PISTOLSTAR®

PortalGuard
Application Programming Interface

API

PistolStar, Inc.
PO Box 1226
Amherst, NH 03031 USA

Phone: 603.547.1200
Fax: 603.546.2309
E-mail: salesteam@pistolstar.com
Website: www.pistolstar.com

Copyright and Disclaimer

Copyright 1999 - 2009 PistolStar, Inc. All rights reserved.

Trademarks

Contents

1 Introduction

About PistolStar

PistolStar, Inc. specializes in tailored authentication, providing software products and services that fit with the customer's environment, as well as optimize authentication processes and address requirements for enhanced usability, security, auditing and compliance. With its comprehensive solution set, PistolStar responds to an organization's need to secure access to information and ensure regulatory compliance, while simplifying the login process and reducing the IT staff's burden of managing passwords and tracking login threats. Launched in 1999, PistolStar is a pioneer in enabling authentication via Microsoft Active Directory and is an authority on authentication using Active Directory and Kerberos.

Overview of PortalGuard

This guide describes how to install and configure PortalGuard. With PortalGuard, administrators can lock down access to their corporate intranet and extranet portals while also providing end users with a more stream-lined, user-friendly authentication experience and giving them the ability to automatically recover their password directly from the browser without having to call the help desk.

PortalGuard will increase the security of your passwords by adding features such as password quality, history, expiration and lockout due to incorrect logins. By also allowing users to securely reset their *forgotten* passwords via the web, PortalGuard also reduces Help Desk calls and related user downtime.

Administrators can easily set password policies, such as having the user reset the password immediately following initial logon, or specifying a set number of days that the user will have until actually being forced to reset the password.

In addition to the many preferences that an Administrator can set, PortalGuard offers a user-friendly interface for users to logon to portal servers and change their passwords.

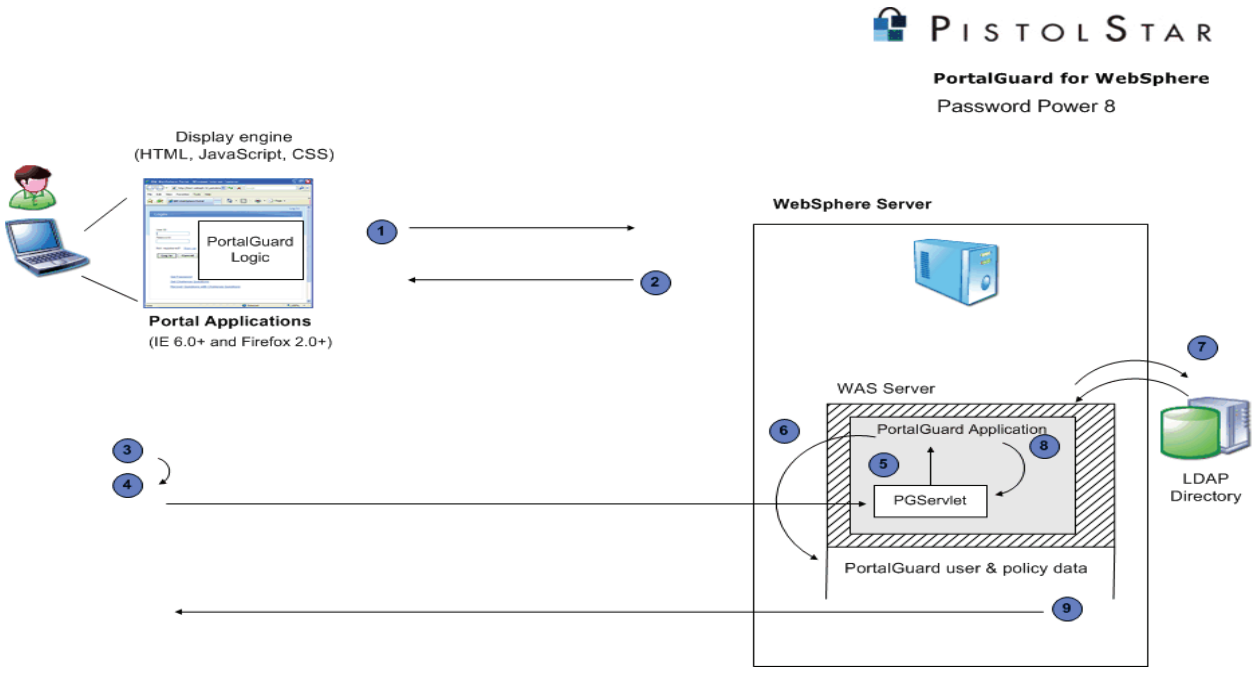
PortalGuard does not replace the user repository you currently have in place, nor does it store a copy of users' current passwords. It simply stores its own data (e.g. expiration date, challenge answer hashes, strike data) in a look-aside data store and contacts your current repository only to authenticate the credentials input by the user or change their password there.

How It Works

PortalGuard works by intercepting authentication requests from your portal and redirecting them to PortalGuard's login page. From this interface, the user will still be able to login using the same username and password as normal, but they will also be notified of things like password expiration, account lockout, insufficiently complex passwords and be able to reset their forgotten passwords using highly configurable challenge question and answer functionality. The image and workflow below describes how PortalGuard communicates with IBM's WebSphere/WebSphere Portal server in more detail.

1. Unauthenticated user opens web browser and requests a resource from the WebSphere Portal server.

2. The Portal's login page is returned as the response from the server. Alternatively, PortalGuard's logic can also be added to the source HTML of the standard login page.



3. The user enters their username & password in the normal fields and clicks the Login button.

4. The PortalGuard logic in the login page uses AJAX to send the login information in the background to a Java servlet called "PGServlet". This servlet resides on the Portal server and is created as part of the PortalGuard installation. The user's browser does not navigate away from the login page during this time.

5. The PGServlet dynamically loads and calls into the PortalGuard application on the Portal server. PortalGuard is implemented as stand-alone DLLs that are installed on the Portal server. These DLLs are loaded by the PGServlet using Java Native Interface (JNI).

6. PortalGuard reads in its policy and user profile data to determine how to authenticate the user. For example, if the user account is already locked, a response structure is populated with this information and the process skips to step #8.

7. The user is looked up in the designated LDAP server and if the user exists, the validity of the provided password is checked by performing a LDAP bind operation against the server with the submitted credentials.

8. PortalGuard checks to see if there are any constraints which should prevent this user from logging in (expired password, already a current login session, password is insufficiently complex, etc). A response structure is populated with the result of these checks and passed back to the PGServlet.

9. The PGServlet generates the response from the structure and sends the information back to the PortalGuard logic on the login page. The response is interpreted by JavaScript within the PortalGuard logic and any error conditions or messages are intuitively displayed to the user directly on the login page clearly prompting them with any corrective actions they may need to take.

If there are no conditions that should prevent the login, the PortalGuard logic allows the original login attempt from item #3 to continue which passes the authentication request to the Portal server's default authentication mechanism. This will result in the user receiving a LTPA token from WebSphere that will be used for the remainder of their browser session or until they manually log out.

2 PortalGuard Concepts

Policies

PortalGuard uses the concept of policy-based settings to enforce rules for users. You can have multiple sets of rules defined within PortalGuard. Each set of rules is referred to as a *policy*. You can then assign users to a policy based on group, domain hierarchy or individually. If a policy is not applied to anyone, then its rules will never be enforced.

Policies can also be enabled or disabled. Only policies which are both enabled and have users assigned to them will be enforced (NOTE: the "default" policy does not need to have users assigned). A disabled policy will not be enforced regardless of the number of people to which it is applied. Disabling policies is simple way to eliminate complexity when debugging a problem.

If a user is not assigned to at least one policy, then the rules of the *default policy* will automatically apply to them. The default policy has the lowest priority and will never be applied to a user if some other policy is applied to them.

Policy Search Order and Precedence

With policies capable of being applied to groups and domain hierarchies, it is a common occurrence for a user to have multiple policies applied to them. At run-time however, only a **single** policy will be enforced for the user. This disparity is resolved by searching for applicable policies in the following manner:

- Policies applied directly to a user
- Policies applied to a group
- Policies applied to a domain or OU
- The default policy

The policy search always occurs in this order. If one or more policies are found in step 1, then searching stops. Otherwise, the search continues to step 2. If one or more policies are found in step 2, then search stops, and so on.

If any one step produces multiple policies that match, then *policy precedence* is used to determine the applicable policy. Precedence is simply a ranking automatically applied to a policy once it's been created. Precedence 1 is the highest ranking policy and will thus override any other policies. Precedence 2 is the 2nd highest and so on.

User Profiles

User profiles are where all user-specific information is stored. Some examples of the data include, but are not limited to:

- Strike count
- Last login time
- Password expiration date
- Hashed answers to challenge question (using SHA-1)
- Last password change time

A profile will be created for each user automatically as they login through PortalGuard so it is not necessary to "preload" any user accounts.

3 Functionality

The PortalGuard API allows you to add all functionality available within PortalGuard to your own web applications. The lists below describe this functionality.

Password Recovery

Challenge Question & Answer Recovery - Configurable number of answers the user must provide to prove their identity and reset their forgotten password

Multiple Authentication Factors

Challenge Answer Required on Login - Allows for more robust authentication by requiring the user to provide a username, password & answer to a randomly chosen question they have previously answered to login

Login Session Concurrency

Prevent Concurrent Login Sessions - PortalGuard can prevent users from having more than one concurrent login session with the portal server. This is achieved by assigning the user a session identifier cookie that persists for the life of the browser. This session identifier is also stored in the user's profile. In order to clear the session from the user profile, the user must perform a logout on the website. If a user shuts down their browser without manually logging out, attempted logins with that account will not be permitted until the session is cleared from the user profile.

User Lockout

User Strikeouts - A configurable number of consecutive failed authentications will result in the user's account being locked. Once an account has reached the strike limit, no end-user operations can be performed on it (login, change password, reset forgotten password, etc). PortalGuard counts invalid passwords -AND- invalid challenge answers (if password recovery is enabled) as strikes. If not struck out, a successful authentication (either via password or challenge answers) will clear any strike count on the user's profile.

Strikeout Warnings - During authentication, the end user will see a message that a strike has been written to their account. The information includes how many strikes are on the account and the number remaining before the account is locked out.

Strikeout Expiration - To ease user frustration and administrative overhead, strikeouts can be configured to automatically expire. The number of minutes strikeouts should remain in effect is configurable in the policy. When this feature is enabled, user attempts to authenticate with a locked account will receive notification of the number of minutes they must wait before the account is unlocked.

Inactivity Lockout - A hold can be placed on user accounts based on a configurable number of days of inactivity. This hold prevents all actions. PortalGuard uses its capability to track last login times to enable this functionality. This setting does not make changes to the user account in LDAP to achieve this lock - it is only within the user's PortalGuard profile that the lock is enforced.

Password Expiration

Expiration Period - The number of days until the user will be forced to change their password through the PortalGuard interface. This period counts from the last password change through PortalGuard.

Grace Period - The number of days after a user's password expires that will still be allowed to login using the current password. When logging in with an expired password, the user will receive notification of the number of days until the grace period ends and have the option of continuing the login without changing their password. When the grace period ends, the user will be forced to change their password in order to login to the web server.

Expire On First Use - If desired, the first time a user logs in through the PortalGuard interface, they can

be required to change their password immediately. This is one way to force the user to comply with the password complexity rules enforced by PortalGuard.

Check Password Strength on Login - The other way to ensure compliance with the password rules configured through PortalGuard is to proactively check the password strength of user's passwords when used to login. This is a much more granular approach to ensuring password quality compliance because it only affects accounts with passwords that are not complex enough. If you are required to make a change to PortalGuard's password complexity rules, this setting will help ensure that users comply with the rule changes immediately.

Auditing

Log Last Login - Stores the date and time the user last logged in through PortalGuard. The value is always stored in Greenwich Mean Time (GMT).

Log Last Password Change - Stores the date and time the user last changed their password through PortalGuard. This is updated when the user changes their password by providing the current one or by resetting their password with challenge questions and answers. The value is always stored in Greenwich Mean Time (GMT).

Log Last Password Recovery - Stores the date and time the user last reset their password through PortalGuard using a password recovery mechanism. This can help show Return on Investment (ROI) by tracking how often users are able to restore their own access without calling the Help Desk. The value is always stored in Greenwich Mean Time (GMT).

Password Complexity Rules

PortalGuard maintains its own password settings apart from those for your user repository. It is recommended that you set PortalGuard's rules equal to or stricter than those for the user repository to ensure that passwords acceptable to PortalGuard are not considered unacceptable by the user repository.

Password History - PortalGuard can store hashes (using SHA-1) of passwords previously changed through its interface. This functionality allows specification of the number of entries to maintain in the history. This functionality follows a "first in, first out" (FIFO) mechanism for storing the hashes. Alternatively, the number of days worth of hashes can be specified to prevent users from iterating through permutations of their password to get back to their "favorite" one during the same day(s).

Minimum Length - The minimum number of characters a password must contain. This check can be skipped.

Maximum Length - The maximum number of characters a password may contain. This check can be skipped.

Minimum Lowercase - The minimum number of lowercase characters a password must contain. This check can be skipped.

Minimum Uppercase - The minimum number of uppercase characters a password must contain. This check can be skipped.

Minimum Numeric - The minimum number of numeric characters a password must contain. This check can be skipped.

Minimum Special - The minimum number of non-alphanumeric characters a password must contain. This check can be skipped.

4 JavaScript API

The highest-level interface for PortalGuard is the JavaScript API. It is best used when called through AJAX. As such, XML is returned and must be properly parsed and interpreted (all major browsers now support built-in XML parsers). Out of the box, PortalGuard uses this API to achieve its functionality.

Requirements

- PortalGuard is already setup on a web server
- If PortalGuard is not installed on the web server which users are authenticating too, then it should be in same DNS domain (e.g. *.acme.com) to prevent cross-site scripting JavaScript errors

Access Methods

doWSPAuth()
void doWSPAuth(objAuthForm, objFeedbackDiv)

This is the main entry point of this API. It utilizes AJAX to call PortalGuard on the back-end web server and its source code can be viewed in the "pg.js" JavaScript library file. It automatically calls a secondary function called *checkError()* to interpret the returned XML and handle updates to the UI and form to display feed back to the user.

Parameters & Return Values
Return value: None

objAuthForm - A JavaScript variable pointing to the HTML form containing the fields with the user's supplied username, password or other authentication-related data (e.g. challenge answer fields). All fields of type *text*, *hidden*, *password* and *select-one* will be POST-ed to the PortalGuard agent on the server.

objFeedbackDiv - A JavaScript variable pointing to the HTML "div" element where the informational or error message from the PortalGuard response will be displayed. The complete contents of the div (".innerhtml") will be overwritten but the div must already be visible for the message to display.

This function also uses the global *AUTHTYPE* variable to determine what type of action is to be performed. The actions are listed in the following table:

ACTION	Description
1	*Login* - Validates the provided username and password
2	*Set Password* - Changes the user's password if the current password is correct
3	*Set Answers* - Allows the user to set their answers to the challenge questions. Second phase (of 2) of the "Set Challenges" feature.
4	Reserved for future use
5	*Get Questions* - Retrieves the challenge questions. First phase (of 2) of the "Set Challenges" feature that first prompts the user for their name and password.
6	*Start Challenge Recovery* - Retrieve the challenge questions. First phase (of 3) of the "Use Challenges" feature that first prompts the user for just their username.
7	*Check Answers* - Validates the challenge answers for the given user. Second phase (of 3) of the "Use Challenges" feature.
8	*Reset Password* - Prompts the user to enter their choice for their new password after they have been authenticated using challenge answers. Third phase (of 3) of the "Use Challenges" feature.
9	*Login with Challenge* - Only used when the answer to a randomly chosen challenge question is required to login. Validates the username, password and provided answer.
10	*Logout* - For authenticated users, destroys their current session with PortalGuard. Only used with Concurrent Session prevention

Interpreting Results

The returned XML document has the following structure:

Top-level/root Element

<*pg_return*> - The lone root element of the XML document, there can be only one instance of it.
Value: None
Attributes:
version - String representing the version of PortalGuard on the server that created the response.
Direct children: See list of all First-level Children

First-level Children Elements

<*maj_error*> - The major return error as a number, there can be only one instance of this element.
Value: Numeric value representing the major error number, see corresponding table below.
Attributes: None
Direct children: None

<*min_errors*> - The container element for all minor error codes. There can only be a single instance of this element.
Value: None
Attributes:
count - Represents the number of minor error code child elements that follow
Direct Children: <*min_error*>

<*expiration*> - The container element for expiration related values. There can only be a single instance of this element.
Value: None
Attributes: None

Direct Children: <*days_until*>, <*days_grace*>

<*lockout*> - The container element for all lockout related values. There can only be a single instance of this element.
Value: None
Attributes: None
Direct Children: <*strikes*>, <*strikesleft*>, <*exp_mins*>, <*inactivity_thresh*>

<*pwquality*> - The container element for all password quality related values. There can only be a single instance of this element.
Value: None
Attributes: None
Direct Children: <*minlength*>, <*maxlength*>, <*minlower*>, <*minupper*>, <*minnumeric*>, <*minspecial*>

<*challenge*> - The container element for all challenge recovery related values. There can only be a single instance of this element.
Value: None
Attributes: None
Direct Children: <*questions*>

<*session_id*> - There can only be a single instance of this element.
Value: The randomly generated PortalGuard session ID to be assigned as a session cookie for the user
Attributes: None
Direct Children: None

Second-level Children Elements
<*maxlength*> - There can only be a single instance of this element.
Value: None
Attributes: *rule* - The maximum number of characters specified by the rule
Direct Children: None

<*minlower*> - There can only be a single instance of this element.
Value: None
Attributes: *rule* - The minimum number of lowercase characters specified by the rule
Direct Children: None

<*minupper*> - There can only be a single instance of this element.
Value: None
Attributes: *rule* - The minimum number of uppercase characters specified by the rule
Direct Children: None

<*minnumeric*> - There can only be a single instance of this element.
Value: None
Attributes: *rule* - The minimum number of numeric characters specified by the rule
Direct Children: None

<*minspecial*> - There can only be a single instance of this element.
Value: None
Attributes:
rule - The minimum number of non-alphanumeric characters specified by the rule
Direct Children: None

<*questions*> - There can only be a single instance of this element.
Value: None
Attributes:
count - The number of challenge questions that follow as children
Direct Children: <question>

Third-level Children Elements
<*question*> - There can be multiple instances of this element.
Value: The text of the challenge question
Attributes:
idx - The index number of the question
Direct Children: None

Major Error Codes

The major error codes are defined in "pg.js". For all failures, see the minor error code(s) that accompany the major error code in the XML response.

Name	Description
PGAPI_RC_NONE	No error - the *Login* action succeeded
PGAPI_RC_LOGIN_FAILED	The *Login* action failed
PGAPI_RC_PWCHANGE_SUCCESS	No error - the *Set Password* action succeeded
PGAPI_RC_PWCHANGE_FAILED	The *Set Password* action failed
PGAPI_RC_PWRECOVERY_SUCCESS	No error - the *Reset Password* action succeeded
PGAPI_RC_PWRECOVERY_FAILED	The *Reset Password* action failed
PGAPI_RC_CHECKLOGINANS_SUCCESS	No error - the *Login with Challenge* action succeeded
PGAPI_RC_CHECKLOGINANS_FAILED	The *Login with Challenge* action failed
PGAPI_RC_CHECKANS_SUCCESS	No error - the *Check Answers* action succeeded
PGAPI_RC_CHECKANS_FAILED	The *Check Answers* action failed
PGAPI_RC_GETQS_SUCCESS	No error - the *Get Questions* action succeeded
PGAPI_RC_GETQS_FAILED	The *Get Questions* action failed
PGAPI_RC_LOGOUT_SUCCESS	No error - the *Logout* action succeeded
PGAPI_RC_LOGOUT_FAILED	The *Logout* action failed
PGAPI_RC_SETANS_SUCCESS	No error - the *Set Challenges* action succeeded
PGAPI_RC_SETANS_FAILED	The *Set Challenges* action failed
PGAPI_RC_GENERAL_FAILURE	A general failure occurred
PGAPI_RC_PG_UNAVAILABLE	PortalGuard was reached on the server, but some component it depends on was unavailable

Minor Error Codes

The minor error codes give the detail about why the current action failed. They are also defined in the "pg.js" JavaScript library. There can be multiple minor error codes returned for a single action so they must all be parsed to determine the cause of the failure. See the table below for more details:

Minor Code	Description
PGAPI_RC_TRIAL_EXPIRED	Signifies that the PortalGuard trial has ended. Only occurs in trial versions of PortalGuard. Please contact sales@pistolstar.com to resolve this.
PGAPI_RC_AUTH_SERVER_UNAVILABLE	The LDAP server could not be reached from the server where PortalGuard is installed. Could represent a network issue with the PortalGuard server or the LDAP server is down or overloaded.

PGAPI_RC_LDAP_FAILURE	Returned if there was a protocol or operations error with the LDAP request made by PortalGuard. Could also signify that the schema was violated or that the identity used by PortalGuard does not have the appropriate level of access. Check the PortalGuard log for further details.
PGAPI_RC_LDAP_UNKNOWN	An unmapped LDAP error occurred. Check the PortalGuard log for further details.
PGAPI_RC_LDAP_DSA_UNWILLING_SSL	Only returned if a protected LDPA operation was being performed (e.g. a password change), but it was not requested over a secure/SSL connection. Verify the proper SSL port is used and that the UseSSL flag in the bootstrap configuration is enabled.
PGAPI_RC_NO_USERNAME_SUPPLIED	A blank username was received in the POST-ed data. This could also signify that the wrong form was passed to the PortalGuard API.
PGAPI_RC_NO_PASSWORD_SUPPLIED	A blank password was received in the POST-ed data, but one is required by the requested action. This could also signify that the wrong form was passed to the PortalGuard API.
PGAPI_RC_NO_NEWPW_SUPPLIED	A blank new password was received in the POST-ed data, but one is required by the requested action. This could also signify that the wrong form was passed to the PortalGuard API.
PGAPI_RC_NEWPWS_NOT_MATCH	The two new password fields received in the POST-ed data were not the same.
PGAPI_RC_PWEXPIRED	The user's password was expired. See the <*expiration*> XML element.
PGAPI_RC_BAD_USER	The username provided was not found in the user repository used by PortalGuard. This error code is not returned if strikeouts are enabled or if generic failure messages are enabled.
PGAPI_RC_BAD_PASSWORD	The password provided was not valid for the given username according to the user repository used by PortalGuard. This error code is not returned if strikesouts are enabled or if generic failure messages are enabled.
PGAPI_RC_GENERIC_BAD_LOGIN	Either the username could not be found in the user repository or the provided password was incorrect. Returned only when generic error messages are enabled. Useful when confirming usernames as valid is unwanted.

PGAPI_RC_STRIKE	The password provided was not valid for the given username according to the user repository used by PortalGuard. Only returned if strikeout are enabled and generic failure messages are not enabled. See the <*lockout*> XML element.
PGAPI_RC_BLANK_CHAL_ANSWER	The user was either setting their challenge answers or needed to provide a challenge answer to login, but at least one of the answer fields did not have a value.
PGAPI_RC_BAD_CHAL_ANSWER	The user was attempting to reset their password by answering challenge questions, but at least one of the challenge answers was incorrect.
PGAPI_RC_CHAL_ANSWER_NOT_COMPLEX	The user was attempting to set their challenge answers, but at least one of their answers was insufficiently complex. The answers must be at least 4 characters long and the first character cannot be repeated more than twice.
PGAPI_RC_CHAL_ANSWERS_NOT_SET	Returned if the user is attempting to login but they have not yet set their challenge answers and it is required to do so by policy. It can also be returned if the user is attempting to recovery their password by answering challenge questions but they have not yet set them.
PGAPI_RC_LOGIN_REQUIRES_CHAL	Returned if the user is required to answer a challenge question before logging in, but has not yet been prompted to do so. This distinction can only be made AFTER the user has provided a valid username and password. Without the username, PortalGuard cannot reference their associated policy and make the determination that they must answer a question to login.
PGAPI_RC_INACTIVITY_LOCKOUT	The user's account has been locked out due to inactivity. This is only returned if the Inactivity Lockout feature is enabled and the user has not logged in through PortalGuard for an extended period of time. See the <*inactivity_thresh*> XML element.
PGAPI_RC_TOO_MANY_SESSIONS	There is currently an active session from another browser or machine for this user and their attempted login was prevented because of this. This is only returned if the Prevent Concurrent Sessions feature is enabled.

PGAPI_RC_NOT_INITIALIZED	Only returned if PortalGuard could not initialize itself using the bootstrap configuration. Check it and the PortalGuard log file to help determine the problem.
PGAPI_RC_UNKNOWN	An unexpected or unknown error occurred. Typically only occurs if a low-level exception was caught. Check the PortalGuard log file for more information.
PGAPI_RC_INTERNAL_ERROR	An internal error occurred. Typically only occurs if a cryptographic operation fails or if a configuration error is present. Check the PortalGuard log file for more information.
PGAPI_RC_DOCUMENT_NOT_SAVED	A user profile or policy document could not be updated. Check the PortalGuard log file for more information.
PGAPI_RC_PW_TOO_SHORT	The new password provided by the user is not long enough. Check the <*minlength*> XML element.
PGAPI_RC_PW_TOO_LONG	The new password provided by the user is too long. Check the <*maxlength*> XML element.
PGAPI_RC_PW_INSUFF_LCASE	The new password provided by the user does not contain enough lowercase letters. Check the <*minlower*> XML element.
PGAPI_RC_PW_INSUFF_UCASE	The new password provided by the user does not contain enough uppercase letters. Check the <*minupper*> XML element.
PGAPI_RC_PW_INSUFF_NUMERIC	The new password provided by the user does not contain enough numeric letters. Check the <*minnumeric*> XML element.
PGAPI_RC_PW_INSUFF_SPECIAL	The new password provided by the user does not contain enough non-alphanumeric letters. Check the <*minspecial*> XML element.
PGAPI_RC_PW_PREVIOUSLY_USED	The new password provided by the user has already been used by them. Only returned when Password History functionality is enabled.