which are actively querying the backend, the connection may never be inactive and the user will not get logged out until 5 minutes after they close their browser.
- Setting a short inactivity timeout will log the user out quickly after they close their browser, but it will not be honored if the browser is left open.

Manual User Logout

This use case provides a manual logout option for users by providing them with a URL that when accessed will automatically log them out of the ProxySG appliance. The logout page is commonly bookmarked by users or placed on an Intranet banner.

1. Create an authentication realm that challenges for credentials (Local, RADIUS, LDAP, etc).
2. Authenticate users.
   - Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   - From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   - Name the Web Authentication Layer (optional) and click “OK”.
   - In the action column right-click and select “Set”.
   - Click “New” and select “Authenticate”.
   - Name the action created (optional).
   - Choose the realm to authenticate users from the list provided.
   - Choose either “Form Cookie Redirect” or “Origin Cookie Redirect” mode depending on how you would like to challenge the user. A cookie-based surrogate is required for this use case to provide the ProxySG with a mechanism of tracking the login session.
   - Click “OK”.
   - Click “OK”. 
1. Create a logout Web page on an internal server. The logout page should not contain any embedded objects as they will force the user to be logged back in (if an embedded object is required see note below). The page should also not be cacheable by the browser or proxy. In IIS this can be set by selecting the file’s properties and setting HTTP Headers → Enable Content Expiration → Expire Immediately.

Note: If an embedded object is required (such as a company logo) the logout URL should be configured as the embedded object URL instead of the logout webpage URL. The embedded object should also be unique [not the exact URL used on other webpages] else it will log the user out when accessing other pages with the same object.

2. Create policy to log out the user when they access the logout Web page.

   - Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   - From the pull-down menu at the top select Policy → Add Web Access Layer.
   - Name the Web Access Layer (optional) and click “OK”.
   - In the destination column right-click and select “Set”.
   - Click “New” and select “Request URL”.
   - Enter the URL of the log out webpage and click “Add” and then “Close”.
   - Click “OK”.
   - In the action column right-click and select “Set”.
   - Select “Logout User” and click “OK”.
   - Select “Install Policy” in the upper right-hand corner of the VPM.

Resulting CPL

```cpl
<proxy>
authenticate(local_realm) authenticate.force(no)
authenticate.mode(origin-cookie-redirect)
</proxy>
<proxy>
url.domain="http://intranet.company.com/user_logout.html" user.login.log_out(=yes)
</proxy>
```

Caveats

- Works with SSL traffic only if the ProxySG appliance is intercepting SSL traffic.
- If a cookie-based authentication mode is not used, the user will still correctly be logged out of the ProxySG appliance, but if they do not close their browser they will not see a challenge. This is because the browser will automatically resubmit the user’s credentials. The user will therefore seem not to have been logged out.
One Login per User (Log out old session)

This use case limits the number of active logins per user to only one. A common scenario that would use this configuration would be to limit a particular user from logging on to multiple workstations. Successfully logging into one workstation when an active login for that user already exists on another workstation results in the older login on the previous workstation being logged out.

1. Create an authentication realm that challenges for credentials (Local, RADIUS, LDAP, etc).

2. Authenticate users.
   - Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   - From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   - Name the Web Authentication Layer (optional) and click "OK".
   - In the action column right-click and select “Set”.
   - Click “New” and select “Authenticate”.
   - Name the action created (optional).
   - Choose the realm to authenticate users from the list provided.
   - Choose either “Form Cookie Redirect” or “Origin Cookie Redirect” mode depending on how you would like to challenge the user. A cookie-based surrogate is required for this use case to provide the ProxySG with a mechanism of tracking the login session.
   - Click “OK”.
   - Click “OK”.

3. Create policy to logout any existing login sessions associated with the user when a new login is successful.
   - From the pull-down menu at the top select Policy → Add Web Access Layer.
   - Name the Web Access Layer (optional) and click "OK".
   - In the source column right-click and select “Set”.
   - Click “New” and select “User Login Count”.
   - Name the source object created (optional) and enter a user login count of 2 (this will match 2 or more logins).
   - Click “OK”.

Technical Brief: Enhanced Authentication Use Cases
• Click “OK”.
• In the action column right-click and select “Set”.
• Select “Logout User’s Other Sessions” and click “OK”.
• Select “Install Policy” in the upper right-hand corner of the VPM.

Resulting CPL

```xml
<proxy>
authenticate(local.realm) authenticate.force(no)
authenticate.mode(origin-cookie-redirect)
<proxy>
user.login.count=(2) user.login.log_out_other(yes)
```

Caveats

• Works with SSL traffic only if the ProxySG appliance is intercepting SSL traffic.

• If a cookie-based authentication mode is not used, the user will still correctly be logged out of the ProxySG appliance, but if they do not close their browser they will not see a challenge. This is because the browser will automatically resubmit the user’s credentials. The user will therefore seem not to have been logged out.

One Login per User (Deny new sessions)

This use case limits the number of active logins per user to only one. A common scenario that would use this configuration would be to limit a particular user from logging on to multiple workstations. A successful login on a workstation when an active login already exists on another workstation results in the user being denied and immediately logged out until logging out of the workstation with the active login.

1. Create ANY authentication realm.
2. Authenticate users.
   • Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   • From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   • Name the Web Authentication Layer (optional) and click “OK”.
   • In the action column right-click and select “Set”.
   • Click “New” and select “Authenticate”.
   • Name the action created (optional).
• Choose the realm to authenticate users from the list provided.
• Click "OK".
• Click "OK".

3. Create policy to deny and logout any new login sessions associated with the user when other active login sessions exist.
• From the pull-down menu at the top select Policy → Add Web Access Layer.
• Name the Web Access Layer (optional) and click "OK".
• In the source column right-click and select "Set".
• Click "New" and select "User Login Count".
• Name the source object created (optional) and enter a user login count of 2 (this will match 2 or more logins).
• Click "OK".
• Click "OK".
• In the action column right-click and select "Set".
• Click "New" and select "Combined Action Object".
• Name the Combined Action Object (optional).
• Select both "Deny" and "Logout User" using the "Add" button and click "OK".
• Click "OK".
• Select "Install Policy" in the upper right-hand corner of the VPM.

Resulting CPL

<proxy>
authenticate(local_realm) authenticate.force(no) authenticate.mode(auto)

<proxy>
user.login.count=[2] Deny user.login.log_out(yes)

One Login per IP Address (Log out old sessions)

This use case limits the number of active logins per IP address to only one. A common scenario that would use this configuration would be to prevent multiple users from simultaneously logging in to the same workstation. A successful login on a workstation when an active login already exists on that same workstation results in the previous user being automatically logged out.
1. Create an authentication realm that challenges for credentials (Local, RADIUS, LDAP, etc).

2. Authenticate users.
   - Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   - From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   - Name the Web Authentication Layer (optional) and click "OK".
   - In the action column right-click and select "Set".
   - Click "New" and select "Authenticate".
   - Name the action created (optional).
   - Choose the realm to authenticate users from the list provided.
   - Choose either “Form Cookie Redirect” or “Origin Cookie Redirect” mode depending on how you would like to challenge the user. A cookie-based surrogate is required for this use case to provide the ProxySG with a mechanism of tracking the login session.
   - Click "OK".
   - Click "OK".

3. Create policy to log out any existing login sessions associated with the IP address when a new login is successful.
   - From the pull-down menu at the top select Policy → Add Web Access Layer.
   - Name the Web Access Layer (optional) and click "OK".
   - In the source column right-click and select "Set".
   - Click "New" and select "Client Address Login Count".
   - Name the source object created (optional) and enter a client address count of 2 [this will match 2 or more logins].
   - Click "OK".
   - Click "OK".
• In the action column right-click and select “Set”.
• Select “Logout Other Users With Same IP” and click “OK”.
• Select “Install Policy” in the upper right-hand corner of the VPM.

Resulting CPL

```xml
<proxy>
authenticate(local_realm) authenticate.force(no) authenticate.mode(origin-cookie-redirect)
<proxy>
client.address.login.count=(2) client.address.login.log_out_other(yes)
```

Caveats

• Works with SSL traffic only if the ProxySG appliance is intercepting SSL traffic.
• If a cookie-based authentication mode is not used, the user will still correctly be logged out of the ProxySG appliance, but if they do not close their browser they will not see a challenge. This is because the browser will automatically resubmit the user’s credentials. The user will therefore seem not to have been logged out.

One Login per IP Address (Deny new sessions)

This use case limits the number of active logins per IP address to one. A common scenario that would use this configuration would be to prevent multiple users from simultaneously logging in to the same workstation. An attempt to log in to a workstation in which an active login already exists results in the user being denied until the user with the active session on that workstation logs out manually or is automatically logged out after the inactivity timeout passes. This example assumes that the inactivity timeout will initiate the logout.

1. Create any authentication realm.
2. Authenticate users.
   • Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   • From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   • Name the Web Authentication Layer [optional] and click “OK”.
   • In the action column right-click and select “Set”.
   • Click “New” and select “Authenticate”.
   • Name the action created [optional].
   • Choose the realm to authenticate users from the list provided.
   • Click “OK”.
   • Click “OK”.

1
2
3
Create policy to deny and log out any new login sessions associated with the IP address when other active login sessions exist.

- From the pull-down menu at the top select Policy → Add Web Access Layer.
- Name the Web Access Layer (optional) and click "OK".
- In the source column right-click and select "Set".
- Click "New" and select "Client Address Login Count".
- Name the source object created (optional) and enter a client address count of 2 (this will match 2 or more logins).
- Click "OK".
- In the action column right-click and select "Set".
- Click "New" and select "Combined Action Object".
- Name the Combined Action Object (optional).
- Select both "Deny" and "Logout User" using the "Add" button and click "OK".
- Click "OK".
- Select "Install Policy" in the upper right-hand corner of the VPM.

Resulting CPL

```xml
<proxy>
authenticate(local_realm) authenticate.force(no) authenticate.mode(auto)

<proxy>
client.address.login.count=(2) Deny user.login.log_out(yes)
```

Limit Login Session Time

(Force user to reauthenticate every N seconds/minutes/hours)

This use case limits the amount of time that a user can be logged in without being explicitly challenged for credentials to 2 hours. A common scenario that would implement this would be an administrator that wants to have a user explicitly challenged at a regular interval instead of the browser automatically providing credentials when the surrogate credentials expire.
1. Create an authentication realm that challenges for credentials (Local, RADIUS, LDAP, etc).

2. Authenticate users.
   - Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   - From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   - Name the Web Authentication Layer (optional) and click "OK".
   - In the action column right-click and select "Set".
   - Click "New" and select "Authenticate".
   - Name the action created (optional).
   - Choose the realm to authenticate users from the list provided.
   - Choose either "Form Cookie Redirect" or "Origin Cookie Redirect" mode depending on how you would like to challenge the user. A cookie-based surrogate is required for this use case to provide the ProxySG with a mechanism of tracking the login session.
   - Click "OK".
   - Click "OK".

3. Create policy to log out sessions when they reach 2 hours.
   - From the pull-down menu at the top select Policy → Add Web Access Layer.
   - Name the Web Access Layer (optional) and click "OK".
   - In the source column right-click and select "Set".
   - Click "New" and select "User Login Time".
   - Name the source object created (optional) and enter a user login time of 7200. This will match 2 or more hours.
   - Click "OK".
   - Click "OK".
   - In the action column right-click and select "Set".
   - Click "New" and select "Combined Action Object".
   - Name the Combined Action Object (optional)
   - Select "Logout User" and click "Add"
• Click "New" and select "Return Exception"
• Name the Return Exception Object (optional)
• From the “built-in exception” list select "refresh" and click "OK"
• Select the newly created “Return Exception” object and click “Add”
• Click "OK”.
• Select “Install Policy” in the upper right-hand corner of the VPM.

Resulting CPL

```xml
<proxy>
authenticate(local_realm) authenticate.force(no) authenticate.mode(origin-cookie-redirect)
</proxy>

<proxy>
user.login.time=(7200..) user.login.log_out(yes) exception[refresh]
</proxy>
```

Caveats

• Works with SSL traffic only if the ProxySG appliance is intercepting SSL traffic.
• If a cookie-based authentication mode is not used, the user will still correctly be logged out of the ProxySG appliance, but if they do not close their browser they will not see a challenge. This is because the browser will automatically resubmit the user’s credentials. The user will therefore seem not to have been logged out.

Permit Authentication Error and Guest Authentication Use Cases

General Permit Authentication Error Caveats and Reminders

• Authentication errors cannot be permitted during admin authentication.
• Realms that do not support split authentication and authorization [e.g. IWA, SiteMinder, COREid] will still fail authentication if authorization fails for any reason.
• Authorization cannot succeed if authentication has failed. In those cases, authorization is not attempted.
• If no authorization errors are explicitly permitted or not permitted in policy then authorization will default to permitting the same set of errors [plus not_attempted] that have been specified for authentication.
• Important: Tolerating the error group “all” will result in all authentication errors being tolerated including ‘need_credentials’. If specified, users will never be challenged which is often not the desired behaviour. Use the error group “all” very carefully.
General Guest Authentication Error Caveats and Reminders

- A realm is specified to associate with the guest but it does not actually authenticate the user against the backend authentication server. Instead it determines the guest username using policy substitution evaluation as needed and marks the user as authenticated and authorized in the specified realm.

- Regular (non-guest) authentication has priority over guest authentication. If a transaction encounters an `authenticate` and `authenticate.guest` policy condition it will attempt the `authenticate` first. If the regular authentication fails with a tolerated error then the transaction will authenticate as the guest user.

- To avoid having to re-authenticate the user as guest on every connection (which may first require a challenge and authentication failure or a guest login URL request) use an authentication mode that uses IP or cookie-based surrogates. If both an authenticate and authenticate.guest action will be matched by the transaction then the authentication mode specified for the regular authenticate action will also be used for the guest authentication action. If a particular transaction will not match a regular authenticate action but it is desired to assign a surrogate to the guest user (such as when providing a guest login URL) then a separate authentication mode must be specified for the authenticate guest action. Specifying an authentication mode for an authenticate guest action is not possible in the VPM but can be specified using CPL. If specifying an authentication mode for both regular and guest authentication, ensure that they match so that the surrogates can be used when the transaction hits either authenticate action.

- Permitted authentication and authorization errors do not apply to guest authentication. Guest authentication will very rarely fail (typically only due to internal errors such as memory allocation failures).

- Guest authentication cannot be used for admin authentication.

- Guest authentication after an authentication challenge causes the client software (browser) to believe that the credentials offered were valid. The browser will therefore re-offer the cached credentials on a subsequent challenge even though they may not be valid.

Windows Single Sign-On Failure – User not logged into domain

This use case allows users who are not logged into the single sign-on domain to be granted access to network resources. This example also identifies such users as guest users.

1. Configure a Windows SSO authentication realm.
2. Configure policy to authenticate users but allow single sign-on authentication failures.

   - Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   - From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   - Name the Web Authentication Layer (optional) and click “OK”.
   - In the action column right-click and select “Set”.
   - Click “New” and select “Authenticate”.
• Name the action created (optional).
• Choose the single sign-on realm to authenticate users from the list provided.
• Choose an authentication mode that uses either an IP or cookie-based surrogate so that the surrogate can be used for both guest and regular authentication.
• Click “OK”. 
• Click “New” and select “Permit Authentication Error”.
• Click the “Selected Errors” radio button and select “Single Sign-On Failure”.
• Click “OK”. 
• Click “New” and select “Combined Action Object”.
• Name the Combined Action Object created (optional).
• From the Combined Action Object window select the two objects just created using the “Add” button.
• Click “OK”.
• Click “OK”. 
 Configure policy to identify unauthenticated users as guests by assigning them a username of “unauthenticated_sso_user”.
• From the pull-down menu at the top add a second Web Authentication Layer by selecting Policy \rightarrow Add Web Authentication Layer.
• Name the Web Authentication Layer (optional) and click “OK”.
• In the action column right-click and select “Set”.
• Click “New” and select “Authenticate Guest”.
• Name the action created (optional).
• Enter the guest username to assign to guest users – this example uses “unauthenticated_sso_user”.
• Choose the single sign-on realm to associate with guest users.
Configure a surrogate refresh time [optional].

Click "OK".

Click "OK".

Select "Install Policy" in the upper right-hand corner of the VPM.

**Resulting CPL**

```cpl
<proxy>
authenticate(wsso) authenticate.force(no) authenticate.mode(proxy-ip) authenticate.tolerate_error(sso_failure)[yes]
</proxy>

<proxy>
authenticate.guest("unauthenticated_sso_user", 0, wsso)
</proxy>
```

**Communication Failure – Server is down/unreachable**

This use case allows users to be granted access to network resources in the event of a communication failure between the ProxySG and the authentication server. Users allowed access during the communication failure are identified as guest users with a short surrogate refresh time configured so that reauthentication with the server is attempted frequently. Once the authentication server is available and the guest surrogate has expired, users will successfully authenticate with their regular username and will no longer be identified as guest users.

*Note:* Server failures are part of the “Communication Error” group in the VPM as well as part of a larger error group “General Offbox Error”. The “Communication Error” group contains BCAAA, LDAP, RADIUS, and NetBIOS communication errors. If an administrator would like to create policy that tolerates only a very specific authentication error such ldap_busy policy must be written using CPL. Such granular policy is not currently available via the VPM. A list of all of the VPM error groups and their members is available by issuing the “show security authentication-errors” CLI command.

1. Configure an authentication realm that communicates with an external server (LDAP is used in this example).

2. Configure policy to authenticate users but allow communication failures.
   - Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   - From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   - Name the Web Authentication Layer [optional] and click "OK".
   - In the action column right-click and select "Set".
   - Click "New" and select "Authenticate".
   - Name the action created [optional].
   - Choose the realm to authenticate users from the list provided.
- Choose an authentication mode that uses either an IP or cookie-based surrogate so that the surrogate can be used for both guest and regular authentication.
- Click “OK”.
- Click “New” and select “Permit Authentication Error”.
- Click the “Selected Errors” radio button and select “Communication Error”.
- Click “OK”.
- Click “New” and select “Combined Action Object”.
- Name the Combined Action Object created (optional).
- From the Combined Action Object window select the two objects just created using the “Add” button.
- Click “OK”.
- Click “OK”.

Configure policy to identify unauthenticated users as guests by assigning them a username of “unauthenticated_ldap_user”.
- From the pull-down menu at the top add a second Web Authentication Layer by selecting Policy → Add Web Authentication Layer.
- Name the Web Authentication Layer (optional) and click “OK”.
- In the action column right-click and select “Set”.
- Click “New” and select “Authenticate Guest”.
- Name the action created (optional).
- Enter the guest username to assign to guest users – this example uses “unauthenticated_ldap_user”.
- Choose the realm to associate with guest users.
- Configure a short surrogate refresh time (300 seconds) so that authentication against the LDAP server is re-attempted sooner (optional).
• Click “OK”.
• Click “OK”.
• Select “Install Policy” in the upper right-hand corner of the VPM.

Resulting CPL

```<proxy>
authenticate(ldap) authenticate.force(no) authenticate.mode(origin-cookie-redirect) authenticate.tolerate_error[communication_error](yes)
</proxy>
<proxy>
authenticate.guest("unauthenticated_ldap_user", 300, ldap)
</proxy>```

Automatic Guest Login After Multiple Authentication Failures

The previous two examples illustrate automatically identifying a guest user after a single authentication failure. It is not possible to automatically identify a guest user after multiple authentication attempts. It is possible, however, to allow users to explicitly login as a guest user after multiple authentication attempts. This is accomplished by providing a guest user login URL in the exception page returned by the ProxySG after an authentication failure. See the Guest Login Based on URL use case for additional information.

Guest Login Based on Source IP/Subnet

This use case allows users to be identified as guest users and granted access to network resources based on source IP or subnet. Authentication is not attempted for these users. This scenario is popular with public/guest workstations.

1. Configure any authentication realm.
2. Configure policy to identify guest users based on IP or subnet.
   • Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   • From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   • Name the Web Authentication Layer (optional) and click “OK”.
   • In the source column right-click and select “Set”.
   • Click “New” and select “Client IP Address/Subnet”.
   • Enter the desired IP address and subnet for guest authentication.
   • Click “Add” and then “Close”.
   • Click “OK”.

- In the action column right-click and select “Set”.
- Click “New” and select “Authenticate Guest”.
- Name the action created (optional).
- Enter the guest username to assign to guest users – this example uses “guest_” prepended to the client’s IP address.
- Choose the realm to associate with guest users.
- Configure a surrogate refresh time (optional).
- Click “OK”.
- Click “OK”.

3. Configure policy to authenticate regular network (non-guest) users.
- In the existing Web Authentication Layer click “Add Rule”.
- In the action column right-click and select “Set”.
- Click “New” and select “Authenticate”.
- Name the action created (optional).
- Choose the realm to authenticate users from the list provided.
- Click “OK”.
- Click “OK”.
- Select “Install Policy” in the upper right-hand corner of the VPM.

**Resulting CPL**

```xml
<proxy>
  client.address=192.168.52.0/24 authenticate.guest("guest_${client.address}", 0, ldap)
  authenticate(dap) authenticate.force(no) authenticate.mode(auto)
</proxy>
```

**Guest Login Based on URL (Transparent Authentication Deployments)**

This use case allows users to be identified as guest users and granted access to network resources based on clicking on a “Guest User Login URL”. There are two common variations of this use case. A link can be made available on a company website providing guest user access with no authentication attempt required (shown in this example). Another variation is to provide a guest login link on the authentication failure page which allows users to log in as guests only after failing authentication. The end-user experience associated with the second scenario will depend on the client’s user agent. Some browsers attempt authentication a set number of times (usually 3) and then present the exception page. Other browsers continuously rechallenge the user until the
user selects Cancel at which point the exception page is displayed. Once the user has clicked on the guest user login URL they can be presented with a page indicating that they have logged in as a guest user or, alternately, they can be automatically redirected to the original page they had requested.

1. Configure any authentication realm.
2. Configure policy to identify guest users based on a guest URL.
   - Navigate to Configuration → Policy → Visual Policy Manager → Launch.
   - From the pull-down menu at the top select Policy → Add Web Authentication Layer.
   - Name the Web Authentication Layer (optional) and click “OK”.
   - In the destination column right-click and select “Set”.
   - Click “New” and select “Request URL”.
   - Enter the URL that will be used for guest authentication (e.g. www.company.com/guest_login.html).
   - Click “Add” and then “Close”.
   - Click “OK”.
   - In the action column right-click and select “Set”.
   - Click “New” and select “Authenticate Guest”
   - Name the action created (optional).
   - Enter the guest username to assign to guest users – this example uses “guest_” prepended to the client’s IP address.
   - Choose the same realm that will be used to authenticate users with credentials (using a different realm requires a different policy solution).
   - Configure a surrogate refresh time (optional).
   - Click “OK”.
   - Click “OK”.
3. Configure policy to authenticate regular network (non-guest) users.
   - In the existing Web Authentication Layer click “Add Rule”.
   - In the action column right-click and select “Set”.

Technical Brief: Enhanced Authentication Use Cases
Click "New" and select "Authenticate".

Name the action created (optional).

Choose the realm to authenticate users from the list provided.

Choose an authentication mode that uses either an IP or cookie-based surrogate so that the surrogate can be used for both guest and regular authentication (this includes auto for transparent authentication deployments).

Click "OK".

Click "OK".

Select "Install Policy" in the upper right-hand corner of the VPM.

Resulting CPL

```xml
<proxy>
  url.domain=//www.company.com/guest_login.html authenticate.guest("guest_${client.address}", 0, ldap)
  authenticate(ldap) authenticate.force(no) authenticate.mode(origin-cookie-redirect)
</proxy>
```

**Guest Login Based on URL (Explicit Authentication Deployments)**

This use case is identical to the Guest Login Based on URL for Transparent Authentication Deployments with the exception that it requires that the administrator force an IP or cookie-based surrogate for the guest authenticate action. In most explicit deployments the only surrogate used is the connection. As a result, users must re-authenticate on each connection which is typically not desired. To prevent this behavior in explicit deployments, an authentication mode that uses IP or cookie-based surrogates must be used for all authentication actions. While this can be easily configured for the authenticate action in the VPM (as in the Transparent Authentication example), there is no mechanism available in the VPM to configure the surrogate used for the guest authenticate action. This use case therefore requires a CPL-based solution. The CPL to accomplish a Guest Login Based on URL in an explicit authentication deployment is as follows:

```xml
<proxy>
  url.domain=//www.company.com/guest_login.html authenticate.guest("guest_${client.address}", 0, ldap)
  authenticate.mode(origin-cookie-redirect)
  authenticate(ldap) authenticate.force(no) authenticate.mode(origin-cookie-redirect)
</proxy>
```

**Password Expired (Redirect user to password change page)**

This use case allows the password expired failure and redirects the user to a “password change” page that the administrator has created.

Note: Password expired failures are part of the “User Account Error” group in the VPM. If an administrator would like to create policy that redirects to the password change page only in the event of ‘expired_credentials’ or ‘account_must_change_password’ errors policy must be written using CPL. Such granular policy is not currently available via the VPM. A list of all of the VPM error groups and their members is available by issuing the “show security authentication-errors” CLI command.
Configure an authentication realm that supports password expiration.

Configure policy to authenticate users but allow password expiration failures.

- Navigate to Configuration → Policy → Visual Policy Manager → Launch.
- From the pull-down menu at the top select Policy → Add Web Authentication Layer.
- Name the Web Authentication Layer (optional) and click “OK”.
- In the action column right-click and select “Set”.
- Click “New” and select “Authenticate”.
- Name the action created (optional).
- Choose the realm to authenticate users from the list provided.
- Click “OK”.
- Click “New” and select “Permit Authentication Error”.
- Click the “Selected Errors” radio button and select “User Account Error” using the “Add” button.
- Click “OK”.
- Click “New” and select “Combined Action Object”.
- Name the Combined Action Object created (optional).
- From the Combined Action Object window select the two objects just created using the “Add” button.
- Click “OK”.
- Click “OK”.

Configure policy to redirect users to the password change page upon receiving the password expiration failure.

- From the pull-down menu at the top select Policy → Add Web Access Layer.
• Name the Web Authentication Layer [optional] and click “OK”.
• In the source column right-click and select “Set”.
• Click “New” and select “User Authentication Error”.
• Click the “Selected Errors” radio button and select “User Account Error”.
• Click “OK”.
• Click “OK”.
• In the action column right-click and select “Set”.
• Click “New” and select “Return Redirect”.
• Name the action created [optional].
• Enter in the URL of the password change page and click “OK”.
• Click “OK”.
• Select “Install Policy” in the upper right-hand corner of the VPM.

**Resulting CPL**

```
<proxy>
authenticate[ldap] authenticate.force(no) authenticate.
mode(auto) authenticate.tolerate_error[user_account_error][yes]

<proxy>
user.authentication_error={user_account_error} action. ReturnRedirect1[yes]

define action ReturnRedirect1
redirect(302, ".*", "http://www.company.com/password_update.asp");
end
```

**Authorization Failure – Authorization data cannot be obtained**

This use case assigns authenticated users to a default group in the event that their authorization data cannot be obtained. This example would be used in the scenario where an administrator has configured different realms for authentication and authorization and authorization could not be obtained due to a communication failure with the authorization server. This example uses a Policy Substitution realm to authenticate users and an LDAP realm to obtain authorization data for those users. Note that the purpose of assigning unauthorized users to a default group is to implement group-based policy (not shown in this example).
Configure authentication and authorization realms.

Configure policy to authenticate users but allow authorization failures.

2. From the pull-down menu at the top select Policy → Add Web Authentication Layer.
3. Name the Web Authentication Layer (optional) and click “OK”.
4. In the action column right-click and select “Set”.
5. Click “New” and select “Authenticate”.
6. Name the action created (optional).
7. Choose the realm to authenticate users from the list provided.
8. Click “OK”.
9. Click “New” and select “Permit Authorization Error”.
10. Click the “Selected Errors” radio button and select “Communication Error” using the “Add” button.
11. Click “OK”.
12. Click “New” and select “Combined Action Object”.
13. Name the Combined Action Object created (optional).
14. From the Combined Action Object window select the two objects just created using the “Add” button.
15. Click “OK”.
16. Click “OK”.

Technical Brief: Enhanced Authentication Use Cases
Configure policy to assign unauthorized users to a default group.

- From the pull-down menu at the top add a second Web Authentication Layer by selecting Policy → Add Web Authentication Layer.
- Name the Web Authentication Layer [optional] and click "OK".
- In the source column right-click and select "Set".
- Click "New" and select "User Authorization Error".
- Click the "Selected Errors" radio button and select "Communication Error" using the "Add" button.
- Click "OK".
- Click "OK".
- In the action column right-click and select "Set".
- Click "New" and select "Add Default Group".
- Enter a name for the default group and select the realm to associate with the default group.
- Enter a group base DN [required or optional depending on realm selection].
- Click "OK".
- Click "OK".
- Select "Install Policy" in the upper right-hand corner of the VPM.

Resulting CPL

```
<proxy>
authenticate(policy_substitution_realm) authenticate.force(no) authenticate.mode(auto) authorize.tolerate_error[communication_error](yes)
<proxy>
user.authorization_error=[communication_error] authorize.add_group("cn=default_group, cn=Users, dc=company, dc=com")
```