CA SiteMinder®

Advanced Password Services

12.52

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- Character Values—Updated with the default scores for international (multibyte) characters.

Contents

Chapter 1: Introduction 17

Advanced Password Services 18
- Change My Password (SmCPW) 19
- Checking at Authentication (SmAPS) 20
- Help Desk Support (APSAdmin) 21
- Forgotten Password Services (FPS) 21
- Application Programming Interface (APSAPI) 22
- Interfaces for Delegated Management Services (DMS2) 23

SiteMinder (Basic) Password Services & Advanced Password Services 23

Chapter 2: Installation & Configuration 31

Preparation & Prerequisites 31

Installation 32
- SiteMinder Policy Server 32
- Web Server – Change Password Interface 35
- Web Server – Forgotten Password Interface 38
- Web Server – Help Desk (APSAdmin) Interface 42
- Delegated Management Services (DMS2) Interface 45

Schema Changes 47
- ObjectClasses 48
- User Entries 48

LDAP Updates 49

ODBC Queries 49

Other User Directory Changes 49

Configuration 49
- Change Password Interface (SmCPW) 50
- Forgot (Forgotten Password User Interface Stub) 54
- APSAdmin (Help Desk Interface Stub) 55

Policy Database Configuration 55
- User Directory Credentials 55
- Change Password Page Configuration 56
- Forgotten Password Configuration 57
- Help Desk Interface Configuration 57
- Event Handling 58
- Authentication Scheme 60

Chapter 3: APS Configuration File (APS.CFG) 61

Overview 61
Format 62
Define Macros 63
    External Macro File 64
Setting Overrides 65
    User Classes 68
    Context Macros 69
    A note about "Old Style" overrides 69
    Performance 69
    Testing Expressions 70
File Sections 70
Settings That Can Appear Anywhere 71
    Define 71
General Settings 72
    Trace 72
    LDAP Write Back Information 74
    Other LDAP Settings 76
Password Content Settings 80
    Password Length 81
    Required Content 82
    Combinations of Character Types 85
    Other Content Requirements 89
Password Reuse 93
    Reuse Count 94
Change Required 94
    Percentage 95
Restricting Words from the User's Profile 95
    Attribute Match Maximum 96
Run-time Password Checking 98
    Auto Force Change 98
    Failure Tracking 99
    Expiring Password 108
    Account Expiration & Purge 111
Event Redirection 113
    Active Expression Parameters 113
Generational & Other Automatic Redirection 118
    Generational Redirect 119
Sending Mail 121
    Disabled User Mail 121
Mail Settings 125

[Mail] 126
Custom Logging 128
[Logging] 128
Custom Extensions 128
[Custom] 129
Invalid Password Dictionary 129
[Dictionary] 129
Password Complexity 130
[Complexity] 130
Field Mappings 133
Attribute Mapping 134
Attribute Suppression 135
Renaming LDAP Groups 135
LDAP Reverse Groups 135
ODBC Queries 136
[ODBC] 136
APSExpire 150
LDAP Directories 150
ODBC Directories 151
General FPS Settings 151
Audit Log 152
FPS LDAP Settings 154
FPS Identify Process 154
[FPS-Identify] 155
Collecting Identification Information 155
Error Cases 156
Form Processing 158
Controlling Frequency of Use 160
Preventing Attacks 163
FPS Verify Process 165
Forms 165
Error Conditions 167
Form Handling 169
FPS Change Password Process 175
[FPS-Change] 175
FPS Confirm Process 176
[FPS-Confirm] 176
FPS Errors 186
[FPS-Errors] 186
Handling FPS Mail Errors 187
[FPS-No Mail] 187

Chapter 4: User Directories: Schema, Storage and Capabilities 189

Schema & Storage 189
    Date Format 190
    Attribute/Column Names 190
Forgotten Password Services (FPS) Data 208
LDAP Directories 211
    Schema Updates 211
    Special Setup 214
    Native Password Policy Support 214
    Disabling/Enabling User Accounts 215
    Special APS Processing/Limitations 218
    Fault Tolerance & Performance 219
    Known Directory Idiosyncrasies 221
Microsoft Active Directories 223
    Schema Updates 223
    Special Setup 225
    Native Password Policy Support 225
    Disabling/Enabling User Accounts 226
    Special APS Processing/Limitations 229
    Fault Tolerance & Performance 231
    Known Directory Idiosyncrasies 233
Microsoft Windows Domains 233
    Schema Updates 234
    Special Setup 234
    Native Password Policy Support 234
    Disabling/Enabling User Accounts 235
    Special APS Processing/Limitations 236
    Fault Tolerance & Performance 237
    Known Directory Idiosyncrasies 237
ODBC (RDBMS) Directories 237
    Schema Updates 237
    Special Setup (Queries) 239
    Disabling/Enabling User Account 249
    Special APS Processing/Limitations 251
    Fault Tolerance & Performance 251
    Known Directory Idiosyncrasies 252
Password Replication/Synchronization 252

Chapter 5: Administration and Operations 255

Introduction 255
APS Processing during User Authentication 256

Password Lifetime 260
Backup Policy Servers 263
Special Case: Three Strikes, You're Out 263
SiteMinder's [Basic] Password Services 265
Using Persistent Cookies 266
  Account Expiration 266
  Account Inactivity Warning 266
  Password Expiration Warnings 266
  True password expiration 266
Authorization Mapping and AZRedirect 267
FPS Process Flow 267

Chapter 6: Change Password Interface (SmCPW) 271

Change Password Forms 271
  Customization without Custom Forms 272
  Handling Errors 273
Query String Options 273
Setup 275

Chapter 7: Help Desk Interface (APSAdmin) 277

User Selection 278
  Customization Options 278
User Display 282
  Customization Options 282
User Update 285
Specifying Form Content and Access Control 285
  Access Control 285
  User Directory "Portability" 286
  Specifying Attributes/Columns 287
  Specifying Groups 287
  Grouping into Sections 288
  Prompts 289
  Varying Content by User 290
  Varying Content by Context 290
  Using Self() 291
  Validating Password Resets 292
  Forcing Password Changes 292
  Disabling, Enabling & Redisabling 293
  Special Field & Group Handling 294
  Other Field & Group Handling 295
Running APSAdmin from the Command Line 295

Security Note....296

Chapter 8: Daily Processing (APSExpire) 297

APSExpire....297
APSExpire Event Log....299
Event Processing....300
Command Line Parameters....301
Performance Adjustments/Job Definitions....302
LDAP Directories....302
ODBC Directories....303
Performance....303

Chapter 9: Forgotten Password (FPS) Interface (Forgot) 305

Programming Notes....305
Initial Invocation....306
Identify Pages....307
URL....307
Missing URL....308
Multiple URL....308
Not Found URL....309
Disabled URL....309
Too Recently Used URL....310
Too Recently Attempted URL....310
Lockout URL....310
Verify Forms....311
URL....311
Missing URL....312
Retry URL....313
Invalid URL....313
No Data URL....314
Timeout URL....314
Change Pages....314
URL....315
Timeout URL....315
Confirm Pages....315
Error Pages....318
Mail Error Pages....318

Chapter 10: Application Programming Interface 319

Compiling & Linking....319

Run-Time Requirements 320
Function Descriptions 320
IsPasswordValid 320
IsMyPasswordValid 322
IsMyPasswordValid2 324
ChangeMyPassword 326
ChangeMyPassword2 328
Authenticate 330
APSAdminGetDirectory 331
APSAdminRead 333
APSAdminWrite 337

Chapter 11: Authentication Scheme 341

Configuring as a Wedge 341
Configuring for FPS 342

Chapter 12: Custom Logging Extension (SmAPSLog) 343

Chapter 13: Custom Extensions (SmAPSEx) 345

Chapter 14: Utilities 347

APSComplexity - Calculate Password Complexity 347
APSTestSettings - Check APS Configuration 348
Command Line Options 348
APSEncrypt - Encrypt an Administrator Password 349
APSMailTest - Test Email Settings 349
APSPing - Test connectivity 350
APSVersion - Display Module Version Numbers 350
APSXlateTest - Test Translation Settings 350
SmCPW - Load Test Password Changes 351
APSForcePWChg - Set Force Password Change Flag 352
SmPortalVfy - Verify SmPortal Configuration 353

Chapter 15: Integrating APS with User Management Tools 355

Integration 355
Enabling and Disabling Accounts 355
Ensuring Password Validity 356
Maintaining Forgotten Password Services (FPS) Questions/Answers 357
Triggering APS Events Properly 357

Delegated Management Services (DMS2) 358
DMS Workflow Library 358
Enabling and Disabling User Accounts 358
Integrating Forgotten Question and Answers 361
CA Identity Manager 362

Chapter 16: Using Email 363

Introduction 363
Simple Mail Transfer Protocol (SMTP) 363
Performance 364
Configuration 365
Email Templates 367
How APSMail Uses Templates 367
Macros 367
Lookups 368
Directives 369
Message Body 378
Strategies & Best Practices 378
SMTP (Mail) Servers 380
Using [REM] 380
IUsing [INCLUDE] 381
Internationalization/Localization 381
Addressing Mail 383
Return Addresses 383
Handling Returned Mail 384
Mail Macros 384
Best Practice Recommendation 386
Troubleshooting 387
The SiteMinder Console Log 387
The Send Log 387
The Communications Log 388
Formatting Problems 388
Delivery Problems 388
Limitations 389

Chapter 17: Internationalization 391

Introduction 391
SmCPW, APSAdmin & Forgot (User Interfaces) 393
SmCPW 393
APSAdmin 394
Redirection 394

Email 395

Chapter 18: About Password Security 397

Cracking Passwords 397
Keeping Passwords Safe 398
  DOs 398
  DON'Ts 398

Chapter 19: SmPortal/SmTransact Installation and Configuration 401

Introduction 401
  About Firewalls 402
Installation 402
  SmPortal - The Client Side 402
  SmTransact - The Policy Server Side 403
Operation 404
Configuration 405
  The Server Section 408
  The Agent Section 410
  The Service Section 411
Verification 412
  SmPortalTest 413
  SmPortalVfy 415
  APSPing 415

Chapter 20: Internationalization & Localization (APSXLate) 417

Introduction 417
Installation & Configuration 417
  Log Files 418
  APSXlateTest 418
Translation Files 419
  Translation File Format 420
  Comments 420
  Continuation Lines 420
  Header Keywords 420
  Button Definitions 421
  Translations 422

Chapter 21: Best Practices for Storing Legacy/Back-End Credentials 423

Executive Summary 423

Details 425
Updating Application Credentials 426
Updating User Credentials 426
Additional Advantages 426
Summary 427

Appendix A: How Do I? 429

How Do I: Set up passwords to expire? 430
How Do I: Customize the change password form? 432
How Do I: Configure "3 Strikes and You're Out"? 432
How Do I: Force users to change their password at next login? 434
How Do I: Determine why a user's authentication is rejected? 434
How Do I: Enable an account that APS has disabled? 435
How Do I: Retrieve the date & time that a user last logged in? 435
How Do I: Limit password changes to once per day? 435
How Do I: Eliminate the current password from the change password form? 436
How Do I: Use a backup LDAP server (replicant)? 436
How Do I: Use a consumer LDAP server (satellite)? 437
How Do I: Detect & Prevent FPS Misuse? 437
How Do I: Lock FPS (mis) users out of our site? 437
How Do I: Re-enable FPS for an account? 437
How Do I: Automatically login the user using FPS? 438
How Do I: Allow a user to select their own password during FPS? 438
How Do I: Retrieve statistics about FPS usage? 438
How Do I: Securely support FPS challenge questions? 439
How Do I: Get the FPS challenge questions into the User Directory? 441
How Do I: Integrate the APSAdmin interface into my own Identity Management application? 441
How Do I: Make the APS API return error codes instead of error text? 442

Appendix B: Troubleshooting 445

Connectivity Problems 446
Client Side (SmPortal) Errors 447
Server Side (SmTransact) Errors 449
SiteMinder Authentication Log 450
New passwords chosen by users are always rejected 450
User Accounts are mistakenly disabled 451
All (or many) accounts are disabled when a single user fails authentication 451
APS does not disable LDAP user accounts 451
Automated Email Notification does not send email 451
Windows NT Users cannot change passwords 452
"Change Password Service is not configured correctly" 452

Error reading user entry ("Not Found") during password change 452
User is asked to authenticate during the FPS process 452

Index 453

Chapter 1: Introduction

Passwords are the earliest and most common security and user verification mechanism; password cracking is the most common security hazard. The careless use and maintenance of passwords represents the greatest threat to the security of a network.

SiteMinder is designed to protect resources on your network, on the Internet, and on corporate intra- and extranets. SiteMinder authenticates users by interfacing to one or more user directories or Directory Services. SiteMinder can authenticate users stored in Windows NT Domains, LDAP-based directories (such as iPlanet's Directory Server), and several external relational databases. Interfaces to new Directory Service Providers are always under development, so you should contact your CA sales representative for support for other Directory Providers.

Passwords are stored in each directory, associated with each user. Most Directory Service Providers can enforce limitations on the content of passwords (password content policies) and can control the lifetime of passwords (password lifetime policies) and user accounts (account lifetime policies) to varying degrees. These controls are collectively called password policies.

Policies implemented by Directory Services Providers do not always fulfill a site's security requirements and using more than one directory service can create inconsistencies between directories. This can cause administrative headaches.

CA offers a SiteMinder add-on module to help administrators implement and enforce robust, consistent, flexible, and comprehensive password policies across multiple directories: Advanced Password Services (or APS).

This component can greatly enhance your web site's security by forcing your users to conform to administrator-defined rules for what a password should consist of and to control when, how, and how often a password must be changed.

SiteMinder Version 4.1 (and later) provides a feature called Password Services. This functionality is based on early versions of APS. It contains some of the basic functions provided by these earlier versions. Advanced Password Services 18 Introduction SiteMinder APS Administrator's Guide - 5.5 provides considerable additional functionality, plus the ability to extend this functionality for your site.

This section contains the following topics:

Advanced Password Services (see page 18)
SiteMinder (Basic) Password Services & Advanced Password Services (see page 23)

Advanced Password Services

SiteMinder invokes Advanced Password Services (APS) as each user attempts to login (authenticate), whether the login is valid or not. If the Directory Service and SiteMinder have determined that the user has successfully identified itself, APS is given the opportunity to do a final check. At that time, APS can determine that the password has expired or will expire shortly and whether the user has been inactive for too long.

If the user has failed to authenticate because an invalid password has been supplied, APS increments a counter associated with the user. Once this counter exceeds an administrator-supplied value, the user account can be disabled automatically.

APS can detect many such situations or events. Once APS has detected one of these events, it can take any of a number of actions, all configurable by the Administrator.

Many events may actually cause the user account to become disabled.

APS can automatically send email to users and/or administrators identifying that an event has occurred. Either the current on-line user can be redirected to another web page or a message can be displayed. All of this is under the full control of the Administrator.

APS can alert users and/or site administrators, via email notification, of any potential security breaches to their accounts and/or web sites. These breaches could be hackers attempting to access a valid user's account or a disabled (invalidated) user account attempting to gain access to a protected resource.

It is not uncommon for sites to configure APS to continue to send email when it detects ongoing attacks against an account. Typically, this mail is sent through Introduction 19 SiteMinder APS Administrator's Guide - 5.5 an SMTP pager gateway (not supplied) to notify a site's security administrators, *in real time*, that an ongoing attack is taking place.

By limiting the amount of idle time that can elapse before invalidating a user, administrators can prevent inactive users from becoming *back-door* logins to your system. For example, if a customer's employee leaves the job, the employee's account will retire immediately, before it can be used for mischief.

APS can also prevent hackers from gaining access via password-cracking tools by limiting the number of consecutive failed login attempts. At a predetermined number of consecutive failures, the accessed account will be disabled and the user (and/or administrator) notified by email.

Advanced Password Services capabilities can be broken out into several functional areas:

- Change My Password (SmCPW)
- Checking at Authentication (SmAPS)

- Help Desk Support (APSAdmin)
- Forgotten Password Services (FPS)
- Application Programming Interface (APSAPI)

Change My Password (SmCPW)

Advanced Password Services allows users to change their own passwords according to a comprehensive, flexible password content policy or *rules*. As with the rest of APS, this service is highly configurable by the site administrator.

The interface used to enter password changes is easily configurable by an HTML programmer. A simple, but complete, interface is provided with APS and limited branding may be performed without programmer intervention.

For those sites desiring more comprehensive integration, a full Application Programming Interface (API) is provided for password validation and change.

The administrator can create the password policy, or rules, that a new password must pass before a user may use it. These rules include:

- Require certain combinations of character types (alphabetic, numeric, punctuation and symbol characters) that a password must contain.
- Limit the number of passwords that must be used before a given value may be reused.
- Specify a minimum amount of time that must pass before a user may reuse a password.
- Check passwords against a dictionary of disallowed words.
- Disallow repeating characters in the password.
- Force the password to change by a minimum percentage from previous password.
- Enforce minimum and maximum password lengths.
- Prevent the use of information stored in the directory itself as all or part of a password (such as preventing the use of the user's first name in a password).

- Force the password to match, or not match, one or more patterns of characters.
- Require a certain level of password "complexity".

Using this service, administrators can impose limitations on newly entered passwords, regardless of which Directory Service the user is stored in. A consistent password security policy makes it considerably more difficult to break into a system using a password-cracking program.

By limiting the types of passwords that can be used (e.g., at least eight characters in any combination of letters and numbers, eliminating all entries in a dictionary, etc.), site security can be greatly enhanced.

Checking at Authentication (SmAPS)

The SiteMinder Authentication Service invokes SmAPS each time that a user attempts to authenticate, whether the authentication is successful or not. The purpose is to detect certain events that might occur, such as password expiration, and to act upon them.

SmAPS will log operational and informational messages to the SiteMinder Authentication Server Console Log. The information written to this log is often useful to understand and determine the activities being performed by APS.

SmAPS also supports a special extension library called SmAPSLog that can be used to customize and extend the logging capabilities of APS. The SmAPSLog library is supplied with APS in source code.

LDAP and Windows NT Domain directories support password policies of their own, whereas ODBC directories do not.. It is important that these functions either be disabled in the underlying directory or that their settings are *less* strict than the ones set within APS. The Directory Service Provider will perform its *lifetime* tests *before* APS has a chance to perform its tests. If the provider rejects the login, SiteMinder and APS will not know *why* the login was rejected and the configured actions (mail, redirection) will not be taken.

Help Desk Support (APSAdmin)

APS handles not only passwords, but user account enablement/disablement as well. A Help Desk tool, called APSAdmin, is supplied with APS starting at version 4 (it replaces an earlier command line utility called SmBlob).

APSAdmin is a fully configurable web-based interface that can be used by Help Desk.(and QA) users to maintain user entries in your User Directories. APSAdmin does not support Windows Domain Directories.

APSAdmin can be used as a stand-alone utility or can be integrated into an existing Help Desk system. While the interface is very flexible and its look and feel can be heavily customized, some sites may wish to implement its functionality themselves. Such code can call the APSAdmin functions of the APS Application Programming Interface (API). These functions transfer XML data between the caller and APS.

Forgotten Password Services (FPS)

APS can force users to change their passwords on a periodic basis and can force these passwords to be more complex than simple words. This is recommended for site security. However, it sometimes creates difficulties for users because they cannot necessarily use easy-to-remember passwords and they must change their passwords regularly. Thus, users *will* forget their passwords (and sometimes their login id!).

Most sites create some sort of Customer or User Help Desk. After some time, they realize that Help Desk Representatives spend much of their time resetting users' passwords that have been forgotten.

APS includes a solution to this problem called Forgotten Password Services (FPS). FPS provides a highly customizable mechanism for users to reset their own passwords without human intervention.

FPS is an *engine* that drives the password recovery process. It presents no HTML screens itself (except as a result of a communications error). However, it drives the presentation of site-written forms, processing user input and determining the next page to be presented. It handles all User Directory access and provides a high level of security within the process logic. Sample forms, both in ASP and JSP, are provided with the APS package.

Forgotten password recovery, as a process, is probably the most unsecure part of your site. Forgotten Password Services (FPS) tries to provide the capability in as secure a manner as possible. By using the FPS capabilities of APS, you can take advantage of the paranoia and experience of all of the other FPS users, rather than discovering the various gotchas and security holes on your own.

FPS is very flexible. Sites can use any or all of its features to control the security of the process.

For the most part, FPS itself only displays catastrophic error messages. All other displays are generated by site-supplied code. FPS primarily acts as a "traffic cop", directing the user from one page to another, based on user input.

The business logic for FPS runs in the SiteMinder Authentication Service process as part of APS. There are no additional modules to buy. A CGI stub, called Forgot (Forgot.exe on Windows), runs on the Web Server to act as a client on behalf of the user.

FPS is configured, once again behind the firewall, with knowledge of how to speak with the User Directory, how to search it, the names of pages (in the DMZ) that it can use to communicate with the user, and information about those pages.

At this time, the FPS component of APS only supports LDAP and ODBC directories.

Rogue users *will* attack your site. If you have anything of value on your site, somebody will want to get to it. Some rogue users do not even need *that* reason to hack a site; they will do it just to try.

Once such a user has targeted your site, they will look for weaknesses in the security. One of the first points of attack is that little button (or link) that says, "Forgot your password?"

Application Programming Interface (APSAPI)

APS includes three different Application Programming Interfaces (APIs).

- A forward-facing API (APSAPI) allows applications to determine if a given password conforms to the formatting rules defined for a specific user, change a user password, or perform APSAdmin-like functions using an XML-based interface. All APS utilities use this API. This API is provided with both a C++ and a COM interface. Due to the nature of the encrypted communications used by this API, a native Java interface is not possible. However, a JNI interface to the API can be built with minimal effort.
- A rearward-facing API (SmAPSLog) that APS calls during processing can be used to perform custom APS event logging. This is a published API that is described in this document. A sample of this library is provided in the source code.
- A rearward-facing API (SmAPSEx) can be used by CA Professional Services to customize APS behavior beyond the configurable options. This API is restricted, for security reasons, to CA Professional Services only.

Interfaces for Delegated Management Services (DMS2)

APS includes templates and interfaces for Delegated Management Services (DMS) product line. Sites can create custom self-registration, user self-service profile management and delegated user administration systems that communicate with APS to enforce password policies, manage forgotten password options and control user account enabling/disabling.

SiteMinder (Basic) Password Services & Advanced Password Services

Starting with Version 4.1, SiteMinder included Password Services. The functionality included is essentially a subset (with a few extensions) of the Advanced Password Services Version 1.1 functionality. Compared to APS Version 1.1, PS adds policies based on the user's location in the Directory and removes email support and support for Windows NT user directories.

Not every site requires Advanced Password Services. For some sites, the functionality provided by Password Services is sufficient.

It should be noted that there are no utilities for converting data from PS to APS or vice versa. Each system stores its information separately; they cannot access each other's data.

The following table compares the features of PS with this version of APS. Check with your CA Representative for current comparisons.

Feature	Basic PS (v6.0)	APS (v5.5)	Comment
Password format control			
Minimum Length	Yes	Yes	
Maximum Length	Yes	Yes	
Minimum Letters	Yes	Yes	
Minimum Uppercase Letters	Yes	Yes	
Minimum Lowercase Letters	Yes	Yes	
Minimum Digits	Yes	Yes	

Feature	Basic PS (v6.0)	APS (v5.5)	Comment
Minimum Alphanumeric	Yes	Yes	
Minimum Punctuation	Yes	Yes	
Minimum Symbols	No	Yes	
Minimum non-alphanumeric	Yes	Yes	
Maximum Repeat	Yes	Yes	
Minimum Type Combinations	No	Yes	
Complexity Thresholds	No	Yes	
Forced case (case-insensitivity)	Yes	Yes	
Regular Expression Match	Yes	Yes	
Regular Expression Forbid	Yes	Yes	
Allowed Characters List	No	Yes	
Disallowed Characters List	No	Yes	
Reuse Timer	Yes	Yes	
Reuse Counter	Yes	Yes	
Require Percentage Change	Yes	Yes	
Prevent use of Profile Values	Yes	Yes	APS can parse words in the profile and exclude specific attributes
Directory Support (see the CA Support Site for specific vendors & versions)			
LDAP Consumers	Yes	Yes	
Access to all values	No	Yes	Except password history
Requires Schema mods	No	Yes	
Exclude Accounts from Processing	No	Yes	

Feature	Basic PS (v6.0)	APS (v5.5)	Comment
Event Handling			
Number of Unique Events	1	7+	APS can selectively trap events
Redirect user	Yes	Yes	
Macro substitution into URL	No	Yes	Attribute substitution as well
Vary URL by realm/user	No	Yes	
Send Mail	No	Yes	
Macro substitution into Mail	N/A	Yes	Attribute substitution as well
Notify of Password Change	N/A	Yes	
Password Expiration			
Warning before expiration	Yes	Yes	APS can send warning without requiring a user login
Password Expires	Yes	Yes	
Grace period after expiration	No	Yes	
Grace logins after expiration	No	Yes	
Per-user overrides of period	No	Yes	
Account Expiration			
Disable at login	Yes	Yes	
Warn days before disabling	No	Yes	
Disable upon expiration	No	Yes	
Disable at specific date/time	No	Yes	

Feature	Basic PS (v6.0)	APS (v5.5)	Comment
Disable if no login by date/time	No	Yes	
Disable until specified date/time	No	Yes	
Per-user overrides of period	No	Yes	
Report when eligible for purge	No	Yes	
Arbitrary Account Disable	No	Yes	With custom reason codes
"n"-strikes, you're out processing			
Configurable number of strikes	Yes	Yes	
Works with LDAP outage	No	Yes	
Automatic reset after time	Yes	Yes	
Manual reset	Yes	Yes	
Permanently disable account	Yes	Yes	Optional in both cases
Notify Administrator	No	Yes	Via email event/pager interface
Notify Administrator of continuing attack	No	Yes	Via email event/pager interface
Configuration			
By location in LDAP DIT	Yes	Yes	
By arbitrary expression	Limited	Full	
Policies stored in	Policy Store	Flat File	
Simple Policy Configuration	Yes	Yes	
Cascading Policies	Yes	Yes	

Feature	Basic PS (v6.0)	APS (v5.5)	Comment
Easy migration of configuration information between environments	No	Yes	Meaning promotion from DEV to QA to Production environments by separating environment specific settings from configuration settings into separate files.
Password Change Forms			
Languages supported	Limited	Any	
Internationalized messages	No	Yes	
Customized messages	No	Yes	
User-initiated password change	Yes	Yes	
Redirect on error	No	Yes	
Administrator Interface			
Supports all product features	No	Yes	
Limit access to subsets of users	No	Yes	
Audited	Yes	Yes	
Can be externally accessible	No	Yes	
Can add custom attributes	No	Yes	
Attribute access by user	No	Yes	
Look & Feel configurable	No	Yes	By user, if desired
Can be tied into existing apps	No	Yes	
Tools Supplied			
Set Force Change Flag	Yes	Yes	

Feature	Basic PS (v6.0)	APS (v5.5)	Comment
Command line change password	No	Yes	
Other tools	None	7	
Per User Usage Statistics			
Available in responses	Limited	Yes	
Last Login Date	Available in responses	Yes	Includes IP address
Previous Login Date	Available in responses	Yes	Optional. Includes IP address
Last Password Change Date	No	Yes	
Last Failure Date	No	Yes	Optional. Includes IP address
Login History	No	Yes	Optional. Includes IP address
Failures since last login	No	Yes	Optional. Includes IP address
Failures since previous login	No	Yes	Optional. Includes IP address
Max failures between logins	No	Yes	Optional
Total Logins	No	Yes	Optional
Total Failures	No	Yes	Optional
Forgotten Password Usage	No	Yes	Optional
Forgotten Password Support			
Included with package	No	Yes	
User-selected password	N/A	Yes	
Automatically login at end	N/A	Yes	
Lockout with Counter	N/A	Yes	
Sample Forms	N/A	Yes	
Consumable questions	N/A	Yes	

Feature	Basic PS (v6.0)	APS (v5.5)	Comment
One-use passwords	N/A	Yes	
Secure new password delivery	N/A	Yes	
Encrypted/hashed answers	N/A	Yes	
Sample forms provided	N/A	asp/jsp	
SiteMinder Integration			
Policy Server different from Web Agent's Policy Server	No	Yes	
Failover Policy Servers	Yes	Yes	
Round-robin Policy Servers	Yes	Yes	
Configured through Policy GUI	Yes	No	
Integrates with DMS2	Yes	Yes	
Application Programming Interface	Limited	Yes	
Miscellaneous			
Custom Logging	No	Yes	Source provided
Custom Extensions	No	Yes	
Disabled groups	No	Yes	
Redirect at first/next login	No	Yes	
Message of the Day Service	No	Yes	

Chapter 2: Installation & Configuration

Advanced Password Services (APS) is an add-on component for SiteMinder and requires an existing SiteMinder installation. Some additional configuration of the SiteMinder Policy Server is also required.

Note: Installation Checklists for Solaris and Windows are supplied with the APS Installation kit as separate documents. This section exists for clarity and detail, but those documents break these instructions into concrete steps. You may wish to refer to those documents while reading this chapter.

This section contains the following topics:

Preparation & Prerequisites (see page 31)
Installation (see page 32)
Schema Changes (see page 47)
LDAP Updates (see page 49)
ODBC Queries (see page 49)
Other User Directory Changes (see page 49)
Configuration (see page 49)
Policy Database Configuration (see page 55)

Preparation & Prerequisites

Before you install Advanced Password Services, install and configure the following CA SiteMinder components:

- A Policy Server.
- A web agent or the Secure Proxy Server. Either component must be installed in front of the web server on which the APS client executables reside.

  Note: For more information about which component versions APS supports, see the Advanced Password Services Support Matrix.

Although a version of the SmTransact library is part of SiteMinder itself, APS requires a later version of this package. This version is included and installed with the APS installation kit.

The service name for the SmPortal.cfg file is smaps (case is significant).

See the chapter entitled SmPortal/SmTransact Installation and Configuration (see page 401) for information about how to configure and troubleshoot these modules.

See the chapter entitled Internationalization & Localization (APSXLate) (see page 417) for information about how to install, configure, and troubleshoot the APSXLate module.

See the instructions on how to configure the APS Mail (see page 363) module.

Installation

Note: Any instructions included in the readme or in release notes supercede instructions included here.

SiteMinder Policy Server

Windows

r12.5 and 12.51

The Policy Server installer installs the required shared library and all other required APS server files as part of the Policy Server installation.

Follow these steps:

1. Navigate to *siteminder_home*\bin and locate the following shared library file:

   `smaps.rename4aps.dll`

   siteminder_home

   Specifies the Policy Server installation path.

2. Rename the shared library to the following:

   `smaps.dll`

3. Restart the Policy Server to enable the Policy Server–side components.

   Note: The Policy Server installer installed all of the other required APS server files as part Policy Server installation.

r6.x and r12.0.x

Follow these steps:

1. Log in to the Policy Server host system.
2. Unzip smaps–5.5_*version*.zip in to a temporary directory.

   version

   Specifies the latest version of the zip.

3. Double–click the installation executable.

4. Follow the installer prompts and select the items that you would like to install. The program determines which parts of APS can be installed on the current system.

   Consider the following:

   ■ Any LANG (translation) or CFG (configuration) files are not replaced in production directories. However, new copies are always added to *siteminder_home*/Samples/APS_Translation_Files.

   ■ Only a default APS.cfg file is installed. Modify it to suit the needs of your site. If an APS.cfg file existed before installation, it is not overwritten. Locate the new APS.cfg file installed to SiteMinder\Docs.

5. Restart the Policy Server to enable the Policy Server–side components.

UNIX

r12.5 and 12.51

The Policy Server installer installs the required shared library and all other required APS server files as part of the Policy Server installation.

Follow these steps:

1. Log in to the Policy Server host system.
2. Navigate to *siteminder_home*/lib/ and locate the following shared library file:

   ```
   libsmaps_rename4aps.so
   ```

   siteminder_home

   Specifies the Policy Server installation path.

3. Rename the shared library to the following:

   ```
   libsmaps.so
   ```

4. Restart the Policy Server to enable the Policy Server–side components.

r6.x and r12.0.x

Each kit supplies two installation scripts. One script for the Policy Server–side components and one kit for web–side components. These scripts are supplied in a single file, tarred, and compressed.

The installation program does the following:

■ Backs up existing files by appending a timestamp as an extension.

■ Verifies that all text files have only UNIX style line endings.

- Copies files to the correct directories.
- Records the list of installed files in a logfile named:

  `siteminder_home/apsinstall_datestamp.log`

Follow these steps:

1. Decompress the supplied file and then untar it to gain access to the two scripts.
2. Log in to the Policy Server host system as the user who installed the Policy Server.
3. Run the Policy Sever shell script.

   Consider the following:

   - The installation program can locate the SiteMinder Policy Server directories automatically.
   - The shell script is a korn script. The script includes all installation files in the form of a uuencoded, compressed, tar file appended to the end of the script.
   - Existing files are backed up in place by appending a timestamp as an extension.
   - All LANG (translation), CFG (configuration), or sample source files are backed-up in place. If you changed a file, merge the changes into the new version of the file that is installed.
   - The default APS.cfg file is installed. Modify it as needed. If an APS configuration file existed before installation, the installer renames it with the date of installation as an extension. The new version is installed in the same directory. Modify the new version by merging changes forward from the old version. If this is a new installation, edit it manually as needed.
4. Restart the Policy Server to enable the Policy Server–side components.

Web Server – Change Password Interface

Windows

r12.5 and 12.51

The web agent installer installs the Change Password Interface client component. You do not have to use the APS installation kit to install it.

Follow these steps to complete the installation:

1. The default SmPortal.cfg file is installed. Modify the configuration file as needed.

   Note: For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

2. Map the *agent_home*\bin\web\CPW directory to your web server as a virtual CGI directory.

   Note: For more information, see your vendor–specific documentation.

r6.x and r12.0.x

Follow these steps:

1. Log in to the web agent host system.
2. Unzip smaps–5.5_*version*.zip in to a temporary directory.

   version

   Specifies the latest version of the zip.

3. Double–click the installation executable.
4. Follow the installer prompts and select Change Password Interface when prompted to select components.

   Note: You can install all web–side components at the same time.

5. Follow the remaining prompts to specify destination locations and installer options.

   Consider the following:

   - Existing LANG (translation) or CFG (configuration) files are not overwritten. New copies of these files are added to *agent_home*/Samples/APS_Translation_Files.
   - The default SmPortal.cfg file is installed.
   - If a Policy Server is installed on this system, some files are copied in to CA SiteMinder® directories instead of web agent directories.

6. Restart the host system to complete the installation.

After you complete the installation:

1. Modify the SmPortal configuration file to meet the needs of your organization. Verify that the file references a service named smaps.

   Note: For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

2. Map the *agent_home*\bin\web\CPW directory to your web server as a virtual CGI directory.

   Note: For more information, see your vendor–specific documentation.

UNIX

r12.5 and 12.51

The web agent installer installs the Change Password Interface client component. You do not have to use the APS installation kit to install it.

Follow these steps to complete the installation:

1. The default SmPortal.cfg file is installed. Modify the configuration file as needed.

   Note: For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

2. Define a virtual CGI directory for the directory that contains the SmCPW CGI program.

   Note: For more information, see your vendor–specific documentation.

3. Create the following environment variables in the CGI process space of the web server:

   SMPORTAL

   The full path to the SmPortal.cfg file.

   APS_LANG_PATH

   The path to the Language directory.

   Note: Update LD_LIBRARY_PATH to let SmCPW find the SmPortal library. For more information, see your vendor–specific documentation.

   Important! Do not put the definitions in to a user profile. Place them in to a special configuration file that the web server uses.

r6.x and r12.0.x

Each kit supplies two installation scripts. One script for the Policy Server–side components and one kit for web–side components. These scripts are supplied in a single file, tarred, and compressed.

The installation program does the following:

- Backs up existing files by appending a timestamp as an extension.
- Verifies that all text files have only UNIX style line endings.
- Copies files to the correct directories.
- Records the list of installed files in a logfile named:

  `agent_home/cpwinstall_datestamp.log`

Follow these steps:

1. Decompress the supplied file and then untar it to gain access to the two scripts.
2. Log in to the web agent host system as the user who installed the agent.
3. Run the agent shell script.

   Consider the following:

   - The installer requires you to enter the absolute path to the parent directory of the web agent.

     Example: If you installed the agent in /opt/smuser/ca/webagent, enter the following when prompted:

     ```
     /opt/smuser/ca
     ```

   - The shell script is a korn script. The script includes all installation files in the form of a uuencoded, compressed, tar file appended to the end of the script.
   - All LANG (translation), CFG (configuration), and sample source files are backed-up by appending a timestamp as an extension. If you changed a file, merge the changes into the new version of the file that is installed.
   - The default SmPortal.cfg file is installed.
4. Restart the system to complete the installation.

After you complete the installation:

1. If this is a new installation, modify SMPortal configuration file as needed. For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

   If an SmPortal configuration file existed before the installation, the installer renames it with the date of installation as an extension. The new version is installed in the same directory. Modify the new version by merging changes forward from the old version. Verify that the file references a service named smaps.
2. Define a virtual CGI directory for the directory that contains the SmCPW CGI program.

   Note: For more information, see your vendor–specific documentation.

3. Create the following environment variables in the CGI process space of the web server:

   SMPORTAL

   The full path to the SmPortal.cfg file.

   APS_LANG_PATH

   The path to the Language directory.

   Note: Update LD_LIBRARY_PATH to let SmCPW find the SmPortal library. For more information, see your vendor–specific documentation.

   Important! Do not put the definitions in to a user profile. Place them in to a special configuration file that the web server uses.

Web Server – Forgotten Password Interface

Windows

r12.5 and 12.51

The web agent installer installs the Forgotten Password Interface client component. You do not have to use the APS installation kit to install it.

Follow these steps to complete the installation:

1. The default SmPortal.cfg file is installed. Modify the configuration file as needed.

   Note: For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

2. Map the *agent_home*\bin\web\FPS directory to your web server as a virtual CGI directory.

   Note: For more information, see your vendor–specific documentation.

3. The installation provides sample FPS presentation forms. Create, modify and install the required forms. After you create the forms, configure them in the APS configuration file (APS.cfg) on the Policy Server host systems.

   Note: For more information, see the Forgotten Password Interface chapter.

r6.x and r12.0.x

Follow these steps:

1. Log in to the web agent host system.
2. Unzip smaps–5.5_version.zip in to a temporary directory.

   version

   Specifies the latest version of the zip.

3. Double–click the installation executable.
4. Follow the installer prompts and select Forgotten Password Interface when prompted to select components.

   Note: You can install all web–side components at the same time.

5. Follow the remaining prompts to specify destination locations and installer options.

   Consider the following:

   ■ Existing LANG (translation) or CFG (configuration) files are not overwritten. New copies of these files are added to the Docs directory for reference.

   ■ The default SmPortal.cfg file is installed.

   ■ If a Policy Server is installed on this system, some files are copied in to CA SiteMinder® directories instead of web agent directories.

6. Restart the host system to complete the installation.

After you complete the installation:

1. Modify the SmPortal configuration file to meet the needs of your organization. Verify that the file references a service named smaps.

   Note: For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

2. Map the *agent_home*\bin\web\FPS directory to your web server as a virtual CGI directory.

   Note: For more information, see your vendor–specific documentation.

3. The installation provides sample FPS presentation forms. Create, modify and install the required forms. After you create the forms, configure them in the APS configuration file (APS.cfg) on the Policy Server host systems.

   Note: For more information, see the Forgotten Password Interface chapter.

UNIX

r12.5 and 12.51

The web agent installer installs the Forgotten Password Interface client component. You do not have to use the APS installation kit to install it.

Follow these steps to complete the installation:

1. The default SmPortal.cfg file is installed. Modify the configuration file as needed.

   Note: For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

2. Define a virtual CGI directory for the directory that contains the Forgot CGI program.

   Note: For more information, see your vendor–specific documentation.

3. Create the following environment variables in the CGI process space of the web server:

   SMPORTAL

   The full path to the SmPortal.cfg file.

   APS_LANG_PATH

   The path to the Language directory.

   Note: Update LD_LIBRARY_PATH to let SmCPW find the SmPortal library. For more information, see your vendor–specific documentation.

   Important! Do not put the definitions in to a user profile. Place them in to a special configuration file that the web server uses.

4. The installation provides sample FPS presentation forms. Create, modify and install the required forms. After you create the forms, configure them in the APS configuration file (APS.cfg) on the Policy Server host systems.

   Note: For more information, see the Forgotten Password Interface chapter.

r6.x and r12.0.x

Each kit supplies two installation scripts. One script for the Policy Server–side components and one kit for web–side components. These scripts are supplied in a single file, tarred, and compressed.

The installation program does the following:

- Backs up existing files by appending a timestamp as an extension.
- Verifies that all text files have only UNIX style line endings.

- Copies files to the correct directories.
- Records the list of installed files in a logfile named:

  `agent_home/fpsinstall_datestamp.log`

Follow these steps:

1. Decompress the supplied file and then untar it to gain access to the two scripts.
2. Log in to the web agent host system as the user who installed the agent.
3. Run the agent shell script.

   Consider the following:

   - The installer requires you to enter the absolute path to the parent directory of the web agent.

     Example: If you installed the agent in /opt/smuser/ca/webagent, enter the following when prompted:

     ```
     /opt/smuser/ca
     ```

   - The shell script is a korn script. The script includes all installation files in the form of a uuencoded, compressed, tar file appended to the end of the script.
   - All LANG (translation), CFG (configuration), and sample source files are backed-up by appending a timestamp as an extension. If you changed a file, merge the changes into the new version of the file that is installed.
   - The default SmPortal.cfg file is installed.
4. Restart the system to complete the installation.

After you complete the installation:

1. If this is a new installation, modify SMPortal configuration file as needed. For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

   If an SmPortal configuration file existed before the installation, the installer renames it with the date of installation as an extension. The new version is installed in the same directory. Modify the new version by merging changes forward from the old version. Verify that the file references a service named smaps.
2. Define a virtual CGI directory for the directory that contains the Forgot CGI program.

   Note: For more information, see your vendor–specific documentation.
3. Create the following environment variables in the CGI process space of the web server:

   SMPORTAL

   The full path to the SmPortal.cfg file.

APS_LANG_PATH

The path to the Language directory.

Note: Update LD_LIBRARY_PATH to let SmCPW find the SmPortal library. For more information, see your vendor–specific documentation.

Important! Do not put the definitions in to a user profile. Place them in to a special configuration file that the web server uses.

4. The installation provides sample FPS presentation forms. Create, modify and install the required forms. After you create the forms, configure them in the APS configuration file (APS.cfg) on the Policy Server host systems.

   Note: For more information, see the Forgotten Password Interface chapter.

Web Server – Help Desk (APSAdmin) Interface

Windows

r12.5

The web agent installer installs the Help Desk Interface client component. You do not have to use the APS installation kit to install it.

Follow these steps to complete the installation:

1. The default SmPortal.cfg file is installed. Modify the configuration file as needed.

   Note: For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

2. Map the *agent_home*\bin\web\APSAdmin directory to your web server as a virtual CGI directory.

   Note: For more information, see your vendor–specific documentation.

r6.x and r12.0.x

Follow these steps:

1. Log in to the web agent host system.
2. Unzip smaps–5.5_version.zip in to a temporary directory.

   version

   Specifies the latest version of the zip.

3. Double–click the installation executable.
4. Follow the installer prompts and select Help Desk Interface when prompted to select components.

   Note: You can install all web–side components at the same time.

5. Follow the remaining prompts to specify destination locations and installer options.

   Consider the following:

   - Existing LANG (translation) or CFG (configuration) files are not overwritten. New copies of these files are added to the Docs directory for reference.
   - The default SmPortal.cfg file is installed.
   - If a Policy Server is installed on this system, some files are copied in to CA SiteMinder® directories instead of web agent directories.

6. Restart the host system to complete the installation.

After you complete the installation:

1. Modify the SmPortal configuration file to meet the needs of your organization. Verify that the file references a service named smaps.

   Note: For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

2. Map the *agent_home*\bin\web\APSAdmin directory to your web server as a virtual CGI directory.

   Note: For more information, see your vendor–specific documentation.

UNIX

r12.5

The web agent installer installs the Help Desk Interface client component. You do not have to use the APS installation kit to install it.

Follow these steps to complete the installation:

1. The default SmPortal.cfg file is installed. Modify the configuration file as needed.

   Note: For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

2. Define a virtual CGI directory for the directory that contains the APSAdmin CGI program.

   Note: For more information, see your vendor–specific documentation.

3. Create the following environment variables in the CGI process space of the web server:

   SMPORTAL

   The full path to the SmPortal.cfg file.

   APS_LANG_PATH

   The path to the Language directory.

Note: Update LD_LIBRARY_PATH to let SmCPW find the SmPortal library. For more information, see your vendor–specific documentation.

Important! Do not put the definitions in to a user profile. Place them in to a special configuration file that the web server uses.

r6.x and r12.x

Each kit supplies two installation scripts. One script for the Policy Server–side components and one kit for web–side components. These scripts are supplied in a single file, tarred, and compressed.

The installation program does the following:

- Backs up existing files by appending a timestamp as an extension.
- Verifies that all text files have only UNIX style line endings.
- Copies files to the correct directories.
- Records the list of installed files in a logfile named:

  `agent_home/APSAdmininstall_datestamp.log`

Follow these steps:

1. Decompress the supplied file and then untar it to gain access to the two scripts.
2. Log in to the web agent host system as the user who installed the agent.
3. Run the agent shell script.

   Consider the following:

   - The installer requires you to enter the absolute path to the parent directory of the web agent.

     Example: If you installed the agent in /opt/smuser/ca/webagent, enter the following when prompted:

     ```
     /opt/smuser/ca
     ```

   - The shell script is a korn script. The script includes all installation files in the form of a uuencoded, compressed, tar file appended to the end of the script.
   - All LANG (translation), CFG (configuration), and sample source files are backed-up by appending a timestamp as an extension. If you changed a file, merge the changes into the new version of the file that is installed.
   - The default SmPortal.cfg file is installed.
4. Restart the system to complete the installation.

After you complete the installation:

1. If this is a new installation, modify SMPortal configuration file as needed. For more information, see the SmPortal/SmTransact Installation and Configuration chapter.

   If an SmPortal configuration file existed before the installation, the installer renames it with the date of installation as an extension. The new version is installed in the same directory. Modify the new version by merging changes forward from the old version. Verify that the file references a service named smaps.

2. Define a virtual CGI directory for the directory that contains the APSAdmin CGI program.

   Note: For more information, see your vendor–specific documentation.

3. Create the following environment variables in the CGI process space of the web server:

   SMPORTAL

   The full path to the SmPortal.cfg file.

   APS_LANG_PATH

   The path to the Language directory.

   Note: Update LD_LIBRARY_PATH to let SmCPW find the SmPortal library. For more information, see your vendor–specific documentation.

   Important! Do not put the definitions in to a user profile. Place them in to a special configuration file that the web server uses.

Delegated Management Services (DMS2) Interface

APS Attributes/Enable/Disable (all platforms)

From this point, the installation of this add-on component requires manual steps. After installing DMS, please make a backup of the following eight files (as mentioned in sections 1, 2, and 3 below) before copying them to the directories mentioned.

Note: *working-dir* is the directory where SiteMinder is installed.

1. Copy the following modified files into the *working-dir*/DMS/pages directory:

   - inc_actions.jsp
   - inc_org_roles.jsp
   - manage_data.js
   - manage_data_submitter.jsp

- user_submitter.jsp
- user_data_collector.jsp

2. Copy the following modified file into the *working-dir*/DMS/properties/default/locale directory:
   - dms_en_US.properties
3. Copy the following modified file into the *working_dir*/DMS/properties/default/tables directory:
   - user.properties
4. Copy the following new files (5 files) into the *working-dir*/DMS/pages directory:
   - aps.js
   - disable.jsp
   - apsblob.jsp
   - aps_attribute.jsp
   - disableGroup.jsp

Forgotten Password Questions/Answers (all platforms)

1. Copy the following modified files located under the pages subdirectory into the *working-dir*/DMS/pages directory:
   - forgottenpassword.jsp
   - newuserjs.jsp
2. Copy the following modified files located under the forms subdirectory into the *working-dir*/DMS/forms directory:
   - loginandregisterwithforgottenpassword-dms-i.fcc
   - loginandregisterwithforgottenpassword-dms.fcc
3. Copy the modified file Verify.jsp, located under pages directory, into the directory as mentioned in the URL setting in the [FPS-Verify] of the APS Configuration File.
4. Copy the jarfile questions.jar into the directory *working-dir*/DMS/java.
5. Put the required property file related to forgotten questions into the directory *working-dir*/DMS/properties/default/locale.

   The generic name format of the question property file is: questions_*language_country*.properties, where *language* is the LANGUAGE concerned and *country* is the COUNTRY concerned. For example the name of the property file may look like questions_en_US.properties, where en signifies the language "English" and US signifies the country "USA".
6. Configure the **Servlet** by adding the following in the servlet classpath:
   ```
   working-dir/DMS/java/questions.jar
   ```

Schema Changes

After APS Version 2.x, APS no longer uses an existing LDAP attribute to hold all of its data for a given user (the "Blob"). It now stores its data in separate attributes with specific names that need to be added to your LDAP schema. These attributes are described in detail in the chapter entitled User Directories: Schema, Storage and Capabilities (see page 189).

Version 4.0 of APS requires a new attribute, smapsNextAction, that did not exist in APS 3.0. APSExpire will automatically set this attribute the first time that it is run.

Note: APSExpire, while its performance is greatly enhanced with Version 4, will still take some time to perform the process of initializing smapsNextAction for the entire directory. In your rollout schedule, plan for this additional time. Since the performance enhancements for APSExpire in Version 4 involve the use of smapsNextAction (which is not yet set at this point), do not use timings for this initial APSExpire run for planning for later operations.

The installation kit installs two .conf files to the APS_Docs directory. These files describe, in iPlanet (Netscape) terms, the schema changes required. They can be used (by an iPlanet LDAP Server-savvy administrator) to automate the update to the schema on **iPlanet LDAP Version 4.x** servers. These files should be appended to the end of the existing slapd.user_at.conf and slapd.user_oc.conf files (be sure to back up the existing files first). After appending the files, restart your LDAP server.

For iPlanet Directory Server version 5.0 (and later), these .conf files cannot be used. Instead, you *must* use the 56NetegrityAPS.ldif file installed onto the SiteMinder/APS_Docs directory. To use this file, perform the following steps:

1. Edit the 56NetegrityAPS.ldif file using your favorite editor.
   a. Locate the two instances of smapsInfo (case-insensitive). Rename this class to the name of the class that you want to use. Note that if you want to *add* the APS attributes to an existing class, you will have to *remove* this class definition and add the attributes to your chosen class in the file in which it is defined (probably 99user.ldif).
   b. Save the file.
2. Locate the file 99user.ldif in your iPlanet directory. Make sure that you locate the one associated with the desired LDAP directory instance.
3. Put the modified 56NetegrityAPS.ldif file into the same directory.
4. Restart the LDAP Directory Server.

These steps only need to be performed on a single Master Directory. The changes will be propagated to other masters and consumers (though the changes will be recorded in subsidiary servers in the 99user.ldif file).

The schema can also be updated manually, using the user interface provided with your LDAP implementation (this will certainly be required for non-LDAP native implementations, such as Active Directories and X.500 implementations). The chapter on the Schema defines each of the required attributes.

ObjectClasses

It is not sufficient to merely add the new attributes to your schema. You must also make these attributes available for your user objects. This can be done in three different ways.

1. Create the smapsInfo objectClass with all of the new attributes.
2. Add the new attributes to your existing custom user object class, if one exists.
3. Add the new attributes to an existing standard user object class (such as inetOrgPerson). This is not desirable and some LDAP implementations (iPlanet/Netscape for one) forbid changing the standard object classes.

Your choice of the three options may have serious implications. You should read the next several sections before deciding.

You may be able to use the supplied smaps.user_oc.conf file to create a smapsInfo object class, or modify it to create a new object class with a different name. Use the file as described in the previous section. This will only work for iPlanet (Netscape) directories *prior* to version 5. Other implementations will require that you perform the object class updates manually. All attributes for the object class should be marked in the schema as optional (as opposed to mandatory).

User Entries

Just because the schema is updated does not mean that the changes are available for your user entries. The object class that allows the attributes must be part of *every* user entry in the system. If you have an existing object class that is already referenced by every user entry, then you should add the attributes to that object class.

Something to consider is your user maintenance tool. When creating new user entries, the tool must create new entries with the object class. Failure to do so will prevent APS from working with the new users, since it will be unable to update the user's entry as required.

LDAP Updates

Even if your site is *not* upgrading from a prior version of APS, it will still be necessary to run the APSExpire utility against your existing LDAP directory *after* updating your schema. APSExpire will update all of the users in your directory, initializing the smapsBaseDate and smapsNextAction attributes.

ODBC Queries

If your site is using an ODBC (RDBMS) User Directory, you will also need to update the ODBC Queries used by APS to read, add, update and delete information from the directory. The queries are defined in APS Configuration File (APS.CFG) (see page 61).

Other User Directory Changes

You must ensure that *every* new LDAP and ODBC user has access to the new attributes in order for APS to function properly. If you are using some sort of User Administrator Tool, such as Distributed Management Services (DMS2) or CA Identity Manager, you need to ensure two things:

1. Every new user is created with the objectclass that allows access to the new attributes.
2. smapsBaseDate is initialized to the creation date of the user. APSExpire will do this for you, but it will have to run daily if you wish these dates to be correct.

Configuration

APS is configured using a standard text editor. Configuration settings are stored in a file called APS.cfg. Under Windows NT, this file must be located in the same directory as the SMAPS library (SmAPS.dll, installed under the SiteMinder/Bin directory). Under Solaris, this file is pointed to by an environment variable called APS_SETTINGS. This environment variable points to the file itself, not just the directory containing the file. The configuration file controls all behavior of APS.

Under Solaris, if the APS_SETTINGS environment variable is not set, APS will look for the file at $SMHOME/siteminder/bin/APS.cfg.

If the file does not exist, an error will be logged to the SiteMinder console and default settings will be used. If any parsing errors occur, such errors will be written to a file with the same name as the APS.cfg file, with the date and .LOG appended to the file name. This file will be placed in the same directory with the APS.cfg file. If the process does not have the rights to create this file, no file will be created. If errors are logged to this file, APS will display a warning in the SiteMinder console log identifying that such errors were logged.

The format of the APS configuration file is described, in detail in the chapter APS Configuration File (APS.CFG) (see page 61).

Users may become frustrated attempting to find a new password if they are not presented with all of the criteria that a password must meet ahead of time. It is possible to restrict permissible passwords such that it is extremely difficult or even impossible for users to randomly pick a valid password. For this reason, we do not recommend setting all Advanced Password Service password policy options without care and need.

Change Password Interface (SmCPW)

SmCPW (SmCPW.EXE on Windows) is a CGI program that can be accessed through a Web Server so that end users can request a password change. This program can be installed on any Web-agent protected Web Server. The Web Agent need not be installed on the *same* Web Server with SmCPW. However, the Web Server upon which SmCPW resides must be protected by a Web Agent. This Web Agent may be plugged in to the local Web Server, may be the Secure Proxy Server or some similar configuration.

SmCPW communicates with the SmAPS library (running on a SiteMinder Policy Server) via the APS API, which uses CA's SmPortal/SmTransact communications module.

When running under a Web Server, SmCPW runs in two modes: GET and POST, defined by the type of request from the client browser. In GET mode, SmCPW displays a default HTML form for changing the password. In POST mode, SmCPW calls the APS API, passing the data posted by the change password form (either the default form or a custom form). The APS API uses CA's SmPortal/SmTransact module to communicate with the SmAPS library (running on the SiteMinder Policy Server). SmAPS checks the password and changes it, if allowed. If an error occurs, it will be returned (reversing the same communications path) to SmCPW and a message displayed to the user.

SmCPW can also be run from the command line (not under a Web Server). If run with no arguments (or just switches), SmCPW simulates a GET request from the browser, displaying the default form (command line arguments can be used to specify the language for the default form). Additional command line arguments can be used to simulate a POST operation. This allows a site to automate testing without using a browser.

If a new password is the same as the previous password, or the password is invalid, reused, insecure, or fails to pass APS's many tests for any other reason, the user is presented with an error message and asked to try a different password. The error message is not a concatenated list of problems; rather, it presents the first encountered problem with the password, such as "New password exceeds maximum length". It is possible that a user will have to try several passwords before finding one that meets all of the criteria.

All text that is displayed by SmCPW is subject to translation using two different APSXlate translation files. SmCPW.lang exists as a file on the Web Server and contains translations for all locally generated messages (including password confirmation) and for text to be displayed on the default form. APS.lang exists on the Policy Server and it contains translations for all messages that can be generated during password validation and update.

By default, SmCPW displays error and confirmation messages in a JavaScript message box (or, for browsers that do not support JavaScript, on a standard HTML page). If the translated message *looks* like a URL (starts with "http:", "https:", or "/"), SmCPW will redirect the browser to the URL instead of displaying the message.

This URL can have replacement parameters contained within it. SmCPW will process its own argument list as a set of key/value pairs. A target URL is examined for values contained within angle brackets ("<" and ">"). The text between the brackets is used as a key and replaced, along with the angle brackets, with the value associated with the key. Thus, values, including the originally requested URL, or *target*, can be passed to the page that will display the message.

Specific examples are included later in this section.

When running under a Web Server, the syntax of SmCPW is as follows. Note that APS can and will automatically construct these arguments if the redirection events are properly configured (see *The APS Configuration File*).

```
SmCPW{.exe}?
    [Optional]
    [&][DaysLeft=<daysleft>]
    [&][GraceLogins=<grace logins left>]
    [&][DaysRemaining=<days remaining>]
    [&][CancelTo=<cancelURL>]
    [&][Target=<targetURL>]
    [&][<additional args>]
```

The .exe component is required on Windows.

If "Optional" is specified, then the default form displayed by SmCPW will have a "Cancel" button. If the user clicks this button, the browser will be redirected back to the page that invoked the form (or <cancelURL>, if specified). The "Cancel" button is dependent on JavaScript support by the browser. If the Browser does not support JavaScript, a standard hyperlink labeled "Cancel" will be displayed instead.

DAYSLEFT=<*daysleft*> may be supplied to indicate how long the password has left before it expires. There are two cases:

- The argument is not supplied at all. SmCPW will consider that the password *must* be changed. By default, no additional messages are displayed. No "Forget it" or "Cancel" buttons are supplied.
- DAYLEFT=<*days left*> is supplied. The password *may* be changed and there is a time limitation. The "Forget It" or "Cancel" button is displayed (as if "Optional" were specified) in the default form. A message indicating that the password must be changed within "<*days left*> days" is displayed by the default form.

Example:

The TARGET=<*target URL*> argument identifies the page to which the user should be directed upon completion of the password change. The <*target URL*> should be passed in a URL-encoded format to avoid problems with embedded characters.

GRACELOGINS and DAYSREMAINING are used when the password has expired, but a grace period or a number of grace logins is configured. SmCPW will modify its displayed text to communicate these values to the user.

Additional key/value pairs may also be supplied, each set separated from its predecessor by an ampersand ("&"). These key/value pairs will be parsed and saved by SmCPW.

When SmCPW redirects to the next page, either by redirecting the specified target or to a custom message page, these key/value pairs will be used to replace macros in the target of the redirection. The URL is examined for text between angle brackets ("<" and ">"). The text and the angle brackets are replaced by the value associated with the text.

For example, if the SmCPW invocation is:

```
SmCPW?Target=%2FMyPage%2Easp%3F<lang>&Lang=EN
```

would be redirected to:

```
/MyPage.asp?EN
```

When APS builds the redirection because of an event, it can construct complex URLs with such query strings and can include user attributes in that query string. See the section about Redirections later in this document.

Setup

SmCPW must be placed into a directory that is visible to a SiteMinder Web Agent protected Web Server. This directory must have *execute* privileges (under Microsoft's Internet Information Server, use the Microsoft Management Console, under Netscape Enterprise Server, modify the obj.conf or magnus.conf file to create a CGI-bin directory). The installation program will usually place this file into a directory called

```
SiteMinder Web Agent/Bin/Web/CPW.
```

It is not necessary that the program actually be called SmCPW.EXE (though for troubleshooting purposes, it is desirable that it be so named). You may rename the program (maintaining the .EXE extension on Windows) to anything that you desire. For the remainder of this document, the program will be referred to as SmCPW.

A response attribute may be required to be passed when posting to SmCPW. This attribute explicitly identifies the user changing the password. With normal Web Agent configuration, this response is not needed (versions of APS prior to version 4 *required* this response). If SmCPW complains that it cannot identify the user, then this response will be required. If this response exists, SmCPW will use it, so it does not hurt to have it set up even when it is not needed.

See your SiteMinder documentation for how to set up Response Attributes. The active expression required is as follows:

```
<@ lib="smaps" func="SMCPW" param="" @>
```

This is set up as a standard HTTP-Variable type of attribute, though you do not specify an attribute name. You must select "Active Expression" as the attribute type, then select the "Manual Entry" page in order to create this Active Response, since there is no variable name.

This attribute is only required on the POST rule for the SmCPW program. The GET rule does not need it.

Custom Forms

Custom forms can be written for password changes. They can be written in any language that can display an HTML form. The form must POST to SmCPW and may pass any or all command line arguments to it. If command line arguments are passed, the TARGET=<*targetURL*> should be passed, so that SmCPW knows where to send the user when the process is complete.

Prior to APS version 2.1, custom forms had to pass their query string on to SmCPW during the posting process. Thus, such custom forms had to be written in some server-side scripting language or as a CGI program. Since version 2.1, this is no longer required; custom forms can be written in "vanilla" HTML, if desired. If SmCPW receives a POST without any query string (the HTML form posts just to SmCPW), it will use its "caller's" (referrer's) query string.

SmCPW, when presented with an HTTP GET request, will generate an HTML form so that user may enter password information.

SmCPW need not be used to produce the form. You may provide your own HTML for the form, if special processing is desired.

Any form that you desire can be used, as long as it provides three text fields called "OldPassword", "NewPassword" and "VerifyPassword". The last two must contain identical values. Set up your form to post its data to SmCPW.EXE (or to whatever you have renamed that file).

An easy way to start is to run SmCPW.EXE through a browser and to select "View Source" from the browser's menu. This will display the raw HTML used by the form. You can save this HTML into a file and modify it as desired.

As an alternative, SmCPW can be run from the command line with no arguments. In this case, the default HTML can be captured to a file and used.

Customizing the Form

SmCPW also supports a considerable amount of customization on the default form. It uses the SmCPW.lang file to determine all of the text to be displayed on the form. In addition, some keys exist in the language file that are used to inject arbitrary HTML into the default form at various places in the output.

Forgot (Forgotten Password User Interface Stub)

Forgot (Forgot.EXE on Windows NT) is a CGI program that is accessed from a Web interface to drive the FPS process.

User Presentation Forms

FPS requires many forms to be presented to the user. As supplied by CA, FPS does not include these forms, though *sample* forms are provided in both ASP and JSP. It is up to the site to create the production forms (though CA's Professional Services can be contracted to produce these forms for you).

Furthermore, the architecture of FPS requires that these pages exist *on the same Web Server* as the FPS stub (FORGOT), since cookies are used extensively and they will only be visible to the pages served up from the same Web Server. All such forms must be installed on a web server, visible to the user base (inside or outside the DMZ) and must *not* be protected by SiteMinder (since unauthenticated users will be accessing these pages).

Note: Many sites use cascading style sheets (CSS) to standardize the look and feel of their forms. If cascading style sheets are to be shared with protected pages, they must be unprotected (or aliases or copies made) so that they will be accessible to unauthenticated users.

If a user is asked to authenticate during FPS operation, it is because either the target page itself or one of its components (graphic, frame or style sheet) is protected.

Note: FPS only works the user through the forgotten password process. It does not provide any mechanism for putting forgotten password questions into the user entry in the User Directory. It is up to the site to provide this mechanism. Delegated Management System (DMS) and CA Identity Manager are highly suited for this process. APS includes sample code to do this with DMS2.

APSAdmin (Help Desk Interface Stub)

APSAdmin (APSAdmin.EXE on Windows NT) is a CGI program that is accessed from a Web interface to provide help desk access to APS data in the users' entries.

APSAdmin uses the APSAdmin portion of the APS API. It does not access APS in any other way. All APSAdmin functionality can be duplicated by custom code, if desired.

For full details of all customizations possible and the configuration it requires, please refer to the APSAdmin utility (see page 295) chapter in this manual.

Policy Database Configuration

User Directory Credentials

SmAPS, the Policy Server component of APS, requires that all User Directories have administrator credentials stored within the policy server database. This is required because APS uses the connection to the User Directory maintained by SiteMinder (for *almost* all operations). APS needs to be able to write to the User Directory such information as the date of the user's last login and needs to be able to disable user accounts.

To enter User Directory credentials, edit the *properties* of the User Directory. Check the box marked "Require Credentials" and then enter the "Administrator" and "Password" values. This set of credentials should have sufficient access to the directory. After applying the changes, you should press the "View Contents" button to test the credentials. If nothing is displayed after pressing the button, then the credentials are incorrect.

For iPlanet/Netscape LDAP directories, CA *strongly* recommends the use of cn=Directory Manager as the administrator account. This account is hard-coded within iPlanet LDAP servers to bypass all Access Control Information (ACI). If any other account is used, it is possible that an ACI will prevent APS (or SiteMinder, for that matter) from reading or writing an attribute value.

There are implications to using the cn=Directory Administrator account under iPlanet that you should be aware of. Not only does it bypass all internal access controls, but it has unrestricted time and size limits on its server-side process and will be given priority by the Directory Server over all other user requests. APS' use of the Directory Server is highly optimized and is very resource-economic.

Change Password Page Configuration

Minimally, two Rules are required for changing passwords (one to GET the form, the other to POST the change) and *must* be protected in a separate Realm from all other pages on the site. This is easiest to create and maintain by creating a special "Change Password" Policy Domain.

This Realm *must not* have authentication and authorization events enabled. This prevents the possibility of infinite loops (where the user is redirected while being redirected to the change password page).

There should be two Rules defined within this realm, one to GET the form, the other to POST the change. Even if your site is using the default form produced by SmCPW (thus the GET and POST are the same resource) and though SiteMinder 3.6 (and later) allows multiple actions for a single resource, CA recommends that two separate Rules be created.

There should be a rule (within the Change Password Realm) for the form resource and an action of GET. If you are using the default form supplied with APS, this resource should be set to SmCPW*. If you are using your own form, be sure to append the wildcard to the resource definition so that various query strings can be passed to the page.

A second Rule for SmCPW* should be defined with an action of POST. Even if you are using a custom form, you will use SmCPW* for posting. Be sure to include the wildcard ("*") in the resource definition, so that the query string arguments can be passed.

A response attribute may be required to be passed when posting to SmCPW. This attribute explicitly identifies the user changing the password. With normal Web Agent configuration, this response is not needed (versions of APS prior to version 4 *required* this response). If SmCPW complains that it cannot identify the user, then this response will be required. If this response exists, SmCPW will use it, so it does not hurt to have it set up even when it is not needed.

See your SiteMinder documentation for how to set up Response Attributes. The active expression required is as follows:

```
<@ lib="smaps" func="SMCPW" param="" @>
```

This is set up as a standard HTTP-Variable type of attribute, though you do not specify an attribute name. You must select "Active Expression" as the attribute type, then select the "Manual Entry" page in order to create this Active Response, since there is no variable name.

This attribute is only required on the POST rule for the SmCPW program. The GET rule does not need it.

Finally, a single Policy must be created. Both Rules should be defined for the Policy and the Response should be assigned *only to the* POST Rule. All users in all directories should be allowed to access this policy.

Forgotten Password Configuration

No special SiteMinder Policy configuration is required for FPS. However, all pages involved with the FPS process, including both custom pages and the Forgot CGI program itself, must be entirely unprotected by SiteMinder. If cascading style sheets or client-side includes are used, they must be unprotected as well.

Help Desk Interface Configuration

The Help Desk Interface (APSAdmin) is designed to be a highly flexible, very secure *tool* that can be used by your Help Desk personnel to reset passwords and enable/disable user accounts. It also has some more generic user view/update capabilities.

APSAdmin is not intended to replace a full CRM system. It is designed so that it can augment an existing system or to provide limited such functionality.

There are three parts of APSAdmin configuration.

- Look & Feel customization. This is done with cascading style sheets and language files as described in the chapter entitled Help Desk Interface (APSAdmin) (see page 277).
- Content customization. This is configured using the APS Configuration file.
- Policy Server Configuration is discussed in this section.

Using the SiteMinder Policy Server User Interface (the Policy GUI), create a new *Policy Domain* called "APS Help Desk Interface".

Within the new Policy Domain, define a Realm named APSAdmin. This realm should be associated with the Agent or Agent Group corresponding to the Web Server(s) upon which this code was installed, *not* the APSAdmin agent defined in the SmPortal.cfg file. Be sure to use this agent/agent group for this realm. The Resource Filter is /APSAdmin/. The Authentication Scheme is whatever is appropriate for your site.

Define a Rule within this Realm called Help Desk Interface. The Resource will be APSAdmin*. The Action is GET *and* POST.

Define a Response called Administrator Credentials. This response needs a single Attribute. This attribute needs to have a type of "WebAgent-HTTP-Header-Variable". Select "Static" as the Attribute Kind. The Variable Name field should be set to "APSAdmin". The Variable Value must contain a SiteMinder Administrator name, followed by a semicolon, followed by *that* administrator's password. Note that this is a *SiteMinder Policy Server User Interface* administrator (the credentials used to log into the SiteMinder Policy Server GUI, not into the Web Site).

- If you do not want this information in your Policy store in clear text, you may encrypt it using APSEncrypt. The entire value, both the Administrator name and password (separated by a semicolon) must be encrypted together.
- This administrator must be defined to SiteMinder with "Manage Users" and "Manage System and Domain Objects" rights.

Create a Policy called Help Desk Administration. Select those users that should have access to this interface. The "Help Desk" rule defined above should be specified. The "Administrator Credentials" response should be tied to the rule.

Event Handling

The following configuration is required for the redirections supported by APS. These should be set up regardless of which events are set up for APS. Later, if an event is enabled in the configuration file, everything will work properly if this setup was performed initially. None of this setup has any affect on email notification of events.

Realms

Every single Realm defined in the Policy Database must have Authentication and Authorization events enabled, *except* the Change Password realm.

Rules

There must be three Rules defined in every Realm, *except* the Change Password realm.

An OnAuthAccept rule to catch password expiration, password change warnings and other events that occur, even though the user properly authenticates.

An OnAuthReject rule to catch "three strikes you're out" and other events that occur when SiteMinder accepts, but APS rejects, the user.

An OnAccessAccept rule to process forced password change requests.

Rule Groups

Three Rule Groups should be defined in each **Policy Domain** *except* the Change Password domain.

Each Rule Group should collect all of the rules within the Policy Domain that are alike. That is, all of the OnAuthAccept rules should be collected together into a single Rule Group, all of the OnAuthReject rules together into a single Rule Group and all of the OnAccessAccept rules into their own Rule Group.

Responses

Three Responses must be created in every Policy Domain *except* the Change Password domain.

The first response, for the OnAuthAccept events, should contain a single Active Attribute. This attribute *must* be of the type OnAccept-Redirect and invoke the following active attribute:

```
<@ lib="smaps" func="SmApsRedirect" param="" @>
```

The second response, for the OnAuthReject events, should contain a single Active Attribute. This attribute *must* be of the type OnReject-Redirect and invoke the following active attribute:

```
<@ lib="smaps" func="SmApsRedirect" param="" @>
```

The third response, for the OnAccessAccept events, should contain a single Active Attribute. This attribute *must* be of the type OnAccept-Redirect and invoke the following active attribute:

```
<@ lib="smaps" func="AZRedirect" param="" @>
```

For all three responses, you can access additional functionality by providing values for the param argument. See the section entitled *Redirection* for a complete discussion of these options.

Policies

Each Policy Domain, *except* the Change Password domain, will need one additional Policy (in addition to its own). This policy will bind each of the three Rule Groups to the three responses. The Policy should apply to all users.

Authentication Scheme

You will need to install and configure the APS Authentication Scheme if you expect to use:

- Password case insensitivity or case forcing.
- The FPS One Shot Password capability.

A complete description of the Authentication Scheme and how to configure it is can be found in the chapter entitled Authentication Scheme (see page 341).

Chapter 3: APS Configuration File (APS.CFG)

This section contains the following topics:

Overview (see page 61)
Format (see page 62)
Define Macros (see page 63)
Setting Overrides (see page 65)
File Sections (see page 70)
Settings That Can Appear Anywhere (see page 71)
General Settings (see page 72)
Password Content Settings (see page 80)
Password Reuse (see page 93)
Change Required (see page 94)
Restricting Words from the User's Profile (see page 95)
Run-time Password Checking (see page 98)
Event Redirection (see page 113)
Generational & Other Automatic Redirection (see page 118)
Sending Mail (see page 121)
Mail Settings (see page 125)
Custom Logging (see page 128)
Custom Extensions (see page 128)
Invalid Password Dictionary (see page 129)
Password Complexity (see page 130)
Field Mappings (see page 133)
ODBC Queries (see page 136)
APSExpire (see page 150)
General FPS Settings (see page 151)
FPS LDAP Settings (see page 154)
FPS Identify Process (see page 154)
FPS Change Password Process (see page 175)
FPS Confirm Process (see page 176)
FPS Errors (see page 186)
Handling FPS Mail Errors (see page 187)

Overview

APS is configured using a standard text editor. Configuration settings are stored in a file called APS.cfg. Under Windows, this file must be located in the same directory as the SmAPS library. Under Solaris, this file is pointed to by the APS_SETTINGS environment variable. This environment variable points to the file itself, not just the directory containing the file.

Under Solaris, if the APS_SETTINGS environment variable is not set, APS will look for the file at $SMHOME/siteminder/bin/APS.cfg.

The configuration file controls all of the behavior of APS and the required formation of passwords during the password change function.

If the file does not exist, an error is logged and default settings will be used. If any parsing errors occur, errors will be written to a file with the same name as the APS.cfg file, with the date and .LOG appended to the file name. This file will be placed in the same location as the APS.cfg file. If the process does not have the rights to create this file, no file will be created.

The APSTestSettings utility will display the settings from this file that apply to a specific user (or, if no argument is supplied to APSTestSettings, the general configuration settings). If the file is changed, APSTestSettings can be used to test the new file, since the file will be parsed and any errors encountered will be logged.

Note: The decision to use a flat file for configuration was a deliberate one. In the years of working in these environments, we have discovered that password policies are essentially static, they very rarely change and when they do, they are changed in a development environment, fully verified, then rolled out into production. By using flat files, this rollout and all of the associated change control becomes trivial to implement.

Format

Within the file, all blank lines are ignored and any line beginning with two forward slashes ("//"), a pound sign ("#") or a semicolon (";") is considered a comment. Embedded and trailing comments are not supported.

For the most part, each operational line consists of a keyword (or words) followed by an equal sign, followed by zero or more spaces, followed by the setting for that keyword. This chapter contains a detailed listing of all of the keywords and their expected arguments. In addition, the sample APS.CFG file supplied contains a considerable amount of discussion for each keyword. If discrepancies exist between this document and comments in the file, use the comments in the file.

Keywords are not case-sensitive. The settings for the keywords usually *are* case-sensitive.

By default, almost all options are disabled in the sample file. Options should be enabled as needed. Options should be set with care, as it is possible to create situations where no password exists that can satisfy all conditions. Even if combinations are possible, you should be careful: complex rules can confuse users.

If the file is modified while APS is running, the changes will apply to the next user accessing the system or changing their password. There is no need to restart SiteMinder services. APS checks the last modification time of the file each time that a user is processed. If the file has changed, it is read in by APS and used for the new user request and for all future requests, until the file is changed.

There is slight performance degradation due to the large number of comments in the configuration file. However, because of file caching, it is only noticeable during startup and when the file actually changes. Thus, we highly recommend that the comments be left intact.

The configuration is intentionally stored in a flat file rather than in some distributed data store. This gives us a much higher degree of configuration flexibility and portability. For example, if the Primary SiteMinder server always accesses a Primary LDAP server, there may be no need for Write Back settings, whereas a Backup SiteMinder Server (that might be accessing a backup LDAP Server) might need to use the Write Backs.

If multiple SiteMinder Policy Servers are used, a copy of this file must exist on all of them. If the same configuration is to be used on all, the site can use one of the many file replication utilities available on the market. Thus, changing the file on one server would cause that change to be replicated to all other servers. CA does not supply, support, or recommend any particular utility to perform this function.

Define Macros

Most settings in the configuration file are site-dependent, not machine- or environment- (DEV, QA, STAGE, PRODUCTION) specific. However, there are a few things in the file that may vary from Policy Server to Policy Server or between environments.

New at Version 4, APS supports define macros. These are simply replacement values that can be defined once, anywhere in the file, then used as many times as necessary, by referencing a name, rather than a value.

To define such a macro, for example, use the following syntax:

```
define X Y
```

After this definition, anywhere that <X> appears will be replaced with Y. For example:

```
define LDAP_SERVER 127.0.0.1
LDAP Writeback Server=<LDAP_SERVER>
```

In this case, the actual value used for the LDAP write back Server setting will be 127.0.0.1

This is incredibly useful for separating environment-specific settings, reducing the errors which occur when an APS.cfg file is copied from one environment to another (such as DEV to QA to PROD). For example:

```
define CPW_SERVER myserver.security.com
Failure Redirect=http://<CPW_SERVER>/Failure.htm
```

Macro names are not case sensitive. If a macro is referenced, but not defined, APS will look into the process environment variables for a match on the macro name. If one is found, its value will be used as the value of the macro. If no matching environment variable is found, the reference is left intact.

Macros are processed during file load. They are not evaluated per user, so there is little performance impact.

Macros take effect only from the point where they are defined forward into the file.

If a macro has the same name (case-insensitive) as a previously defined macro, the new value replaces the old one from that point down.

Define macros can also be used with user classes and overrides (explained in the next section) to make definitions environment independent:

```
define LDAP_SUFFIX o=dev.Airius.com
@Admin=IsInGroup("cn=Admin,<LDAP_SUFFIX>")
```

When the file is promoted into QA, only the DEFINE line needs to change.

Comment checking is performed *after* macro expansion, so the following method can be used to make conditional (not overridden) settings:

```
define NO_MAIL ##
<NO_MAIL>Server=127.0.0.1
```

When NO_MAIL is set to a comment character, the setting is turned off. If NO_MAIL has no value (blank), then the mail server setting is valid. A very useful environment-specific setting (don't send email in DEV). Note that if NO MAIL is undefined (does not appear as a macro), an error would result.

External Macro File

Define macros need not be stated in the APS Configuration File. They can also be defined in a separate file called APS-Macros.cfg located on the same directory as the APS.cfg file.

However, the external macro file will not automatically be loaded. It will only be loaded when the Use External Macros keyword is used in the APS.cfg file. Note that the external macro file can be loaded more than once (which might be useful if you want to change a value temporarily). It requires explicit loading in order to maintain backward compatibility.

For example, the APS.cfg file might contain:The external macro file can be extremely useful. It allows a site to define machine-specific settings externally to the APS.CFG file. Thus, as an APS.cfg file is copied between Policy Servers, either from environment to environment (DEV to QA, for example) or within an environment, settings can "adapt" to the proper machine.

```
@QA_Users=<QA_DEF>
```

In the DEV and QA environments, the APS-Macros.cfg file might contain:

```
define QA_DEF IsInGroup("cn=QA,o=Airius.com")
```

Whereas, in the production environment, the APS-Macros.cfg file might contain:

```
define QA_DEF FALSE
```

Another example, used with multiple Policy Servers within the same environment, might be (in the APS.cfg file):

```
LDAP Writeback Server={<USE_WRITEBACK>}127.0.0.1
```

On a policy server that directly accesses the master, the APS-Macros.cfg file would contain:

```
define USE_WRITEBACK TRUE
```

On a policy server that does not access the master by default, the APS-Macros.cfg file might contain:

```
define USE_WRITEBACK FALSE
```

Note that macros can be defined anywhere in this file, but it is most useful (and avoids confusion) if they are defined at the top of the file.

Setting Overrides

Almost all settings can be overridden, meaning that an expression can be supplied with the setting that describes the users to which the setting applies.

Overrides are always defined within curly braces ("{" and "}") and appear immediately after the equal sign for a setting. For example, to provide an override for the Minimum Length setting, use the following:

```
Minimum Length={<override expression>}5
```

If the current user satisfies the specified override definition, the setting will be considered for use.

When specifying an override, note that it may not always be used, even if it applies. APS will examine all values for a given setting, using all that apply for the current user. If multiple values apply, including a default value, then the most restrictive value will be used. For example, suppose the following two settings were specified:

```
Minimum Length=5
Minimum Length={givenname="Eric"} 4
```

Users with a first name of Eric would have a minimum length of 5, not 4, since both settings apply and 5 is more restrictive than 4.

When multiple string settings might apply, usually the last applicable setting will be used (sometimes something other than the last will be used, as noted in the text describing the specific setting).

Override expressions are logical (boolean) expressions. Expressions can compare a user's attribute value with a constant, test certain functions, test a context macro against a constant or some combination.

Individual expressions may be connected using logical operators. The following logical operators are supported:

NOT or !	Logical NOT
AND or & or &&	Logical AND
OR or | or ||	Logical OR
XOR or ^	Logical Exclusive OR
TRUE	Constant TRUE
FALSE	Constant FALSE

Logical operators are optimized to *short-circuit* (not evaluate a second operand if the result of the operator is already decided).

The following comparison operators are supported. Note that there are both case-sensitive and case-insensitive versions of each.

=	Case sensitive equality compare
~=	Case insensitive equality compare
>	Case sensitive greater than compare
~>	Case insensitive greater than compare
<	Case sensitive less than compare
~<	Case insensitive less than compare
>=	Case sensitive greater than or equal compare
~>=	Case insensitive greater than or equal compare
<=	Case sensitive less than or equal compare
~<=	Case insensitive less than or equal compare
STARTS_WITH	First operand must start with second (case sensitive)
~STARTS_WITH	First operand must start with second (case insensitive)
ENDS_WITH	First operand must end with second (case sensitive)
~ENDS_WITH	First operand must end with second (case insensitive)
CONTAINS	First operand must contain second (case sensitive)
~CONTAINS	First operand must contain second (case insensitive)

All comparison operators require an (unquoted) attribute name as the first operand and a (quoted) constant for a second operand. Furthermore, the first operand (the attribute) can be qualified by SOME or ALL. SOME indicates that if at least one value of the (assumed multivalued) attribute compares true, the result is true. If ALL is specified, then all values of the (multivalued) attribute must compare true in order for the condition to be true. SOME is the default. For example:

```
givenname~="Eric"
```

If givenname is a multivalued attribute, then if any (because SOME is the default) value of the givenname attribute is Eric (case-insensitive), the result is TRUE. This is equivalent to:

```
SOME:givenname ~= "Eric"
```

On the other hand,

```
ALL:givenname ~= "Eric"
```

would require that all of the values of givenname match Eric (case-insensitive). While this is not a viable example, the following might be:

```
ALL:givenname ~STARTS_WITH "E"
```

This, of course, could be further qualified as:

```
NOT ALL:givenname ~STARTS_WITH "E"
```

Note that new users have no attribute values at evaluation time, so these compares are essentially useless in those cases. This typically applies to the self-registration process (additionally, APS cannot check the password against user attributes, since the user record does not yet exist). This problem is typically solved by assigning the new user a random password during the registration process and then forcing the user to change their password at first login (when the user is no longer a new user - the user record exists in the directory).

Some "built-in functions" are also supported. Almost all functions take a single constant, quoted argument. Most function calls will work with new users (IsInGroup is an exception, new users cannot be in a group). All functions return TRUE or FALSE.

```
At("<some DN>") or In("<some DN>")
Above("<some DN>") or Over("<some DN>")
Below("<some DN>") or Under("<some DN>")
IsInGroup("[set the product group or family]")
IsNull("<attribute Name>")
IsInNamespace("<namespace>")
IsInDirectory("<directory id>")
IsNew()
IsLDAP()
IsODBC()
IsInWindows() or IsWin() or IsWinNT()
Self()
AnyBitsSet(<attribute>,<bitmask>)
AllBitsSet(<attribute>,<bitmask>)
```

The Self function can only be used in the APSAdmin section of the APS.cfg file.

Thus, the following might be useful:

```
IsInGroup("cn=Administrators,o=Airius.com")
```

Parentheses can be used to change the order of evaluation. For example:

```
(IsLDAP() AND IsInGroup("cn=Admin,o=Airius.com"))
OR (IsODBC() AND IsInGroup("Admin"))
```

Since the evaluator short circuits, ODBC users will never evaluate the LDAP IsInGroup call.

Nothing outside of quotes is case sensitive (attribute names, function names, textual operators, etc.)

User Classes

The evaluator supports *user classes* as well. A user class can be defined *anywhere* in the configuration file, using the format:

```
@classname=<expression>
```

Such as:

```
@Administrator=IsInGroup("cn=Admin,o=Airius.com")
```

User classes can be referenced anywhere in an override, such as:

```
Minimum Length={@Administrator} 12
Minimum Length={@Administrator AND givenname="Eric"} 14
```

User class names are *not* case-sensitive. They are very useful for defining classes of users just once in the file, then referencing the override in many places.

User classes must be defined before they are used. Duplicate definitions are not allowed.

The class definition (like all override expressions) is parsed at load time, but not evaluated until run time.

Context Macros

Context Macros (not to be confused with define macros) can be passed from an Active Expression (in its param field) or exist only within the context of a specific setting. They are typically used to pass REALM or APPLICATION information at run-time.

Context macros can appear anywhere that an attribute name can appear.

Context macro names are prefixed with a percent sign ("%").

For example:

```
    Immediate Change Redirect={@Administrator AND %App~="Main"} SmCPW.exe
```

A note about "Old Style" overrides

Prior versions of APS (before version 4) supported a different override notation (using square brackets) that only applied to the user's location in an LDAP DIT. This notation is no longer supported and, if used, may cause APS to operate in unexpected ways. Such syntax should be manually converted as soon as possible.

Performance

The evaluator has been highly optimized and is very fast. All results, both intermediate and final, are cached for each user, so that identical clauses will not be re-evaluated. In other words, once a given comparison is made, it will not be made again (regardless of the use of classes), the cached result will be used instead.

While user classes might appear to improve performance, they have very little impact (either way). However, they improve readability of this file and will reduce data entry error, since overrides can be defined for classes, rather than repeating a complex definition.

Testing Expressions

APSTestSettings is a utility provided with APS for testing both the contents of this file (as described above), for testing the lexer (the part of code that breaks up an expression) and the parser (the part of code that evaluates an expression). It can run in 3 modes. See page 270 for the complete description of APSTestSettings and its operation.

File Sections

The APS Configuration File is divided into sections. Each section starts with a single line identifying the name of the section, surrounded by square brackets.

The sections supported in the configuration file are:

[APSAdmin]	Defines behavior of the APSAdmin utility and API
[APSExpire]	Defines job parameters for the APSExpire utility
[Complexity]	Used for tuning password complexity parameters
[Custom]	Settings for the custom extension module SmAPSEx
[Dictionary]	Lists words that are disallowed in passwords
[FPS]	General Forgotten Password Support settings
[FPS-CHANGE]	Defines how FPS allows changes to the user password
[FPS-CONFIRM]	Defines how FPS confirms password changes
[FPS-ERRORS]	Defines how FPS handles errors
[FPS-IDENTIFY]	Defines how FPS identifies users
[FPS-NOMAIL]	Defines how FPS handles mail errors
[FPS-VERIFY]	Defines how FPS verifies user identity
[Logging]	Settings for the custom logging module SmAPSLog
[Mail]	Defines how APS communicates with a mail server
[Mappings]	Maps APS field names to attribute/column names in the underlying directory
[ODBC]	Specifies the SQL queries used to access an ODBC User Directory

There are General settings that must appear *before* any section in the APS.cfg file. These settings are the most commonly changed and actually control the core functionality of APS.

There are a few (very few) settings that can appear anywhere in the configuration file. All of these were discussed previously in text, but will be presented here for reference as well.

Settings That Can Appear Anywhere

These settings can appear anywhere in the APS configuration file. They affect the processing of the configuration file, not really the behavior of APS itself.

These settings can appear anywhere because they are positional, in that they take effect from the point at which they are defined to the end of the file. Entries defined "above" them in the configuration file are not affected.

Define

Range: n/a

Default: none

Recommended: Yes

Complexity Level: Intermediate

This keyword is used to create define macros as described on page 41.

Define macros are used to create symbolic names for textual strings that can be used, by name, in later parts of the APS Configuration File. The symbol will only be replaced when it appears surrounded by angle brackets ("<" and ">").

This keyword can appear anywhere in the file. Text replacement occurs only after the point of definition. If the same keyword is defined again, its new definition will overwrite any prior definition.

```
define FPS_SERVERhttp://www.myserver.com/
[FPS-IDENTIFY]
URL=<FPS_SERVER>Identify.jsp
```

Use External Macros

Range: n/a

Default: no

Recommended: Yes

Complexity Level: Intermediate

Define macros need not be defined in the APS Configuration File. They can also be defined in a separate file called APS-Macros.cfg located on the same directory as the APS.CFG file.

The use of this keyword is described starting on page 42.

This keyword can appear more than once in the file. If so, the external definitions will be reloaded.

@<classname>

Range: n/a

Default: none

Recommended: Yes

Complexity Level: Intermediate

This "keyword" allows the site to define User Classes.

A site may define as many user classes as it requires. User classes may reference other user classes, as long as the referenced classes are already defined (thus you cannot have circular references).

General Settings

Trace

Range: n/a

Default: off

Recommended: Yes

Complexity Level: Basic

If this keyword appears, then tracing is turned on within APS. Traces appear in the Policy Server console log. To turn tracing off, comment out or delete this keyword from the file. This affects tracing of authentication and authorization time processing only. To enable tracing for FPS and SmCPW, use the Trace keyword in the SmPortal.cfg file.

This keyword cannot be overridden per user.

Example:

Trace

Timeout

Range: 0-300

Default: 0

Recommended: No

Complexity Level: Advanced

This setting controls how long to wait for a User Directory to respond to a call. This value is for each call, not overall time. If this time limit is exceeded for a single call, a timeout error will result.

The default is zero (no limit or limit enforced by driver). In general, this setting should be used only with great care, since it may cause timeout failures.

This setting is specified in seconds. The maximum is 300 (5 minutes), but should never be used, since a browser session will time out long before the interaction with the directory server.

If an override is required, it must be a restricted override. That is, it can reference the Directory, but not the user.

This setting is supported for ODBC and LDAP directories only. It **does** affect LDAP Writeback connections.

Example:

Timeout=30

Poll

Range: 10-1000

Default: 100

Recommended: No

Complexity Level: Advanced

This setting controls how long to sleep between checks to see if an ODBC User Directory call has completed. Lower numbers indicate that the driver should be checked more frequently. This may improve response time for the current user, but will increase overall CPU utilization. Higher numbers will decrease user responsiveness, but will use less CPU.

This setting is specified in thousandths of a second. The minimum value is 10 milliseconds (1/100 second) and the maximum is 1000 milliseconds (1 second). Default is 100 milliseconds (1/10 second).

If an override is required, it must be a restricted override. That is, it can reference the Directory, but not the user.

This setting is supported only for ODBC directories. LDAP directories support the Timeout keyword only.

Example:

Poll=500

LDAP Write Back Information

The write back settings indicate where APS should write back information when writing to the LDAP directory. This is used primarily for when SiteMinder is reading from a replicant (consumer) and APS must write back to a master (producer). Instead of attempting the write to the (by definition: read-only) replicant, which will **always** fail, APS will always write to the Write Back server instead (it will still read from the replicant).

If you know that you are going to be doing referrals on writebacks (if SiteMinder reads from a replicant), use these settings.

Whenever APS needs to write information to an LDAP Directory, it looks for a writeback setting for the current user. If one exists, APS will perform the write to the writeback server instead of to the original server. When the LDAP Directory is configured as a Producer/Consumer (rather than Primary/Backup), this is much more efficient than writing back to the consumer, failing, then referring the write to the producer.

All write backsettings can be overridden. It is not necessary to use the IsLDAP() function with these overrides, since they will only ever be used for LDAP User Directories.

Be careful when using overrides with these settings, since they each override separately. Make sure that there are no "overlaps" in the overrides, where one setting is overridden separately from the others.

When SiteMinder supports referrals in all cases, these keywords will technically no longer be needed; however, they may still be desirable, since performance may be greatly enhanced (since APS will apply changes directly to the master, rather than "bouncing" all writes off of the replicant).

A list of writable servers can be provided, separated with a space. APS will treat the list as a list of LDAP servers for fail over purposes. All must be writable. In this way, APS supports multi-mastering.

It may be desirabled to load-balance APS writes across multiple masters. APS will not do this automatically, but sites may do it in either of two ways. Note that both methods provide the same level of fault tolerance.

- If multiple SiteMinder Policy Servers are running in round-robin mode, vary order of the list of servers specified as Writeback servers on each Policy Server. Each Policy Server will then use the failover list in a different sequence, obtaining somewhat random round-robining. This is easy to implement and requires little tuning.
- For better LDAP performance, you can specify multiple writeback settings in each APS.cfg file, using overrides to partition your data. You might, for example, send users whose UID starts with A-M to servers "A", then "B" and all other users to "B", then "A". You might be able to get better performance out of the LDAP servers (due to improved cache utilization), but you may need to do a considerable amount of tuning to make any significant impact.

LDAP Writeback Server

Range: n/a

Default: none

Recommended: no

Complexity Level: Advanced

This setting controls the server to which writebacks will be directed. It can be overridden. To use writebacks, both the Server and a Password must be specified (the DN is not required because it has a default).

```
LDAP Writeback Server=127.0.0.1
LDAP Writeback Server={IsAt("o=security.com"} 10.24.86.177
```

LDAP Writeback DN

Range: n/a

Default: cn=Directory Manager

Recommended: cn=Directory Manager

Complexity Level: Advanced

This is the Distinguished Name (DN) of the administrator account to use to log into the writeback server. It is not required to use writebacks, since there is a default.

```
LDAP Writeback DN=cn=Directory Manager
LDAP Writeback DN={IsAt("o=security.com"}uid=Admin, o= security.com
```

LDAP Writeback Password

Range: n/a

Default: none

Recommended: no

Complexity Level: Advanced

This is the password to use for binding to the writeback server. If both this setting and a server are supplied, it indicates that writebacks should be used, even if the Writeback DN (above) is not supplied.

The writeback Password can be supplied either in clear text or encrypted. To encrypt the password, use the APSEncrypt utility.

```
LDAP Writeback Password=[NDSEnc-A]68SExOCvk(MrHfSSY) cuDHfuyG2
LDAP Writeback Password={IsAt("o=security.com"}password
```

LDAP Writeback Use SSL

Range: n/a

Default: no

Recommended: if needed

Complexity Level: Advanced

If writebacks are used, this setting tells APS that LDAPS (LDAP over SSL) should be used for the writeback connection.

```
LDAP Writeback Use SSL=true
```

Other LDAP Settings

LDAP Disabled Groups

Range: n/a

Default: Search Base defined in SiteMinder User Directory

Recommended: no

Complexity Level: Advanced

By default, APS will search for and create (if necessary) disabled groups in the root of the user account's directory. If APS will not have rights to this location, the root is not identified by o=xxx, or your site wishes these groups to appear elsewhere in the directory tree, you can specify their exact location using this keyword. Note that this is the location, not the names of the groups themselves.

This affects only where such groups are created, not where they are searched for. APS will always search the entire search base as defined in the SiteMinder User Directory entry.

This setting can be overridden, but the IsLDAP function is not required, since the setting will only be used for LDAP User Directories.

APS does not create groups for ODBC directories, so this setting does not apply to them.

```
LDAP Disabled Groups=ou=Groups,o=nds.com
```

LDAP Blob

Range: n/a

Default: audio

Recommended: audio

Complexity Level: Advanced (upgraded sites only)

Prior to APS Version 3.0, APS maintained user information for LDAP users in an attribute called the APS Blob. This keyword allowed the site to specify the attribute in which to store this information. However, at Version 3.0, the APS blob no longer exists.

This keyword exists only for backwards compatibility. Users converting from pre-version-3.0 APS should contact CA for the proper data conversion tool.

If this site did not upgrade from a prior version of APS or it is no longer maintaining old Blob information, then this keyword is not needed and should not be specified.

Check LDAP Password Existence

Range: n/a

Default: off

Recommended: off

Complexity Level: Advanced

This keyword will cause APS to check for the existence of an LDAP password before expiring it or requiring it to be changed. It should be used at sites where some users will be accessing the system using authentication schemes that do not use passwords (and thus the user record may or may not actually have a password).

Note that this was a new feature at version 2.0 and requires that the Directory Manager credentials be entered into the SiteMinder policy GUI for each LDAP directory (it only impacts LDAP User Directories). It may also impact performance.

This functionality requires that the existing password be readable from the directory (at least the encrypted/hashed value). For iPlanet Directories, this requires that the administrator be cn=Directory Manager. Other implementations of LDAP servers may have different requirements. Some implemented, most notably Microsoft Active Directory, do not allow this information to be read in any case and thus this functionality is not supported in those environments.

```
Check LDAP Password Existence
```

LDAP Password Attribute

Range: n/a

Default: userPassword

Recommended: no

Complexity Level: Advanced (Deprecated)

This keyword causes APS to use an attribute other than userPassword as the password attribute within an LDAP directory. The standard LDAP schemas use userPassword; however, some may use a different attribute. Use this keyword to specify a different attribute.

This setting may be overridden (but that would be unusual).

At APS Version 4.0, this setting is deprecated (a warning will be output to the log, if the keyword is encountered). If something other than userPassword should be used, specify it in the new Mappings section.

```
LDAP Password Attribute=userPassword
```

Use Internal Disables

Range: n/a

Default: off

Recommended: true

Complexity Level: Advanced

If this setting is enabled, when APS disables an LDAP user account (it only applies to LDAP User Directories), it will NOT put the user account into a Disabled group. Instead, it will set the Disable Until date to FOREVER, effectively disabling the user account forever.

This feature was new at APS Version 3.0. It provides a considerable performance improvement for very large sites, in that large groups are not used for disabling user accounts. However, it is a change to existing functionality and may impact user administration tools (the tools used to reset the user).

In addition, if this setting is enabled, a user account can only be disabled for a single reason at a time. This may impact how some of the security policies at your site are to be architected.

ODBC User Directories always use internal disables (Disabled groups are *honored*, but APS will never create one or put a user account into one).

```
Use Internal Disables
```

Ignore

Range: n/a

Default: off

Recommended: no

Complexity Level: Advanced

APS functionality is accessed in many ways. For the two secured client mechanisms (Change Password Interface and Help Desk Interface), access is controlled by SiteMinder Policies. FPS is configured separately to allow or disallow access for particular users. The APSAPI is accessed by the applications that call it and thus access can be controlled using any of a variety of methods.

However, APS is involved with all SiteMinder authentications. SiteMinder calls APS, regardless of User Directory or the success or failure of an authentication.

The **Ignore** keyword can be used to tell APS when to ignore a user during the authentication process. While this is rarely used, it does have a purpose. For example, when rolling out a new User Directory, it can be desirable to ignore all of the users in that new directory until the schema is set up and the users are initialized.

The value of this setting is either true or false. However, this setting is almost always overridden, defining the user cases where the user should be ignored.

```
Ignore={IsODBC()} true
```

Disable

Range: n/a

Default: off

Recommended: no

Complexity Level: Advanced

APS has a wealth of mechanisms for enabling/disabling user accounts, but they are mostly geared for APS' own use. Sometimes (especially in legacy directories), other mechanisms are used to disable accounts.

This setting allows sites to specify other conditions under which APS is to consider that an account is disabled. The override of the setting specifies the condition; the value of the setting is used as the "Disabled Reason" passing such codes to redirects or email.

APS will never actually disable an account with this mechanism. However, if the override is true for a particular account, that user will not be allowed to log into the system.

```
Disable={employeeStatus="Terminated"} Term
Disable={customerStatus="IN LEGAL"} Litigation
```

Password Content Settings

These settings allow the administrator to control what a new password must look like. For example, a policy could require passwords to contain all numbers, at least three letters, or at least two numbers and at least four punctuation characters.

Use caution with these settings, as it is possible to set requirements that no passwords could ever satisfy. For instance, should the Maximum Password Length field be set to 8 characters, the Letters field set to 6 and the Digits field set to 6, all passwords would implicitly be required to contain at least 12 characters (6 letters and 6 digits), but no passwords this length are allowed. Thus, all passwords would be rejected.

Because of setting overrides, APS cannot detect such impossible combinations when the configuration file is read. When a password is validated, however, APS will determine if the combination of settings that apply to that user is possible. If it is not, an error is displayed to the user. While APS can detect many such combinations (at run-time), it may not be able to detect all of them.

All settings can be overridden.

These settings are used only when the user changes their password, they may not impact an Administrator changing user passwords except as provided by APSAdmin (see page 210) or if the APS API is called.

Passwords can be tested using the IsMyPasswordValid and IsPasswordValid functions in the APS Application Programming Interface (API).

Password Length

Minimum Length

Range: 4-32 characters

Default: 4

Recommended: 6-8

Complexity Level: Basic

Controls the minimum length allowed for passwords. Any setting below four will be ignored. The minimum may not exceed the maximum. Attempting to set this value outside of the allowed range will cause parsing error and the setting will be ignored.

```
Minimum Length=4
Minimum Length={@Employees}6
```

Maximum Length

Range: 4-32 characters

Default: 32

Recommended: 32

Complexity Level: Basic

Controls the maximum required length of a password. Any setting above 32 will be ignored and a parsing error will be logged. The Minimum Length must be less than or equal to the maximum length.

```
Maximum Length=20
```

Required Content

Minimum Letters

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Intermediate

This setting requires that the password contain a certain minimum number of alphabetic letters. Alphabetic characters are defined as the letters in the alphabet, regardless of case.

```
Minimum Letters=2
```

Minimum Uppercase

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting requires that the password contain a certain minimum number of upper case alphabetic letters. Care should be taken in non-standard character set environments, since APS detection of upper case letters only applies to standard ASCII letters.

```
Minimum Uppercase=2
```

Minimum Lowercase

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting requires that the password contain a certain minimum number of lower case alphabetic letters. Care should be taken in non-standard character set environments, since APS detects lower case only using the standard ASCII character set.

```
Minimum Lowercase=2
```

Minimum Digits

Range: 0-32 characters

Default: 0

Recommended: 2

Complexity Level: Intermediate

This setting requires that the password contain a minimum number of numeric digits ("0" to "9").

```
Minimum Digits=1
Minimum Digits={@Customers} 2
```

Minimum Alphanumeric

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Intermediate

This setting requires that the password contain a certain minimum number of alphanumeric characters ("A"-"Z" or "0"-"9"). If this setting is used along with one of the settings above, characters can satisfy both requirements. For example, if Digits is four and this setting is four, the password 1234 satisfies both requirements.

```
Minimum Alphanumeric=1
Minimum Alphanumeric={@Employees} 3
```

Minimum Punctuation

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Intermediate

This setting requires that the password contain a certain minimum number of punctuation marks. These can be periods, commas, exclamation marks, etc.

```
Minimum Punctuation=1
```

Minimum Symbols

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Intermediate

This setting requires that the password contain a certain minimum number of symbol characters. Symbol characters are defined within APS as the characters:

"~" (tilde)	"@" (at)	"#" (number)
"$" (dollar)	"%" (percent)	"^" (circumflex)
"&" (ampersand)	"*" (asterisk)	"(" (open parenthesis)
")"(close parenthesis)	"_" (underscore)	"-" (hyphen)
"+" (plus)	"=" (equals)	"{" (open brace)
"}" (close brace)	"[" (open bracket)	"]" (close bracket)
"<" (less than)	">" (greater than)	"/" (virgule)
"\" (back slash)	"|" (vertical bar)	

and all extended ASCII characters, including diacritical marks. Note that some browsers do not support the entry of extended ASCII characters and some LDAP directories do not support them as part of an LDAP attribute.

```
Minimum Symbols=1
Minimum Symbols={@Partners} 2
```

Minimum Other

Range: 0-32 characters

Default: 0

Recommended: 1-2

Complexity Level: Intermediate

This setting requires that the password contain a specified minimum number of non-alphanumeric characters. This includes punctuation marks and other symbols located on the keyboard.

Similar to the Minimum Alphanumeric field above, a given character can satisfy this requirement in addition to "Punctuation" or "Symbols".

```
Minimum Other=1
```

Combinations of Character Types

Using the above settings, a site can require certain combinations. However, the use of those settings is fairly inflexible, in that a new password must satisfy all of those settings.

APS can allow a site to specify a series of settings, then require that a new password satisfy a minimum number of them.

The easiest way to think of this is as a "point system". The site specifies the conditions required to get each "point", then the minimum number of "points" that a new password must have.

The Combination Xxxx settings (below) specify the conditions required to earn a "point". The Minimum Combinations setting tells APS how many points are required by new passwords.

For example, if a site wants to require an uppercase letter, a lowercase letter, a digit, and a punctuation mark, they would use the following settings:

```
Minimum Uppercase=1
Minimum Lowercase=1
Minimum Digits=1
Minimum Punctuation=1
```

However, if they only wanted to require any three of the four (instead of requiring all four), a site would use:

```
Combination Uppercase=1
Combination Lowercase=1
Combination Digits=1
Combination Punctuation=1
Minimum Combinations=3
```

APS will log a warning to the SiteMinder console log if:

- One or more Combination Xxxx Settings is in effect, but there is no Minimum Combinations setting.
- There are not enough Combination Xxxx settings defined to satisfy the Minimum Combinations in effect.

Combination Letters

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting defines the number of letters (upper- and lower-case) needed to satisfy one of the possible Minimum Combinations of character types required in the password.

```
Combination Letters=1
```

Combination Uppercase

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting defines the number of uppercase letters needed to satisfy one of the possible Minimum Combinations of character types required in the password. Care should be taken in non-standard character set environments, since APS detection of upper case letters only applies to standard ASCII letters.

```
Combination Uppercase=1
```

Combination Lowercase

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting defines the number of lowercase letters needed to satisfy one of the possible Minimum Combinations of character types required in the password. Care should be taken in non-standard character set environments, since APS detection of lower case letters only applies to standard ASCII letters.

```
Combination Lowercase=2
```

Combination Digits

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting defines the number of numeric digits ("0" to "9") needed to satisfy one of the possible Minimum Combinations of character types required in the password.

```
Combination Digits=1
```

Combination Alphanumeric

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting defines the number of alphanumeric characters ("A"-"Z" or "0"-"9") needed to satisfy one of the possible Minimum Combinations of character types required in the password. If this setting is used along with one of the settings above, characters can satisfy both requirements. For example, if Digits is four and this setting is four, the password 1234 satisfies both requirements.

```
Combination Alphanumeric=1
```

Combination Punctuation

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting requires that the password contain a certain minimum number of punctuation marks in order to satisfy *one* of the Minimum Combinations of character types required in the password. These can be periods, commas, exclamation marks, etc.

```
Combination Punctuation=1
```

Combination Symbols

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting requires that the password contain a certain minimum number of symbol characters in order to satisfy one of the character types required by the Minimum Combinations setting. Symbol characters are defined within APS as the characters:

"~" (tilde)	"@" (at)	"#" (number)
"$" (dollar)	"%" (percent)	"^" (circumflex)
"&" (ampersand)	"*" (asterisk)	"(" (open parenthesis)
")"(close parenthesis)	"_" (underscore)	"-" (hyphen)
"+" (plus)	"=" (equals)	"{" (open brace)
"}" (close brace)	"[" (open bracket)	"]" (close bracket)
"<" (less than)	">" (greater than)	"/" (virgule)
"\" (back slash)	"|" (vertical bar)	

and all extended ASCII characters, including diacritical marks. Note that some browsers do not support the entry of extended ASCII characters and some LDAP directories do not support them as part of an LDAP attribute.

```
Combination Symbols=1
```

Combination Other

Range: 0-32 characters

Default: 0

Recommended: 0

Complexity Level: Advanced

This specified that the password contain a specified minimum number of non-alphanumeric characters in order to satisfy one of the character types required by the Minimum Combinations setting. This includes punctuation marks and other symbols located on the keyboard.

Similar to the Combination Alphanumeric field above, a given character can satisfy this requirement in addition to "Punctuation" or "Symbols".

```
Combination Other=1
```

Minimum Combinations

Range: 0-7 types

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting tells how many different character types must be represented in the new password. In order for a given character type to be counted, one of the Combination Xxx settings (above) must be satisfied.

If any Combination Xxx setting is not zero and this value is not set (or zero), a warning will be issued in the SiteMinder Authentication Log.

If there are not enough Combination Xxx settings defined to possibly satisfy this setting, then this setting will be ignored and a warning will be logged to the SiteMinder Authentication Log.

```
Minimum Combinations=3
```

Other Content Requirements

Maximum Repeat

Range: 0-32 characters

Default: 0

Recommended: 3

Complexity Level: Basic

This setting controls the number of identical characters that cannot appear consecutively. For example, if this setting is four, then aaaa could not appear anywhere in the password.

```
Maximum Repeat=3
Maximum Repeat={@Employees} 2
```

Complexity

Range: 0-400

Default: 0

Recommended: 40

Complexity Level: Advanced

APS can numerically evaluate the complexity of a password (See Password Complexity (see page 130) for details on how APS evaluates password complexity). If the complexity of the new password does not exceed this value, the password is not allowed.

To turn off this checking, set to zero or comment it out.

```
Complexity=40
```

Force Case

Range: upper, lower, or none

Default: none

Recommended: none

Complexity Level: Advanced

Controls whether passwords should be forced to upper or lower case. CA does not recommend the use of this keyword, since it reduces the possible combinations of characters that can be used for passwords (thus easing the duties of a programmatic attack).

APS "knows" upper and lower case based on the standard ASCII character set. It does not recognize the difference between upper and lower case letters in the extended character set (such as those with diacritical marks).

If this option is used, the site must also use the Authentication Scheme wedge described on page 264 to convert entered passwords to the correct case.

Note: Sites should be very careful when implementing this setting for the first time. Once turned on, all passwords entered by users will have their case changed. If users have existing passwords that do not satisfy the new case requirements, they will not be able to login.

To turn off this checking, set it to none or comment it out.

```
Force Case=none
```

Match

Range: n/a

Default: none

Recommended: none

Complexity Level: Advanced

The Match keyword specifies patterns ("regular expressions") that new passwords must match in order to be valid. Unlike most keywords, the Match keyword may appear as many times as required, with or without overrides (the same override may appear more than once). All applicable patterns will be applied.

Each value must include two parts, an error key and a pattern, separated by one or more spaces.

The error key is an index into the APS.lang file for the error message to display if the new password fails to match the associated pattern. There is no default for this value. If Match is to be used, your site must specify the error key and must update the APS.lang file to provide an error message.

To turn off this checking, comment out all values.

The following example requires that a password start with an alphabetic character. Note that a key named ERR_MUST_START_ALPHA must be added to the APS.LANG file so that a proper error message can be displayed.

```
Match=ERR_MUST_START_ALPHA [A-Za-z]*
```

NoMatch

Range: n/a

Default: none

Recommended: none

Complexity Level: Advanced

The NoMatch keyword specifies patterns ("regular expressions") that new passwords must not match in order to be valid. Unlike most keywords, the NoMatch keyword may appear as many times as required, with or without overrides (the same override may appear more than once). All applicable patterns will be applied.

Each value must include two parts, an error key and a pattern, separated by one or more spaces.

The error key is an index into the APS.lang file for the error message to display if the new password matches the pattern. There is no default for this value. If NoMatch is to be used, your site must specify the error key and must update the APS.lang file to provide an error message.

To turn off this checking, comment out all values.

The following example requires that a password not end in a digit (preventing things like *PASSWORD9*). Note that a key named ERR_NO_TRAILING_DIGIT must be added to the APS.LANG file so that a proper error message can be displayed.

```
NoMatch=ERR_NO_TRAILING_DIGIT *[0-9]
```

Allowed Characters

Range: Character list

Default: none

Recommended: none

Complexity Level: Advanced

The Allowed Characters keyword specifies a list of characters that are allowed in a password. Only characters listed with this keyword will be allowed in new passwords.

Each instance of this keyword can specify a list of characters. They may or may not be surrounded by double quotes. Since leading and trailing blanks in a setting value are ignored, these quotes may be necessary. If the value is surrounded by quotes, they will be removed from the list of allowed characters (though any contained quotes will be retained).

Multiple instances of this keyword may exist and may apply. APS will use the characters listed with *every* applicable instance of this setting.

If *no* Allowed Characters keyword is valid, then all characters will be allowed (subject to the Disallowed Characters setting below).

APS does not detect characters that are both allowed and disallowed (they will be disallowed).

```
Allowed Characters=abcdefABCDEF01234
```

Disallowed Characters

Range: Character list

Default: none

Recommended: none

Complexity Level: Advanced

The Disallowed Characters keyword specifies a list of characters that are *not* allowed in a password. Characters listed with this keyword will not be allowed in new passwords.

Each instance of this keyword can specify a list of characters. They may or may not be surrounded by double quotes. Since leading and trailing blanks in a setting value are ignored, these quotes may be necessary. If the value is surrounded by quotes, they will be removed from the list of allowed characters (though any contained quotes will be retained).

Multiple instances of this keyword may exist and may apply. APS will use the characters listed with *every* applicable instance of this setting.

If *no* Disallowed Characters keyword is valid, then all characters will be allowed (subject to the Allowed Characters setting above).

APS does not detect characters that are both allowed and disallowed (they will be disallowed).

```
Disallowed Characters=xyzXYZ56789
```

Password Reuse

The password reuse policy determines when and whether previously used passwords may be reused by the same user. These fields allow an administrator to dictate the minimum number of passwords a user may use in a password cycle. If both values are set, APS uses the higher of the two. In other words, if the settings are a count of 12 and a delay of 365 days, the user may not reuse passwords used in the last year. After a year, if only six passwords have been used, another six would have to be used before the user can go back to the first password.

Values are checked both forwards and backwards, and are not case-sensitive. To turn password reuse checking off, set both values to zero or do not put the keywords into the file (comment them out).

APS will always keep the larger of the Count or one year, regardless of these settings. This is so that these settings can be turned on later without repercussions.

There is an internal restriction on the length of the password history. APS will keep a maximum of only 24K of information (dates and values, encrypted). Thus, programmatic password changers (that might be used for load/volume testing) might be able to re-use a password before these restrictions are satisfied. Only an automated password changer can change a password frequently enough. The password history has duplicate compression, so changing it and changing it back programmatically will not overly enlarge the history.

APS cannot limit users to changing their password once per day. However, the purpose of such a limitation is to prevent users from setting a new password, then setting it back immediately. To accomplish the same purpose, set Reuse Delay to 1. The user can then change their password as many times as they want, but will not be able to set it back for 1 day. See page 67 for a further discussion of this feature.

Reuse Count

Range: 0-500

Default: 0

Recommended: 12

Complexity Level: Intermediate

This controls how many passwords must be used before they can be reused.

```
Reuse Count=12
Reuse Count={@Customers} 500
```

Not supported on Windows NT Domain User Directories.

Reuse Delay

Range: 0-3650

Default: 0

Recommended: 365

Complexity Level: Intermediate

Controls how much time must elapse before a password can be reused.

Note that there is no need for a setting limiting how often a user may change their password. If this setting is set to one, then the user may change their password as many times as they want, but won't be able to reuse a password for 24 hours.

```
Reuse Delay=1
Reuse Delay={@Employees} 365
```

Not supported on Windows NT Domain User Directories.

Change Required

These settings are used to require that the user change *a* certain amount of his password each time that it is changed. Thus, the user can be prevented from changing their password from ABCD1 to ABCD2 if you wish.

There are two settings here. To turn off this checking, set Percentage to zero or comment it out.

Percentage checking takes two forms:

- If Percentage Sequencing is OFF (not specified), then the percentage checking is on overall character counts. APS counts the number of each character and the new password cannot share Percentage percent of common characters. 100% could make life difficult for users.
- If Percentage Sequencing is ON (specified), then the percentage checking is on a character by character basis. Percentage percent of each character position must change. This does not detect a change between BASEBALL12 and 12BASEBALL.

Percentage

Range: 0-100

Default: 0

Recommended: 0

Complexity Level: Advanced

This setting determines the percentage of the characters within a new password that must be different from the last one. A setting of 100% would require that all characters be different from one password to the next change.

```
Percentage=0
Percentage={@Customers} 50
```

Percentage Sequencing

Range: Yes or No

Default: No

Recommended: Yes

Complexity Level: Advanced

If this option is selected, the actual position of the characters in the new password is used when Password Services determines the actual percentage difference.

```
Percentage Sequencing
```

Restricting Words from the User's Profile

These settings, along with the password dictionary (see Dictionary keyword), control what sequences are not allowed in a password.

Attribute Match Maximum

Range: 0-32 characters

Default: 0

Recommended: 4

Complexity Level: Intermediate

Probably the worst passwords are those that are based in whole or in part on personal information about the user, such as their phone number or zip code. This setting controls the minimum sequence length checked by APS against attributes in the user's Directory entry.

If this value were set to four, the user could not include, for example, the last four digits of their phone number in a password.

You can turn this checking off by setting it to zero or commenting it out.

Any value less than 4 can be troublesome to your users.

For LDAP directories, organizational Units and other containers above the user are not checked. However, the user's DN *is* checked. For LDAP users, the objectClass attribute is automatically ignored.

This setting is used for both LDAP and ODBC directories. It is not supported for Windows NT User Directories.

Prior to Version 4.0, this setting was called "LDAP Attribute Match Maximum". At Version 4.0, it was changed to reflect support for ODBC directories. If the older name is found by APS, its value will be used, but a warning will be issued to the APS log.

```
Attribute Match Maximum=4
```

Parse Attributes

Range: n/a

Default: none

Recommended: cn (for LDAP), FullName (or equivalent for ODBC)

Complexity Level: Intermediate

When APS checks passwords against user attributes, this keyword specifies that certain attributes should be *parsed*. When parsing, each "word" in the attribute value greater than two characters in length will be matched against the requested password (it is *not* subject to the Match Maximum above).

No attributes will be checked at all if the match maximum is zero.

Parsing is the process of breaking up the value into words (or *tokens*). Words are considered any sequence of consecutive letters or numbers.

All attributes for a given setting must be placed on the same line, separated by commas.

Prior to Version 4.0, this setting was called "Parse LDAP Attributes". At Version 4.0, it was changed to reflect support for ODBC directories. If the older name is found by APS, its value will be used, but a warning will be issued to the APS log.

```
Parse Attributes=cn,title
Parse Attributes={IsLDAP()} cn
Parse Attributes={IsODBC()} FullName
```

Exclude Attributes

Range: n/a

Default: none

Recommended: none

Complexity Level: Intermediate

This keyword specifies the names of attributes that are to be *excluded* from all matching (parsed or matched). Regardless of this setting, the LDAP attribute objectclass will always be excluded.

All attributes for a given setting must be placed on the same line, separated by commas.

If an attribute is specified as both excluded and parsed, it will be excluded.

Prior to Version 4.0, this setting was called "Exclude LDAP Attributes". At Version 4.0, it was changed to reflect support for ODBC directories. If the older name is found by APS, its value will be used, but a warning will be issued to the APS log.

```
Exclude Attributes=uid,description
Exclude Attributes={IsLDAP()} uid
```

Run-time Password Checking

These settings are used during the Authentication process to determine if the user can log in or if APS must force the user to do or see something.

Range: Yes or NoReset Password

Default: No

Recommended: No

Complexity Level: Basic

SiteMinder may not be the only application that uses an LDAP directory for authentication. APS can obfuscate the user's password when it disables an LDAP user account. This prevents a disabled LDAP user account from using any application, not just SiteMinder protected ones. This means that when the account is to be enabled, not only must he be removed from the disabled group, but the password must be reset as well.

This also means that the administrator who resets the user's account will know the password after it is reset, however. By removing this setting, the administrator needs only to enable the account to allow authentication; the password will remain intact. Note that the administrator cannot tell the user what the password was, so it is assumed that the user really does know his original password.

The default is do not reset passwords.

Prior to APS Version 3.0, there were implications to turning off this setting. These implications no longer exist.

This setting applies to LDAP directories only.

```
    Reset Password
```

Auto Force Change

Range: Yes or No

Default: No

Recommended: Yes

Complexity Level: Intermediate

Starting with Version 3.0, APS can detect if an LDAP (or ODBC at Version 4.0) user's password was changed using an interface other than an APS interface. If this setting is enabled, then if the password has been changed externally, the user will be required to change their password upon their next login.

This is useful for Help Desk applications. For Administrator Password Reset, they need only change the user's password. Once done, and this setting is enabled, the user will automatically be forced to change their password.

To enable this capability for accounts stored in an ODBC directory, the query Set Password Checksum and the query Test Password Checksum must be defined and implemented.

For LDAP implementations, this functionality requires that the underlying LDAP server allow APS to read back the user password attribute in a consistent manner (usually hashed or encrypted). For iPlanet LDAP Directories, this can only be done if the administrator is cn=Directory Manager. For some LDAP implementations, most notably Microsoft Active Directory, there is no way to read this information.

```
Auto Force Change
```

Reset FPS Lockout

Range: Yes or No

Default: No

Recommended: Yes

Complexity Level: Basic

When a user successfully authenticates and this setting is enabled, any FPS Lockout Counter will be reset to zero.

```
Reset FPS Lockout
```

Failure Tracking

Max Failures

Range: 0 or 3-9

Default: 0

Recommended: 5 or 9

Complexity Level: Basic

Password cracking programs attempt to log into accounts by trying various combinations of passwords until one works. A counter to this kind of attack is to disable a user account after a specific number of invalid passwords have been entered consecutively.

This setting tells APS how many consecutive failed passwords must be supplied before an account is disabled.

APS keeps this counter in memory and stores it in a user's LDAP or ODBC entry (Windows users are kept in memory only). The value stored for a user contains a date/time (in GMT) in addition to the actual number of failures. Whenever APS needs the count for a user, it examines the in-memory counter and any value read from disk and uses the one tagged with the latest date/time. This is so that APS can continue to support Max Failures lockout even if a primary LDAP server is unavailable. Note that user accounts cannot be disabled in this case, but will be locked out in memory anyway.

Each server also keeps a date/time stamp on each counter in memory. If there has been no activity for a particular user after a specified time, the counter is cleared. This timeout value is configurable.

This setting should not be confused with the retry count supported by SiteMinder's Form Login capability or with the retry count supported by most browsers. Those retry counts control how many attempts can be made in a single session before the user must either shut down the browser or reselect the URL. This has no impact on the user's entry in the directory. Password cracking programs understand this capability and can shut down, then restart, browsers as required.

The Failure Count setting should usually be higher than any retry count (the Form Login retry count is configurable; most browsers use a value of 3 and cannot be changed). This allows a user a chance to realize that they have forgotten their password before being locked out by APS. Only a determined user (or a password cracker) would continue to make attempts and thus cause the user record to become disabled.

Further discussion of this issue is described in the section entitled Special Case: Three Strikes, You're Out.

The Max Failures setting must be used with great care when using complex LDAP or multiple directories. To understand why, you must understand how SiteMinder locates users in its directories:

SiteMinder usually receives an unqualified user ID. It does not know which directory the user is stored in, nor where, for hierarchical (LDAP) directories, the user exists within the directory. For example, a site might have three different users called JSMITH, one in the NT domain, and two in LDAP as uid=JSMITH,ou=employees,o=Airius.com and uid=JSMITH,ou=customers,o=Airius.com. SiteMinder only receives the id JSMITH and a password.

SiteMinder searches through its directories looking for a match to the supplied user id. When a match is found, the server attempts to log into (bind to) that account using the supplied password. If the password matches, then it has found the correct user, otherwise it continues its search to the next match on the user id.

If no userid is found with the supplied password, the authentication request is denied.

Now consider how APS works. Each time that a user entry is found, SiteMinder notifies APS and tells it whether the entry was authenticated or not. If authenticated, APS resets any counter associated with the name and checks for to see if the account is disabled, etc.

However, if the authentication for this entry has failed, APS increments the failure count for the entry. If the count exceeds this setting, the user account is disabled and the specified actions (mail, redirection) take place.

This can cause real problems when multiple accounts exist with the same id. In our above example, each time that JSMITH logs in, if it is the third JSMITH found, then the first two will each have their counts incremented and, possibly, be inadvertently disabled.

Consider also the case where one of the JSMITHs has forgotten her password. After three attempts, all three JSMITH accounts will be disabled. This is probably not what an administrator wants, but is the action that APS must take. A cracker program in this case is more likely to succeed, since there are three passwords that might work instead of just one. Also consider that APS does not know which instance of that user actually forgot their password.

Keep these issues in mind when using this setting. If duplicate user ids are possible in your directories, it might not be a good idea to use this setting or it might be a good idea to set this to a higher value.

See also How Do I: Configure "3 Strikes and You're Out"?.

```
Max Failures=5
```

Max Failures On Change

Range: 0 or 3-9

Default: 0

Recommended: 3

Complexity Level: Basic

This setting controls how many unsuccessful attempts to change the password are allowed (unsuccessful because the old password is not correct). If this setting is not specified, the Max Failures setting above will be used.

Note that these two settings use the same counter, but set different thresholds for failure.

```
Max Failures on Change=5
```

Failure Count Timeout

Range: 0 or 5-30 minutes

Default: 0

Recommended: 5

Complexity Level: Intermediate

If an account is disabled because of a failure count, this setting determines how long the user account will be disabled (if Auto Reset Failure Count, below, is enabled). If the user does not attempt to login for this amount of time, the user will be allowed to login. If the user is disabled (Auto Reset is off), he remains disabled. If not set and Auto Reset Failure Count is enabled, this value will be internally set to 5 minutes. This setting is in minutes. This value will be ignored if it is less than 5 minutes.

This setting must be larger than the Failure Count Retention setting below. This number should be set higher than 30 minutes only with great care.

If a user attempts to login within the period, regardless of success or failure, the time period will restart. Thus, it is impossible for a programmatic password cracker to break a password, even if this setting is as low as 5, since 5 minutes must pass between attempts (to test one million passwords would take approximately 9.5 years.

```
Failure Count Timeout=10
```

Auto Reset Failure Count

Range: Yes or No

Default: No

Recommended: Yes

Complexity Level: Basic

If this setting is specified, users will not be permanently disabled when they reach the failure count (Max Failures), but will not be allowed to login until the timeout (specified by Failure Count Timeout above) occurs. Any attempt to login (good password or bad) will reset the timer. This feature is off by default. If the keyword appears, then the feature is turned on.

```
Auto Reset Failure Count
```

Failure Count Retention

Range: 0 or 5-30 minutes

Default: 0

Recommended: 5

Complexity Level: Advanced

APS keeps track of the failure count for each user both in memory and on disk (for ODBC and LDAP users). This setting tells APS how long to remember these counts. This is different than the Failure Count Timeout setting (above) that controls how long an account will be disabled before automatically re-enabling it.

If no failed attempts occur within this period, APS will reset the counter to zero.

This setting must be lower than (or equal to) the Failure Count Timeout setting above. This number should be set higher than 30 minutes only with great care.

If a user fails a login attempt within the period the time period will restart. Thus, it is impossible for a programmatic password cracker to break a password, even if this setting is as low as 5, since 5 minutes must pass between attempts (to test one million passwords would take approximately 9.5 *years*.

If this setting is less than 5 or not set, APS will use 5 minutes.

```
Failure Count Retention=10
```

How to Configure a Separate Maximum Failure Counter and Threshold for OTP Authentication

Password cracking programs attempt to log in to accounts by trying various combinations of passwords until one works. A counter to this kind of attack is to disable a user account after a specific number of invalid passwords have been entered consecutively.

The Advanced Password Services Max Failures parameter allows you to specify the maximum number of consecutive failed passwords that can be supplied before an account is disabled.

By default, Max Failures is applied and the failure count is aggregated across all authentication mechanisms. Therefore, if a user exceeds the Max Failures threshold using any combination of authentication schemes, they are locked out. However, you can override Max Failures to configure an independent failure attempt counter and threshold for realms that are configured with OTP authentication.

A separate failure counter is maintained for overridden realms; user login failures are not aggregated between realms that are configured with OTP and all other realms.

You override Max Failures for realms that are configured with OTP authentication by specifying the following two parameters in the APS.cfg file:

Override Max Failure for Realm OIDs

Specifies the Object ID (OID) or OIDs of OTP-protected realms for which to override the Max Failure threshold.

Realm based Max Failures

Specifies the number of consecutive failed logins attempts a user can make on overridden realms before their account is disabled.

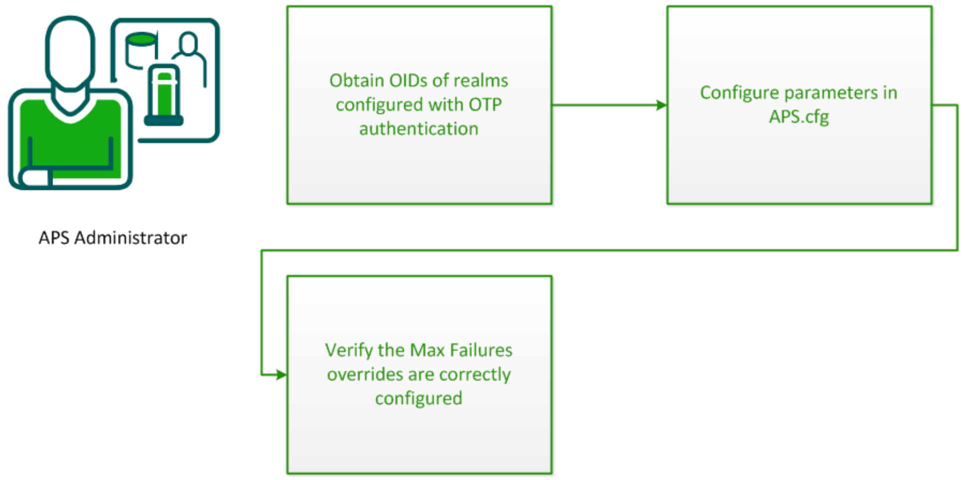
For example, consider a site that is configured to allow users to log in using a standard password *or* a one-time password (OTP). To configure APS to allow three failed standard password attempts or five failed OTP attempts before disabling a user account, specify the following parameters in APS.cfg:

- Max Failures is set to 3.
- Override Max Failure for Realm OIDs specifies the OID of the realm that is protected by the OTP.
- Realm based Max Failures is set to 5.

In this case, a user can now execute two bad username and password attempts *and* four bad OTP attempts without their account being locked. However, if they make three bad username and password attempts or six bad OTP attempts they are locked out.

All failure counters are reset for a user account when either of the following events occurs:

- The user successfully authenticates using correct credentials for any authentication mechanism.
- An administrator enables the user account. For more information, see the *Advanced Password Services Guide*.


APS Administrator
Obtain OIDs of realms configured with OTP authentication
Configure parameters in APS.cfg
Verify the Max Failures overrides are correctly configured

1. Obtain the OIDs of realms that are configured with OTP authentication (see page 105).
2. Configure the required parameters in the APS.cfg file (see page 106).
3. Verify that the Max Failures value is overridden for OTP authentication (see page 107).

Obtain the OID of Realms Protected Configured for OTP Authentication

Obtain the Object Identity (OID) of the realms that are configured with OTP authentication for which you are configuring Max Failures overrides.

Follow these steps:

1. Log in to the Policy Server using an Administrator account with XPSexplorer rights.
2. Open a command window and enter the following command:

   ```
   XPSexplorer
   ```
3. Type "42" and hit Enter to explore realms.
4. Perform the following steps for each realm whose OID require:
   a. Type "s" and hit Enter to search all realms.
   b. Type "b" and hit Enter to build a filter to search for a specific realm.
   c. Type "5" and hit Enter to configure the filter to search for a realm Name attribute.
   d. Type "a" and hit Enter to specify the equals ("=") operation for the filter.
   e. Type the exact name of the realm and hit Enter twice.
   f. Type "f" and hit Enter to apply the filter.

g. If you entered the realm name correctly, XPSexplorer displays information about the realm. For example:

```
1-CA.SM::Realm@06-a0cf82a7-d831-453d-ac9e-8ed814f90369
```

h. Note the realm ID, which is everything after the @ sign. In the previous example, the following string is the realm OID:

```
06-a0cf82a7-d831-453d-ac9e-8ed814f90369
```

i. Type "q".

Configure Required Parameters in the APS.cfg File

Define the parameters to configure separate maximum failure attempts for one-time passwords in the APS.cfg configuration file. APS.cfg is located in one of the following locations:

- Windows: *siteminder_home*\siteminder\bin
- UNIX: *siteminder_home/*siteminder/bin

  Note: On UNIX, APS.cfg can also be stored in another location that is referenced by the APS_SETTINGS environment variable.

  siteminder_home

  Specifies the Policy Server installation path.

Edit APS.cfg using a standard text editor.

Configure the Max Failures Parameter

Configure the Max Failures parameter to specify the maximum number of consecutive failed password attempts before an account is disabled. Max Failures is applied across all realms that are not overridden.

Note: Max Failures may already be configured in your environment. If so, verify that it is set to an appropriate value.

Max Failures

Specifies the maximum number of consecutive failed passwords that can be supplied before an account is disabled.

Default: 0

Limits: Positive integer

Example

```
Max Failures=3
```

Configure the Override Max Failure for Realm OIDs Parameter

Configure the Override Max Failure for Realm OIDs parameter to specify the OIDs (obtained in the previous procedure) of realms configured with OTP authentication.

Override Max Failure for Realm OIDs

Specifies the OID or OIDs of realms for which the Max Failure threshold is overridden by the value specified by the Realm based Max Failures parameter.

Default: None

Limits: Valid realm OID or semicolon-separated list of realm OIDs

Examples

```
Override Max Failure for Realm OIDs=06-a0cf82a7-d831-453d-ac9e-8ed814f90369
```

```
Override Max Failure for Realm OIDs06-a0cf82a7-d831-453d-ac9e-8ed814f90369;
06-1cdeddf8-9346-4626-8c5b-6d674c8fb302
```

Configure the Realm Based Max Failures Parameter

Configure the Realm Based Max Failures parameter to specify the maximum number of consecutive failed password attempts for realms configured with OTP authentication.

Realm based Max Failures

Specifies the number of consecutive failed passwords attempts for realms specified by the Override Max Failure for Realm OIDs parameter.

Default: 0

Limits: Positive integer

Example

```
Realm based Max Failures=5
```

Verify That Attempt Failure Thresholds for OTP and Other Authentication Do Not Aggregate to Lock Out Users

Verify that the separate maximum attempt failure threshold that you have defined for a realm that is configured with OTP authentication is correctly configured.

The following procedure assumes the following conditions:

- Max Failures is configured to allow *M* failed login non-OTP authentication attempts before a user is locked out.
- Realm based Max Failures is configured to allow *N* failed OTP authentication attempts before a user is locked out.

Follow these steps:

1. Attempt to access a resource that is secured with OTP authentication using an incorrect password *N*-1 times.

   The account should not be locked.

2. Attempt to access a resource that is secured with non-OTP authentication using an incorrect password *M*-1 times.

   The account should not be locked.

3. Attempt to access a resource this is secured with OTP authentication using an incorrect password one more time (for a total on *N* times).

   The account should be locked.

4. Reenable the locked out account.

You have successfully configured a separate maximum attempt threshold for one-time password authentication.

Expiring Password

Password Expiration

Range: 0 or 30-180 days

Default: 0

Recommended: 90

Complexity Level: Basic

If a security breach does occur because a password becomes known, the breach is immediately closed when the password changes. By requiring your users to periodically change their passwords, you can close any breaches that you don't even know about.

This setting controls how often users must change their passwords.

When a password fully expires, the user is disabled and the configured action taken (email, redirection).

This setting controls the amount of time that can pass between when the user last changed his password (or was created) and when the password must be changed. Related settings, described below, control warnings and grace periods.

For users stored in a Windows NT Directory, Expiration Delay is not honored. APS honors the password expiration date set in the NT User record. Expiration Warning is honored as described below, though it is relative to the expiration date set in the NT user record. Expiration Grace and Grace Logins are not honored for NT users, thus, when an NT user's password expires, that user is immediately disabled.

```
Password Expiration=90
Password Expiration={@Employees} 60
```

Note: For users stored in LDAP or ODBC directories, the administrator can modify some of the behavior of APS when doing a password expiration check. Before using this setting, APS will check the user's record for a value for the attribute smapsExpirePasswordDays. If set, APS will use the number of days so specified. This feature should be rarely, if ever, used, since it causes a performance hit and creates maintenance problems. It is intended to override specific users, such as administrators, whose password should never expire.

Expiration Warning

Range: 0-99 days

Default: 0

Recommended: 7

Complexity Level: Basic

When a password expires, the user's access to the system is revoked. It is far user-friendlier to warn the user that their password will expire shortly. This setting determines how long before a password expires should the user be warned.

The warning takes the form of email and/or redirection (which must be configured for this setting to work) occurring each time that the user logs in during this period.

```
Expiration Warning=7
Expiration Warning={@Customers} 10
```

Offline Expiration Warning

Range: 0-99 days

Default: 0

Recommended: 7

Complexity Level: Basic

If a user logs in during the Expiration Warning period, defined as the Password Expiration Date less the number of days specified by the Expiration Warning setting above, an event fires and the system takes immediate action (redirection and/or mail sent). However, this process is triggered by an actual authentication attempt.

This setting allows a site to send email warning users of password expiration *even* if the user does not authenticate. The APSExpire utility will send email to users before their password expires using this setting.

Note that this setting may appear more than once. If so, APSExpire will send the email each time within the period that it is run. Thus, a site can send email 15, 10 and 5 days before the password actually expires.

```
Offline Expiration Warning=15
Offline Expiration Warning=10
Offline Expiration Warning=5
```

Expiration Grace

Range: 0-99 days

Default: 0

Recommended: 7-14 days

Complexity Level: Intermediate

Once password expiration is reached, APS no longer treats password changes as optional (it becomes required). How many days after the password expires should APS actually disable the user?

This setting is not honored for users in a Windows NT Domain Directory.

```
Expiration Grace=30
Expiration Grace={@Employees} 7
```

Grace Logins

Range: 0-5

Default: 0

Recommended: no

Complexity Level: Advanced

Once password expiration is reached, APS no longer treats password changes as optional (it becomes required). This setting indicates how many times the user will be allowed to login after the password expires and before the user is disabled.

All but the last grace login are optional. The last grace login will not be optional (using the default form and AZRedirect).

If both Grace Logins and Expiration Grace settings are used, the first value to be used up will cause the user to be disabled. In other words, if the Expiration Grace period expires before the user has consumed all of the available Grace Logins, the user is disabled. If all Grace Logins are consumed, the user is disabled, even if the Expiration Grace period has not expired.

Note that the user will be allowed to login when this value is reached. However, if the password is not changed, the next login attempt will cause the user to be disabled.

This setting is not honored for users in a Windows NT Domain Directory.

```
Grace Logins=3
Grace Logins={@Employees} 2
```

Account Expiration & Purge

Max Inactivity

Range: 0-365 days

Default: 0

Recommended: 90 days

Complexity Level: Basic

This setting controls the maximum amount of time that can occur between user logins. A user attempting to log into an account that has remained dormant for more than this amount of time will cause the account to become disabled. Thus, an old, dormant account that may have been overlooked will be less of a security threat.

For example, one of your employees leaves the company to join one of your competitors. Your Human Resources department does not notify you of the separation. Since the account remains unused for 30 days (before the employee attempts to reconnect), when the attempt is made, the account is disabled (and, optionally, mail can be sent to the administrator).

By default, this setting is reactive, meaning that the account is not actually disabled until it is used, not when the time limit expires. To be proactive, that is, to have APS disable users (LDAP and ODBC users only) when the account actually expires, use the APSExpire utility.

When APSExpire is used, if the user eventually logs in (after being disabled), APS cannot do special redirection to notify the user that he has been disabled due to inactivity, only that the user is disabled.

If APSExpire is used, then you can send warning mail to expiring users several days before the account actually expires. This functionality is controlled by the Inactivity Warning setting below.

For Windows NT Directories, APS honors the Account Expiration date set in the NT User record rather than this setting.

```
Max Inactivity=90
Max Inactivity={@Admins} 180
```

Note: For users stored in LDAP and ODBC directories, the administrator can modify some of the behavior of APS when doing this check. Before using this setting, APS will check the user's record for a value for the attribute **smapsAccountInactivityDays**. If set, APS will use the number of days so specified. This feature should be rarely, if ever, used, since it causes a performance hit and creates maintenance problems. It is intended to override specific users, such as administrators, that are accessed rarely and should never be disabled due to inactivity.

Inactivity Warning

Range: 0-99 days

Default: 0

Recommended: 7-10 days

Complexity Level: Intermediate

If the APSExpire utility is used, then you can send warning mail to expiring users several days before the account actually expires. This setting controls how long before the user record expires such mail is sent.

The mail may not actually be sent this number of days before expiration, since the APSExpire utility may not run. The utility will generate mail when a user's record will expire within this number of days and such mail has not already been sent.

Since APSExpire can only be used for LDAP or ODBC users, this setting does not apply to users in a Windows NT directory.

```
Inactivity Warning=5
Inactivity Warning={@Customers} 10
```

Purge After

Range: 0-730 days

Default: 0

Recommended: 30 days

Complexity Level: Advanced

An account disabled due to inactivity will remain in the User Directory, theoretically forever. The APSAdmin utility will report all users that have been disabled (due to Account Inactivity) for this many days as "eligible for purge". Note that APS will never actually delete a user.

Since APSExpire can only be used for LDAP or ODBC users, this setting does not apply to users in a Windows NT directory.

```
Purge After=30
Purge After={@Customers} 365
```

Event Redirection

When events occur during the authentication process, APS can redirect the user to specific pages (URLs). This section contains descriptions of the settings that specify the URLs to redirect users for each event.

During authentication, APS records the fact that the event has occurred and which event it was. It cannot perform the redirection at that time.

In order to actually perform the redirection, additional configuration settings must be created within the Policy Database. This configuration is described in the configuration section of this document.

Two different actions can be taken for each event:

- The user can be redirected to another page (or a message displayed),
- Email can be sent to the user and/or an arbitrary user (administrator)

Various notes apply to redirection when used as an action associated with an event:

Active Expression Parameters

The parameter field for the Active Expressions used for redirection (SmApsRedirect and AZRedirect) can be used for some very powerful functionality.

The parameter field is in the form:

```
<key>=<value>;<key2>=<value>;...;<default redirect>
```

All fields are optional and the system works fine without specifying any parameters.

The <default redirect> parameter, if required, must be the last section in the parameter list. It cannot contain an equal sign or a semicolon (it is possible to escape such characters with a backslash "\"). If no redirection is to be performed by APS, this is where the user will be redirected to instead. This is used when you want to use one of the redirect events, but it would conflict with APS.

You can specify as many Key/Value pairs as required. APS will save these values. The character case for keys is ignored. If the same key is specified more than once, only the last value will be used.

Values may contain user attribute names surrounded by curly braces ("{" and "}"). Such attribute names (and the braces) will be replaced by the attribute value from the user's directory entry. If the first part of the value will reference an attribute in this way and a setting override is not to be used, you must specify an override of "{TRUE}" in order not to confuse the APS.cfg parser (so it recognizes the attribute reference and can differentiate it from an override).

Event redirection can be overridden using the standard override syntax. In addition, context macros are also available. The *<key>* values passed through the Active Expression are used as context macros whose values are the <value> passed for the key. Some other event-specific context macros are also available (discussed with each event definition).

When a redirect must be determined, APS will evaluate *all* redirects of the correct type in the order in which they appear. The *last* one that applies will be used.

To override a redirection setting, specify a Key/Value pair after the equal sign, within square brackets, such as:

```
Failure Redirect={%LANG="EN"}http:/EnglishFailure.htm
```

If the parameter of the redirect response defined in the Policy were LANG={preferredLanguage}, then there will be a key called LANG with a value of the user's preferred language (from their LDAP entry). If the value from the user's entry were *EN* (for English), then the LANG key will have a value of EN, thus this setting would be selected (assuming it is not superceded by another setting later).

This is most useful to select redirections based on Realm. APS does not support this directly, but it can easily be performed using this feature by specifying a Key called REALM in each response, with a value appropriate for that realm. Overrides can then be defined in this file for each Realm. Realize that this is not truly tied to the realm, it is only a key supplied as a parameter to the redirect.

For example:

```
Failure Redirect={%Realm="AppA"} http:/AppA.htm
```

Will be fired if invoked by an active expression such as the following (presumably defined only in the ApplicationA realm):

```
<@ lib="smaps" func="SmApsRedirect" param="Realm=AppA" @>
```

But would not be fired in other realms where the REALM keyword was not set to AppA.

URL Translation

Once a redirection URL has been selected, APS will perform additional translations on that URL. In addition to the keys supplied in the Active Expression parameter, the following Keys will also be defined:

Server	The server component of the requested Resource URL.
Resource	The resource component of the requested Resource URL.
Action	The action component of the requested Resource URL.
Target	The full URL of the requested resource.

If the user context is defined (the user is authenticated), these keys will be defined as well:

UserName	The user's full DN
UserPath	The user's full path, including directory type and name.
DirPath	The path to the directory server.
DirServer	The directory server.
DirNamespace	LDAP:, ODBC: or WinNT:, depending on the origin of the user.

APS will take the resulting URL (or the default one specified in the parameter field) and replace any references to any key with its value. References to keys should be placed in the URL surrounded by angle brackets ("<" and ">"). To continue the example above:

```
http://www.mysite.com/TargetPage?Language=<LANG>
```

would be translated to:

```
http://www.mysite.com/TargetPage?Language=EN
```

Key names are NOT case-sensitive and values will be URL-encoded (so TARGET will be encoded).

In addition, any text between curly braces ("{" and "}") will be replaced (along with the braces) with equivalent values from the user's directory entry. Note that this processing is only possible if the user has been successfully authenticated. An example might be:

```
http://www.mysite.com/TargetPage? Language=<LANG>&Name={FirstName}
```

would translate (for me) to:

```
http://www.mysite.com/TargetPage? Language=EN&Name=Eric
```

SmCPW

For some of these redirects, the setting will probably be SmCPW, that component of APS that allows users to change their passwords. SmCPW recognizes many arguments in its URL (as described on page 272), including:

```
Target=<target>
DaysLeft=<DaysLeft>
Optional
```

Multiple arguments should be separated by an ampersand ("&"), thus a warning redirect might be:

```
/CPW/SmCPW.exe?DaysLeft=<DaysLeft>&Target=<Target>
```

Additional arguments (either from keys or hard-coded) may be supplied. SmCPW will ignore them, but will pass them on to its own target in the same manner.

Failure Redirect

Default: none

Recommended: no

Complexity Level: Intermediate

When the user reaches **Max Failures** consecutive failures, where should APS send the user? This must be an unprotected page, since the user will not be allowed to login. It is not recommended that this redirect be used (use the mail notification instead), since by using it, your site acknowledges that a valid user id has been discovered and thus you open yourself to a denial of service attack.

```
Failure Redirect= http://www.yoursite.com/maxfails.htm
```

Disable Redirect

Default: none

Recommended: no

Complexity Level: Intermediate

If a user is *already* disabled when a login attempt occurs, APS will send the user to the specified page. This must be an unprotected page, since the user will not be allowed to login. This redirect should *not* be used for the same reasons as stated for the Failure Redirect setting above.

This redirection supports an additional macro called DisabledReason. This contains the reason code(s) associated with why the user is disabled.

```
Disable Redirect= http://www.yoursite.com/disabled.htm?
Disabled=<DisabledReason>
```

Inactive Redirect

Default: none

Recommended: Yes

Complexity Level: Intermediate

If a user has not logged in for Max Inactive days (or the number of days specified by smapsAccountExpirationDays), where should APS redirect the user (after disabling him)? This must be an unprotected page, since the user will not be allowed to login.

This redirection is safe to use. In order for this event to occur, the user has supplied a valid user id *and* password (he just has not done it for too long). Using this setting does not open a security hole.

If the APSExpire utility is utilized, this redirection may never occur (it may occur if the user attempts to log in on the same day as he would expire and APSExpire has not yet run). APSExpire will disable the user in its batch process instead, so this event does not occur. Instead, the Disabled Redirect setting would apply.

```
Inactive Redirect= http://www.yoursite.com/inactive.htm
```

Expired Redirect

Default: none

Recommended: Yes

Complexity Level: Intermediate

If a user has not changed his password for Password Expiration (or the value stored in smapsPasswordExpirationDays) plus Expiration Grace days or has used up all available Grace Logins, where should APS redirect the user (after disabling him)? This must be an unprotected page, since the user will not be allowed to login.

This redirection is safe to use. In order for this event to occur, the user has supplied a valid user ID and password (he just has not changed it for too long). Using this setting does not open a security hole.

```
Expired Redirect= http://www.yoursite.com/expired.htm
```

Force Change Redirect

Default: none

Recommended: [path]SmCPW[.exe]?Target=<target>

Complexity Level: Basic

If a user is required to change his password immediately because the Force Password Change flag is set, where should APS redirect the user? This should be a protected page (probably the change password page).

This page may also be used if the Expire Change Redirect is needed but not specified.

```
Force Change Redirect= http://www.yoursite.com/CPW/SmCPW.exe? Target=<target>
```

Warning Redirect

Default: none

Recommended: [path]SmCPW[.exe]?Target=<target>& DaysLeft=<daysleft>

Complexity Level: Basic

If the user is being warned that his password will expire (starting Expiration Warning days before **Password Expiration**), where should APS redirect the user? This should be a protected page (probably the change password page).

```
Warning Redirect= http://www.yoursite.com/CPW/SmCPW.exe?
Target=<target>&DaysLeft=<daysleft>
```

Expire Change Redirect

Default: none

Recommended: [path]SmCPW[.exe]?Target=<target>

Complexity Level: Basic

If a user is required to change his password immediately because his password has expired and he is still in the grace period, where should APS redirect the user? This should be a protected page (probably the change password page).

If this setting is not specified, but needed, the **Force Change Redirect** will be used.

```
Expire Change Redirect= http://www.yoursite.com/CPW/SmCPW.exe?
Target=<target>
```

Generational & Other Automatic Redirection

Generational Redirects only apply to LDAP and ODBC User Directories.

Starting with Version 3.0, APS supports Generational Redirects. This type of redirect is distinctly different from the event redirects described in the previous section.

While Generational Redirects are not strictly in the purvey of APS, APS already has the infrastructure in place to provide this useful functionality.

Generational Redirects are redirections that are to be applied to the user base at least once. The generational nature is because there can be multiple generations for a specific URL, each user is required to "see" all known generations of that target.

Each generational redirect has a name that is used as a key to describe that redirect. Two typical uses of generation redirects might use PROFILE and LICENSE, in order to force each user to see the user profile and the on-line access agreement.

In addition to a name, each redirect also needs a generation, which is a number and a URL.

Generational Redirect

Default: none

Complexity Level: Advanced

When a user accesses a site, APS looks at values stored in the **smapsGenerationalRedirects** attribute of the user. This attribute is multi-valued (on LDAP) and will contain key/value pairs, such as:

```
LICENSE=2 <comment info, usually date & IP>
PROFILE=1 <comment info, usually date & IP>
```

APS compares the number associated with a specific key with the generation specified in the APS configuration file. If the number is less in the user record (or the key does not exist), the user will be redirected to the URL specified in the configuration file and the user's value for the smapsGenerationalRedirects attribute is updated with the new value.

Under normal conditions, only version 1 would ever be used. However, some sites might require, for example, that all users see an On-Line Access Agreement every time changes. In this case, merely update the version number in APS.cfg and all users will be sent to the associated URL (since new users have no value and existing users will have a stored generation of 1 instead of the new value, 2).

Unlike most settings in APS.CFG, all generational redirects will be processed, in the order in which they appear in the file.

Note that APS can only redirect the user to the URL, it cannot force the user to actually take an action once there.

APS will redirect the user to any applicable generational redirection pages after processing any applicable events. If multiple generational redirects are triggered, the user will be redirected to each, in the order that they appear in the APS.cfg file.

```
Generational Redirect=LICENSE=1 http://www.yoursite.com/Profile.jsp
Generational Redirect=PROFILE=1 http://www.yoursite.com/Profile.jsp
```

Message of the Day

Default: none

Complexity Level: Advanced

This setting will show the designated page to all applicable users the first (and only the first) time that they log in each day.

This setting uses the smapsGenerationalRedirects attribute to "remember" whether a user has been redirected on any given day, with a key value of APS_MOTD. The generation number will be the current date.

The user will be redirected to the message of the day after all APS events and all applicable generational redirects.

If multiple Message of the Day settings exist, the last which applies will be used.

```
Message Of The Day = http://www.yoursite.com/motd.htm
Message Of The Day ={@Employees} http://www.yoursite.com/EmpMotd.htm
Message Of The Day ={@Customers} http://www.yoursite.com/CustMotd.htm
```

Always Redirect

Default: none

Complexity Level: Advanced

This setting will show the designated page to all applicable users every time that the user authenticates.

This setting does not use the smapsGenerationalRedirects attribute.

If multiple Always Redirect settings exist, the last that applies will be used.

```
Always Redirect=http://www.yoursite.com/notices.htm
Always Redirect={@Employees} http://www.yoursite.com/EmpNotices.htm
Always Redirect={@Customers} http://www.yoursite.com/CstNotices.htm
```

Always Redirect AZ

Default: none

Complexity Level: Advanced

This setting will show the designated page to all applicable users every time that the user accesses a page. This setting should be used very carefully, since users will be unable to access all or part of your site.

It is often used to temporarily present a page to users that are trying to access part of your site that is being upgraded.

This setting does not use the smapsGenerationalRedirects attribute.

If multiple Always Redirect settings exist, the *last* that applies will be used.

This setting requires the use of the AZRedirect Active Expression.

```
Always Redirect AZ = {NOT @Tester AND %Realm="AppA"}
http://www.yoursite.com/offline.htm
```

Sending Mail

These settings specify the files that are to be sent by APS when the associated event occurs. There is no such thing as "most restrictive" setting in these cases; the *last* applicable setting will be used.

File names may contain full or relative paths or no path at all. APS will use the Directory setting in the [MAIL] section to locate files, if required.

File names may contain attribute replacement specifications, such as /Mail/{preferredLanguage}/Disabled.email.

Multiple files may be specified, separated by a semicolon (";"). If so, APS will send all of them, if possible. This is useful when one file should be sent to the user and another to an administrator.

The format of these files is described in the section entitled Email Templates.

If no mail is to be sent for a specific event, comment out or leave the setting blank.

If your User Directory does not have valid email addresses, you should not use these settings, except to send mail to internal personnel (fixed email addresses).

For Windows NT users, APS looks at the Description field in the user's entry. Within that field, APS looks for the string Email:. The text immediately following, up to the end of the description field or the next space, will be used as the user's email address.

Disabled User Mail

Default: none

Recommended: Yes, to user

Complexity Level: Intermediate

This setting specifies the file(s) to send when a disabled user tries to authenticate. This is useful to tell the user that they have entered correct information, but their user record is currently disabled. Note that if passwords are reset (see the Reset Passwords setting on page 71) users will fail authentication and not see this mail. This is only sent when they successfully authenticate, then are rejected by APS because the record is disabled.

This event supports a macro called DisabledReason. This will contain the reason code(s) associated with why the user is disabled.

```
Disabled User Mail=Disabled.email
```

Max Failures Mail

Default: none

Recommended: Yes, to user and administrator

Complexity Level: Intermediate

This setting specifies the file(s) to send when the maximum failure count is exceeded and the user is not going to be allowed to login. This is a very secure way to notify a user (or administrator) of a possible attack on the system, since a hacker will not receive the mail (and thus the user id is not confirmed and the hacker will continue attempts that can never succeed). The user will be notified why his mistaken logins do not work and that his account is disabled.

Administrators can be notified of the attack. This is often very usefully overridden for Administrator accounts, since it can be used to notify system or security administrators of attempts to access the system (most sites will not want System Administrator notification for all Max Failure events, only attacks on administrator accounts).

When this mail is sent, a macro called FailureCount can be included in the mail text itself. Its value is the actual number of failed attempts.

```
Max Failures Mail=MaxFailures.email
Max Failures Mail={@Administrators} MaxFailures.email; AdminMaxFailures.email
```

Ongoing Failures Mail

Default: none

Recommended: Yes, to administrator

Complexity Level: Advanced

When an account is under programmatic attack and the Max Failures setting is in effect, APS disables the user and sends the Max Failures Mail when the failure count first reaches the value of Max Failures. Presumably, Max Failures Mail will be sent to the user.

This setting, Ongoing Failures Mail, specifies the file(s) to send as the attack continues. Each Max Failures attempts will trigger the sending of the file(s) indicated in this setting. Under normal circumstances, this mail is sent to an internal administrator to notify that person that the attack continues.

Often, sites will configure this mail to go through an SMTP/Pager gateway so that the administrator is notified in real-time.

When this mail is sent, a macro called FailureCount can be included in the mail text itself. Its value is the actual number of failed attempts. It is often useful to include {SM_USERSESSIONIP} (note the braces) in the body of the mail so that the client IP address (the source of the attack) can be identified. This address is not necessarily trustworthy; it can be spoofed.

```
Ongoing Failures Mail=MoreAttacks.email
```

Inactive User Mail

Default: none

Recommended: Yes, to user and administrator

Complexity Level: Intermediate

If the user is disabled because he is inactive, this setting is used to find the file to send as mail. This setting is used during the authentication process and by the APSExpire utility.

```
Inactive User Mail=InactiveOnline.email
Inactive User Mail={%APSExpire="YES"} NotifyInactiveUsers.email
```

Expired Password Mail

Default: none

Recommended: no

Complexity Level: Advanced

This setting controls the mail file sent when a user's password actually expires (not when he enters the grace period). This is sent when the user is disabled because of password expiration. It can be used both for online expiration and by APSExpire.

```
Expired Password Mail= ExpiredPasswordOnline.email
Expired Password Mail={%APSExpire="YES"} NotifyExpiredPassword.email
```

Force Change Mail

Default: none

Recommended: no

Complexity Level: Advanced

If the user will be forced to change his password, this setting specifies the file that will be sent as email. It is not very useful, since redirection will actually force the password to be changed. It is not used by APSExpire.

```
Force Change Mail=ForceChangePassword.email
```

Password Warning Mail

Default: none

Recommended: Yes

Complexity Level: Intermediate

If the user must be warned that his password will expire, but does not yet have to change the password, the file specified by this setting is sent.

This is actually a very useful setting. Consider, for your site, using this setting instead of redirecting the user. By using email (presumably either containing the URL that the user can use to change their password or instructions as to where a link can be located on your site), you will not interrupt the user's workflow to request an optional password change.

This setting can be used by APSExpire. If, for example, a typical user of your site only is expected to log in quarterly, but password expiration warnings are given 10 days before expiration, users will rarely ever actually see such a warning at login time. You can use this setting with APSExpire to warn users even when they are not logging in frequently.

```
Password Warning Mail={%APSAdmin!="YES"} PasswordWarningOnline.email
Password Warning Mail={%APSAdmin="YES"} NotifyPasswordWarning.email
```

Inactivity Warning Mail

Default: none

Recommended: Yes

Complexity Level: Intermediate

If the **Max Inactivity** and **Inactivity Warning** settings are specified, APSExpire can send a user email before the user's account actually expires. For very important customers, mail could be sent to an administrator instead of (or in addition to) the user. This setting specifies the file(s) to send.

This mail is only sent by APSExpire and only if a user will expire and warnings are to be sent.

```
Inactivity Warning Mail=InactivityWarning.email
```

Inactivity Disabled Mail

Default: none

Recommended: Yes

Complexity Level: Advanced

If APSExpire actually disables a user for inactivity, this setting specifies the mail file(s) that it should send. This is a separate setting from the mail sent if user expiration is detected during the login process, but is often the same file.

```
Inactivity Disabled Mail=InactivityDisabled.email
```

Change Confirmation Mail

Default: none

Recommended: No

Complexity Level: Advanced

When a user changes their password, the change is confirmed on the screen. As an option, APS can also send mail to confirm the password change. This is generally unnecessary.

A special macro is available for this mail only. This macro contains the user's new password and may be inserted into the mail using the text %PASSWORD%. It is generally not a good idea to do this, however.

```
Change Confirmation Mail=ChangeConfirmation.email
```

Mail Settings

When Advanced Password Services needs to send mail, it must communicate with a Simple Mail Transfer Protocol (SMTP) mail server.

The original design of APS used Microsoft's MAPI to send mail. This had advantages and disadvantages. The biggest advantages included the fact that you could use your existing mail address book and that you could send mail internally when an SMTP gateway server did not exist. However, MAPI is limited to interfacing to Microsoft Exchange servers only when server programs are sending mail. Also, MAPI does not exist within the Unix environment.

Since an SMTP interface is used to send mail, addresses must be Internet mail addresses in the form of username@domain.com, rather than an internal mail system address. Multiple addresses may be the target of email, separated by commas. White space (spaces) is ignored.

To control the message sent for each event, create a file for the event, and identify that file (or files) in the APS configuration file using the settings detailed in the previous section. In APSMail parlance, these are called mail templates.

Email is sent to the SMTP server immediately. However, the SMTP server may not send the mail to the target address right away. Thus, if a delivery error occurs, such as the target address is invalid, APS is not notified (since it has long since moved on). To detect and process these cases, a site should either regularly check the SMTP error logs or use a Reply On Error address so that sending errors are returned to the sender

The template file format is described starting on page 287.

[Mail]

All of the Mail Settings appear in a special Mail section in the APS Configuration File. The section starts when the text [MAIL] (case does not matter) appears in the file and continues until the end of the file or another section starts. All of the previously described keywords do not appear in a section and therefore must appear in the file before any other section starts.

No keywords in this section support overrides.

Server

Default: none

Recommended: Yes

Complexity Level: Basic

This is the name or IP address of your SMTP gateway. APS will communicate with this server using port 25, which is assigned to SMTP. This can be either the name (such as MAIL.MYDOMAIN.COM) or the IP address (performance will be slightly faster using the IP address). You may need to obtain this information from your mail administrator.

If this field is blank or not supplied, APSMail will determine the applicable mail server using its own methods. If no server can be determined, no mail will be sent.

APSMail looks up the port number for the smtp service. If it is not explicitly configured for your machine, it will use port 25. To use a different port, configure the smtp service in your protocols file to the correct port number.

The SMTP server must support mail relaying.

```
Server=127.0.0.1
```

Log Path

Default: none

Recommended: During testing or for auditing purposes

Complexity Level: Intermediate

APSMail can log all mail files to a log file, along with the result returned by the SMTP server at the time that the mail was submitted. This setting specifies if and where such logging should occur. Just because a message is logged does not mean that it was sent. It is useful for testing where either a mail server is not available or mail should not really be sent, such as in a development environment that uses production data.

APSMail supports a number of alternate ways to set this value if it is not specified here. See the chapter entitled Using Email (see page 363) for complete details.

```
Log Path=APSMail.Log
```

Directory

Default: none

Recommended: Yes

Complexity Level: Advanced

The Directory setting is similar to the PATH environment variable; it can contain one or more directories on which mail files can reside.

Multiple directories are separated by semicolons on Windows NT, colons on Unix.

Directory names can contain replacement/lookup references. One of the easiest ways to internationalize email is to specify a directory like:

```
/mail/{preferredLanguage}
```

In this case, the current (LDAP) user's value for the attribute called preferredLanguage will replace the reference in the path.

```
Directory=/mail/{preferredLanguage}
Directory=/mail
```

APSMail supports a number of alternate ways to set this value if it is not specified here. See the chapter entitled Using Email (see page 363) for complete details.

Custom Logging

APS supports two different extension libraries, SmAPSEx and SmAPSLog. These libraries allow the extension of APS functionality without modifying the base APS modules.

SmAPSLog is a library that can be used to provide custom logging of APS activity. It might be useful, for example, to produce summarized statistics. The source code for the SmAPSLog library is supplied with the installation kit. However, this code is provided "as is", without warranty and without support.

The APS Configuration File can contain settings for SmAPSLog. Any such settings are automatically passed to SmAPSLog for use by that library.

[Logging]

All of the SmAPSLog settings appear in a special *Logging* section in the APS Configuration File. The section starts when the text [Logging] (case does not matter) appears in the file and continues until the end of the file or another section starts. All general keywords do not appear in a section and therefore must appear in the file before any sections start.

Custom logging settings appear after this keyword in the same type of format supported in the rest of the file.

No setting in this section supports overrides.

Custom Extensions

APS supports two different extension libraries, SmAPSEx and SmAPSLog. These libraries allow the extension of APS functionality without modifying the base APS modules.

SmAPSEx is a library that can be used to provide custom functionality for APS. It can be used to change the behavior of APS without modifying APS itself. However, SmAPSEx must be a secure module, since it can see passwords in clear text (so, for example, it can implement custom password content rules). Thus the library is not made directly available to developers outside of CA. Contact CA Professional Services for information if you think that you need a custom SmAPSEx module.

The APS Configuration File can contain settings for SmAPSEx. Any such settings are automatically passed to SmAPSEx for use by that library. See the documentation provided with your version of SmAPSEx to determine what the valid settings are if you have a copy of SmAPSEx.

[Custom]

All of the SmAPSEx settings appear in a special Custom section in the APS Configuration File. The section starts when the text [Custom] (case does not matter) appears in the file and continues until the end of the file or another section starts. All general keywords do not appear in a section and therefore must appear in the file before any sections start.

Custom logging settings appear after this keyword in the same type of format supported in the rest of the file

No custom setting support overrides.

Invalid Password Dictionary

Dictionary checking is used to prevent common words from being embedded in passwords. Administrators often put words in the dictionary that are common to the type of business of the site. For example, a bank may wish to disallow words like "Account", "Savings", "Checking", and "Money".

[Dictionary]

All of the dictionary words appear in a special Dictionary section in the APS Configuration File. The section starts when the text [Dictionary] (case does not matter) appears in the file and continues until the end of the file or another section starts. All general keywords do not appear in a section and therefore must appear in the file before any sections start.

All entries in this section are disallowed words (all comparisons are case-insensitive and checked both forwards and backwards). Comments and blank lines ARE allowed in this section. Note that words less than four (4) characters will be ignored.

Example:

```
[Dictionary]
        Customers
        Accounts
        Baseball
        Password
```

Password Complexity

APS can calculate a password's complexity and compare it to a specified threshold to determine if the password should be allowed.

There are three metrics that APS uses to evaluate password complexity:

- Character values

  Each character has a value. APS adds up the values assigned to each character. Characters that appear more than once (case-insensitive) do not count multiple times.

- Length

  Longer passwords are better. APS gives points for longer passwords.

- Case switches

  Mixed case passwords are better. Every time that the password "switches case" (goes from lower to upper or upper to lower), a value is added to the complexity.

APS has a utility called APSComplexity that calculates the complexity of a given password. It may be useful to try various passwords to get a feel for these calculations.

The *weight* of each metric can be adjusted in the complexity section of the APS.cfg file. This section begins with a single line in the configuration file that looks like:

It is very unusual for sites to make changes to this section. The defaults are suitable for most sites. This section exists so that a site can fine-tune the calculations still further.

[Complexity]

Normally, this section is not needed, since there are default values for all possible complexity settings. This section can be used to fine-tune the weightings used for these calculations.

Note that *no* settings in this section can be overridden by user.

Sorted

Default: not sorted

Recommended: Yes

Complexity Level: Advanced

When calculating the character values, APS ignores repeating characters. If this keyword appears, the characters will be sorted *before* this calculation. For example, without the **Sorted** keyword, "Passwords" gets credit for the letter "s" twice (the second appearance does not count, since it repeats the prior character). If **Sorted** appears, "Passwords" receives credit for the letter "s" only once.

To turn this setting off, comment it out of the configuration file.

```
Sorted
```

Case Switch

Default: 2

Recommended: use the default

Complexity Level: Advanced

When calculating complexity, APS will add this value to the complexity score each time that the case switches from lower to upper or upper to lower case. Thus, using the default value of 2, "Passwords" would receive 2 points for a case switch and "PassWords" would receive 6. Intervening non-alphabetic characters are ignored in these calculations, so "Pass2W4ords" still received 6 points.

If case switching should not be considered during complexity calculations, set this value to zero.

```
Case Switch=0
Case Switch=4
```

Length

Default: +2

Recommended: use the default

Complexity Level: Advanced

This is actually 29 different keywords: **Length4**, **Length5**, through **Length32**. Each value specifies the score applied to passwords of that length. If no length values are specified, length is scored at zero for length 4, plus two points for each additional character (length 5=2, length 7=6, etc.).

If only some length values are specified, the default ("+2") is used up to the first specified value, then the specified value is used for all lengths up to the next specified value and so on. For example, if the following is specified:

```
Length4=0
Length8=4
Length12=6
```

Would be the same as:

```
Length4=0
Length5=0
Length6=0
Length7=0
Length8=4
Length9=4
Length10=4
Length11=4
Length12=8
Length13=8
```

(and so on).

Character Values

Default: 0-10

Recommended: Yes

Complexity Level: Advanced

There are 256 characters in the ASCII character set. Every character can have its own score and these scores can be overridden in APS.cfg. The default scores are:

- Digits: 5
- Punctuation: 5
- Non-printable: 10

Letter scores are shown in the following table. Lower case characters, by default, have the same value as their upper-case equivalents.

A	1	J	8	S	1
B	3	K	5	T	1
C	3	L	1	U	1
D	2	M	3	V	4
E	1	N	1	W	4
F	4	O	1	X	8
G	2	P	3	Y	3
H	4	Q	10	Z	10
I	1	R	1		

There are three ways to set the score for a specific character. For displayable characters, either of the following methods will work. The examples below set the score for the capital letter "A":

```
A=5
'A'=5
```

Obviously, this mechanism does not work for non-displayable characters. Instead, the hexadecimal value of the character can be specified. For example, the score for Control-Z could be set to 5 using:

```
\x1A=5
```

If only one case of a character is explicitly set, then the other case will automatically be set to the same value.

Default scores for international (multibyte) characters are:

- Printable: 5
- Non-printable: 10

Field Mappings

APS supports Windows NT, LDAP and ODBC User Directories. For LDAP and ODBC User Directories, APS requires that certain attributes or columns be available in each user entry so that it can store operational information.

Internally, APS knows each of these attributes by a certain name (described in the chapter entitled User Directories: Schema, Storage and Capabilities (see page 189)). However, CA recognizes that, in some cases, these names cannot be used in a directory, either because of the implementation or because of column naming policies required by the site.

An additional problem is that some attributes/columns will have a different name in different directories. It is *highly desirable* to be able to reference only a single name when building mail templates, settings overrides and performing attribute replacement within the APS.cfg file.

This section, the [Mappings] section, is used to remap a "known" name of an attribute (or column) to the underlying name.

Every setting in this section can be overridden. However, the expressions that can be used are restricted in that they cannot reference any user information, such as attribute values or group membership (the IsInGroup function). Attempts to do so will be rejected.

Mapping is rarely required. If a "known" name does not exist (at all) within this section, APS will assume that the name in the underlying User Directory is the same as the "known" name.

Note: Certain remappings are required in the APS.cfg file to identify and process Microsoft Active Directories. The following two lines are the minimum required to support an Active Directory as a user store (in the [Mapping] section of the APS.cfg file):

```
userPassword={IsInDirectory("<dirname>")} unicodePwd
inetOrgPerson={IsInDirectory("<dirname>")} user
smapsPassword ={ IsInDirectory("<dirname>") }
```

Note: *<dirname>* is the name of the User Directory entry in your Policy Store.

Note: If the only User Directory is an Active Directory, then the following two lines can be used:

```
userPassword=unicodePwd
inetOrgPerson=user
smapsPassword=
```

Attribute Mapping

To map a standard attribute, use the format:

```
<logical-name>={<restricted-override>} <physical-name>
```

Where <*logical name*> is the name by which the attribute/column is know to APS by. <*physical-name*> is the underlying LDAP attribute or ODBC column name and <*restricted-expression*> is an override expression that is expressed in User Directory, not user or context, terms.

For example, under LDAP, users' full names are stored in an attribute called cn. Under ODBC, this is often a column called *FullName*. These can be mapped to a common name:

```
Name={IsLDAP()} cn
Name={IsODBC()} FullName0
```

Now, Name can be referenced in mail templates and override expressions without worrying about the differences between User Directories.

Attribute Suppression

Some of the information maintained by APS is purely informational; it exists simply for reporting purposes. Sites can suppress this information by mapping the "known" name to a blank name. The descriptions of the APS attributes starting on page 140 tell which of these attributes can be suppressed in this way.

Supressing attributes this way does not significantly improve write performance. APS groups all writes into a single server request. For writes, the performance hits are primarily in the network portion of the request, in the record (index) lookup, and in the fault tolerance (journaling) portion of the update. The difference between updating two attributes and four, for example, is normally not even measurable, as long as the updates are performed on the same request.

However, for very large directories, the savings in *disk space* may be significant enough to be desirable. For example, with 100 million users, not saving smapsPreviousLogin, which might average 24 characters, would save at least 2.4 GB of disk space.

Of course, the saving of this storage should be balanced against the loss of this information for reporting purposes.

```
smapsPreviousLogin=
```

Renaming LDAP Groups

This section must be used to rename LDAP groups so that they can be used by APSAdmin as described on page 157. APSAdmin cannot use full DN names in its references, it needs a single name. This section is used to map a single name to a full LDAP DN for those purposes.

```
FailureGrp=cn=FailureCount,o=Airius.com
```

This is *only* used by APSAdmin. You cannot change the name of a group that APS will use as a disable group.

LDAP Reverse Groups

If LDAP Reverse Groups are required (as described on page 164), they are defined in this section.

To have APS treat *all* LDAP groups as reverse groups, use:

```
LDAP.ReverseGroups=*
```

An asterisk should not be considered a wildcard. It can only be used in the above manner to indicate that all groups are reverse groups.

To specify a single group as a reverse group:

```
    LDAP.ReverseGroups=cn=Disabled-NoCredit, o=nds.com
```

Multiple such lines can exist, one for each reverse group.

ODBC Queries

Wherever possible, APS will use the queries defined in the SiteMinder ODBC Query Scheme associated with the User Directory. However, there are some additional queries that APS needs and there will be times that it is not appropriate for APS to use queries defined to SiteMinder.

In addition to the APS-specific queries, every query defined in the SiteMinder Query Scheme can be overridden for use by APS.

Each query has replacement parameters, or placeholders, embedded within them that indicate where APS (or SiteMinder) is to place values before using the query. Each query will replace these parameters with specific values in the order defined by the query. Thus, the first parameter for a given query may be replaced by the User's ID. It is not possible to change the order of the parameters (the choice of placeholders and their order is defined by SiteMinder's Query Schemes). Placeholders are indicated in a query by "%s".

[ODBC]

All of these queries are defined or overridden in the APS.cfg file in this section.

Replace Quote With

Default: ' + CHAR(39) + '

Recommended: not used

Complexity Level: Advanced

This keyword can be used to specify the exact sequence that single quotes (apostrophes) should be replaced with when embedded inside of other quotes in a SQL query. By default, APS will use

```
    ' + CHAR(39) + '
```

This is the standard supported by SQL. However, some SQL implementations may use other sequences (Microsoft Access, for example, wants CHR instead of CHAR).

Test Handle

Default: off

Recommended: If needed

Complexity Level: Advanced

This setting was created to work around a problem with the base release of SiteMinder 5 and only affects ODBC User Directories. If you are getting "Invalid Handle" errors in your Authentication Log, put this setting into your APS.cfg file. APS will then test any handle passed to it from SiteMinder. If the handle is invalid, APS will open its own handle to the ODBC directory.

Enumerate

Default: n/a

Recommended: not used

Complexity Level: Advanced

The Enumerate query overrides the query by the same name defined to SiteMinder. APS does not use this query at this time.

Get Object Info

Default: n/a

Recommended: not used

Complexity Level: Advanced

The Get Object Info query overrides the query by the same name defined to SiteMinder. APS does not use this query at this time.

Lookup

Default: n/a

Recommended: not used

Complexity Level: Advanced

The Lookup query overrides the query by the same name defined to SiteMinder. APS does not use this query at this time.

Init User

Default: n/a

Recommended: not used

Complexity Level: Advanced

The Init User query overrides the query by the same name // defined to SiteMinder. APS does not use this query at this time.

Authenticate User

Default: SELECT Name FROM SmUser

```
WHERE Name='%s' AND Password='%s'
```

Recommended: Defaults from SiteMinder, required

Complexity Level: Basic

The Authenticate User query overrides the query by the same name defined to SiteMinder. APS uses this query to determine if the old password entered during a password change is valid.

The first parameter is the user's name (entered to SiteMinder) and the second value is the (clear-text) old password as entered during the change password process.

This query can be a stored procedure, but the replacement parameters must retain their order and meaning. If the password is encrypted, then this query almost must be a stored procedure.

Get User Property

Default: SELECT %s FROM SmUser

```
WHERE Name='%s'
```

Recommended: Defaults from SiteMinder, required

Complexity Level: Basic

The Get User Property query overrides the query by the same name defined to SiteMinder. APS uses this query to retrieve the attribute values defined for the user.

The first parameter is always replaced with an asterisk (retrieve all defined columns), the second parameter is the user's id. If all columns should not be returned, this query should be overridden to reference a stored view in the database that returns fewer columns. Note that only columns returned on this query can be used in overrides or as macros in mail or redirections.

The first parameter is always replaced as a constant asterisk. A query could be defined with this, but then APS could not use the query defined to SiteMinder (which contains a replacement parameter in this position).

This query can be a stored procedure (if the underlying RDBMS supports rows returned from stored procedures), but the replacement parameters must retain their order and meaning.

Set User Property

Default: UPDATE SmUser

SET %s='%s'

WHERE Name='%s'

Recommended: Defaults from SiteMinder, required

Complexity Level: Intermediate

The Set User Property query overrides the query by the same name defined to SiteMinder. APS uses this query to set attribute values for the user.

The first parameter is the (mapped) name of the attribute (column) to modify, the second is the value to set it to. The third parameter is the user's name.

APS does special processing when using this query. If the query defined to APS (or SiteMinder) uses UPDATE, then APS will build a query to update all columns at once, using standard SQL syntax.

If, however, a stored procedure is used for this query, APS will call the stored procedure for each change. Parameters must appear in the same order for stored procedures.

The use of the UPDATE query is for higher performance.

If column access is to be restricted, create a stored VIEW in the database and use an UPDATE query to the stored view.

Get User Properties

Default: n/a

Recommended: not used

Complexity Level: Advanced

The Get User Properties query overrides the query by the same name defined to SiteMinder. APS does not use this query at this time.

User Properties

Default: n/a

Recommended: not used

Complexity Level: Advanced

The User Properties setting overrides the setting by the same name defined to SiteMinder. APS does not use this query at this time.

Lookup User

Default: SELECT Name, 'User' AS Class

```
FROM SmUser

WHERE %s
```

Recommended: Defaults from SiteMinder, required

Complexity Level: Basic

The Lookup User query overrides the query by the same name defined to SiteMinder. FPS and APSExpire use this query to locate users in the directory.

The parameter is the WHERE clause built up by FPS or APSExpire.

Get User Groups

Default: See Description

Recommended: Defaults from SiteMinder, required

Complexity Level: Basic

The Get User Groups query overrides the query by the same name defined to SiteMinder. APS uses it to return the list of group in which the current user is a member. The default query is:

```
SELECTSmGroup.Name
FROMSmGroup, SmUser, SmUserGroup
WHERE SmUser.Name='%s' AND
    SmUser.UserID=SmUserGroup.UserID
    AND
            SmGroup.GroupID=SmUserGroup.GroupID
```

The parameter is the user's name.

Each row returned represents a group name of which the user is a member.

Is Group Member

Default: See Description

Recommended: Defaults from SiteMinder, required

Complexity Level: Basic

The Is Group Member query overrides the query by the same name defined to SiteMinder. APS uses it to determine if the user is in a specific group, both for internal purposes and to determine the result of IsInGroup() calls in an override expression. The default query is:

```
SELECT ID FROMSmUserGroup
                WHERE UserID=
                        (SELECT UserID FROM SmUser
                        WHERE Name='%s')
                AND GroupID=
                        (SELECT GroupID FROM SmGroup
                        WHERE Name='%s')
```

The first parameter is the user's name, the second is the name of the group of interest.

If the result of the query contains any rows or data, the user is considered a member of the group.

Set Password

Default: See Description

Recommended: Defaults from SiteMinder, required

Complexity Level: Basic

The Set Password query overrides the query by the same name defined to SiteMinder. APS uses it to actually change the user's password. Typically, it is a stored procedure, but it need not be. The default query is:

```
    UPDATE SmUser SET Password='%s' WHERE Name='%s'
```

The first parameter is the new password, the second is the name of the user.

Note that the password may be encrypted, if SmAPSEx encrypted it.

This query can be a stored procedure (and should be), but the replacement parameters must retain their order and meaning.

Passwords will always be set using this query, never the Set User Properties query above.

Get Login History

Default: None

Recommended: Required if login history to be kept

Complexity Level: Intermediate

The Get Login History query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Login History is to be maintained.

Login History contains an entry for each login attempt by a user. Typically, it will be a separate table containing two fields, the history entry and the user's name.

The next three queries are also used to manipulate login history.

The query has one parameter that is the user's name.

The Get Login History query must return a single column with the login history entry. Its name is irrelevant. The entries should be returned in date order.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
SELECT smapsLoginHistory
                FROM LoginHistory
                WHERE Name='%s'
                ORDER BY smapsLoginHistory
```

Set Login History

Default: None

Recommended: Required if login history to be kept

Complexity Level: Intermediate

The Set Login History query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Login History is to be maintained.

Login History contains an entry for each login attempt by a user. Typically, it will be a separate table containing two fields, the history entry and the user's name.

The query has two parameters. The first is the Login History value and the second is the user's name.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
    INSERT INTO LoginHistory (smapsLoginHistory, Name) VALUES ('%s','%s')
```

Delete Login History

Default: None

Recommended: Required if login history to be kept

Complexity Level: Intermediate

Note: The Delete Login History query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Login History is to be maintained.

Login History contains an entry for each login attempt by a user. Typically, it will be a separate table containing two fields, the history entry and the user's name.

Delete Login History query is used by APS to clean out old history values (before a specific date and time).

The query has two parameters. The first is the date and time to delete before and the second is the user's name.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
    DELETE FROM LoginHistory WHERE LEFT(smapsLoginHistory, 15)<'%s' AND Name='%s'
```

Clear Login History

Default: None

Recommended: Required if login history to be kept

Complexity Level: Intermediate

The Clear Login History query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Login History is to be maintained.

Login History contains an entry for each login attempt by a user. Typically, it will be a separate table containing two fields, the history entry and the user's name.

The Clear Login History query is used by APSAdmin to clean out all of the login history for a specific user.

The query has a single parameter used to identify the user.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
    DELETE FROM LoginHistory WHERE Name='%s'
```

Get FPS Log

Default: None

Recommended: Required if FPS log to be kept

Complexity Level: Advanced

The Get FPS Log query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if FPS Logging is to be maintained.

The FPS Log contains an entry for each FPS usage attempt by a user. Typically, it will be a separate table containing two fields, the log entry and the user's name.

The next two queries defined are also used to manipulate the FPS Log.

The query has one parameter that is the user's name.

The Get FPS Log query must return a single column with the log entry. Its name is irrelevant. The entries should be returned in date order.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
SELECT smfpsLog FROM FPSHistory
                    WHERE Name='%s'
                    ORDER BY smfpsLog
```

Add FPS Log

Default: None

Recommended: Required if FPS log to be kept

Complexity Level: Advanced

The Add FPS Log query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if FPS Logging is to be maintained.

The FPS Log contains an entry for each FPS usage attempt by a user. Typically, it will be a separate table containing two fields, the log entry and the user's name.

The query has two parameters. The first is the FPS Log value and the second is the user's name.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
    INSERT INTO FPSHistory (smfpsLog, Name) VALUES ('%s','%s')
```

Clear FPS Log

Default: None

Recommended: Required if FPS log to be kept

Complexity Level: Advanced

The Clear FPS Log query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if FPS Logging is to be maintained.

The FPS Log contains an entry for each FPS usage attempt by a user. Typically, it will be a separate table containing two fields, the log entry and the user's name.

The query has a single parameter used to identify the user.The Clear FPS Log query is used by APSAdmin to clean out all of the FPS Log entries for a specific user.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
DELETE FROM FPSHistory WHERE Name='%s'
```

Set Password Checksum

Default: None

Recommended: Required if Auto Force Change used

Complexity Level: Advanced

Note: The Set Password Checksum query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Auto Force Change functionality is required.

The Auto Force Change keyword in this file tells APS to check for password changes outside of APS and, if detected, force the user to change their password at next login. It is used to automatically treat external (administrative) password changes as Force Immediate Change situations without requiring changes to the administration utility.

Under LDAP, APS uses the smapsPassword attribute to handle this functionality. Under ODBC, this is not necessarily possible. Databases are typically protected so that passwords cannot be read back.

If Auto Force Change is to be used, this and the following query must be defined. They are almost always stored procedures.

The query has a single parameter used to identify the user (the user's password has already been changed).

Typically, the implementation of this functionality uses some special attribute to store the checksum (or the entire password value, for that matter).

An example query might be:

```
CALL SetPasswordChecksum('%s')
```

Test Password Checksum

Default: None

Recommended: Required if Auto Force Change used

Complexity Level: Advanced

Note: The Test Password Checksum query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Auto Force Change functionality is required.

The Auto Force Change keyword in this file tells APS to check for password changes outside of APS and, if detected, force the user to change their password at next login. It is used to automatically treat external (administrative) password changes as Force Immediate Change situations without requiring changes to the administration utility.

Under LDAP, APS uses the smapsPassword attribute to handle this functionality. Under ODBC, this is not necessarily possible. Databases are typically protected so that passwords cannot be read back.

If Auto Force Change is to be used, this and the previous query must be defined. They are almost always stored procedures.

The query has a single parameter used to identify the user. The function should return a numeric or boolean value, where non-zero indicates that the checksum is invalid.

Typically, the implementation of this functionality uses some special attribute to store the checksum (or the entire password value, for that matter).

An example query might be:

```
?=CALL TestPasswordChecksum('%s')
```

Compare FPS Answer

Default: None

Recommended: Required if FPS Answers are encrypted

Complexity Level: Advanced

Note: The Compare FPS Answer query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if FPSí ODBC Encrypt functionality is required.

The answers to FPS questions can be encrypted in an ODBC database. This is indicated to APS using the ODBC Encrypt keyword in the [FPS-Verify] section of the APS.cfg file. This query is one way that sites can implement this encryption.

There is no default for this query. If not defined and ODBC Encrypt was specified in APS.cfg, FPS will generate an error, since it won't know how to compare an encrypted answer (this is assuming that the encryption is not being performed by SmAPSEx).

The query must be a stored procedure that takes three arguments (or, at least, three substitution parameters) and return a numeric or boolean value (non-zero indicates that the compare is true).

The first parameter is the user name. The second parameter is the name of the attribute, as configured to APS. The third parameter is the user-entered answer (in clear text).

The implementation of this function usually encrypts (or hashes) the user supplied value (the third parameter) and compares it to the value stored in the user entry.

An example query might be:

```
?=CALL CompareFPSAnswer('%s', '%s', '%s')
```

Add To Group

Default: None

Recommended: Required if groups maintained by APSAdmin

Complexity Level: Advanced

Note: The Add To Group query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if the APSAdmin API is to be used to maintain group memberships.

The APSAdmin API functions use this query to add users to an existing group (creation of groups is not supported).

There is no default query for this purpose. If not specified and a user must be added to a group, an error will be logged and the update will fail.

An example query might be:

```
INSERT INTO SmUserGroup (UserID, GroupID) VALUES ('%s', '%s')
```

Remove From Group

Default: None

Recommended: Required if groups maintained by APSAdmin

Complexity Level: Advanced

Note: The Remove From Group query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if the APSAdmin API is to be used to maintain group memberships.

The APSAdmin API functions use this query to remove users from an existing group.

There is no default query for this purpose. If not specified and a user must be removed from a group, an error will be logged and the update will fail.

An example query might be:

DELETE FROM SmUserGroup

WHERE UserID='%s' AND

GroupID='%s'APSAdmin

Admin Translation

Default: None

Recommended: Required in certain APSAdmin scenarios.

Complexity Level: Advanced

Note: The Admin Translation query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if the APSAdmin API is to be used in certain scenarios.

This query takes a single parameter that is the value entered by the administrator on the APSAdmin user selection form. The query is expected to return a single record with at least one column that is the user's name (the one used in all of the other queries). If not specified, then the query is not used and the data entered is expected to be the user's name. If multiple records or no records are returned, a "User record could not be found" error is displayed to the administrator. If multiple records are returned, an error is issued to the console log as well.

This query is typically used if the administrator is expected to identify users to APSAdmin using something other than userid, such as membership number.

An example query might be:

```
SELECT Name FROM Users WHERE MemberID='%s'
```

APSExpire

The **[**APSExpire**]** section of the APS configuration file controls the operation of the APSExpire utility, described in the chapter entitled Daily Processing (APSExpire) (see page 297).

In the APS Configuration File, a site defines *jobs* by name. Each setting is the name of a job and the values define the criteria for the job. When APSExpire executes, a job name must be specified on its command line. The program will look for the definition of this job in this section of the configuration file.

Each job defines a user directory or subset of a user directory. The syntax is different for ODBC and LDAP directories.

Overrides are not supported in this section.

LDAP Directories

For LDAP directories, jobs are defined using this syntax:

```
<job name>= <LDAP directory>
                          READ(<ip>)
                          BASE(<base DN>)
                          SCOPE(<scope>)
                          FILTER(<filter>)
```

<job name> is an arbitrary name for the job.

<ip of LDAP directory> is the ip address, the network name, or the SiteMinder User Directory name of an LDAP directory defined to SiteMinder through the Policy Interface (it cannot contain spaces if used here). If it is an ip address, it may contain the port address as well. This must match up with the definition of a User Directory in the SiteMinder Policy Store (APSExpire will attempt to look up the directory using this value).

READ(<*ip*>) is an optional clause that tells APSExpire to read from a different directory than the base directory. In some cases, much higher performance can be achieved by reading from a dedicated replicant directory that either SiteMinder does not use at all or is the last directory in a failover chain. If specified, however, the alternate directory must be a replicant of the "real" directory.

BASE(<*search Base*>) defines the scope of the search. It is entire optional. If not specified, APSExpire will search the entire directory using the search base defined in the SiteMinder User directory entry. This is useful when an entire LDAP directory is not to be processed as a single job. Sites do this when the LDAP directory is very large and APSExpire processing is to be spread over multiple servers or jobs.

SCOPE(<*scope*>) is optional and is generally used with the BASE option above. <*scope*> can either be "base" or "sub" (without quotes). It specifies how the LDAP search should be processed.

FILTER(<*extra filter*>) is another optional setting that allows a site to further refine a job. This filter is ANDed with any filters that APSExpire uses for its own operations. Once again, this is intended to segregate an LDAP directory into smaller jobs for performance reasons.

When using BASE, SCOPE and FILTER, it is the responsibility of the site to make sure that every user will be processed. APSExpire does not examine the sum of all defined jobs to ensure that all users get processed.

ODBC Directories

For ODBC directories, jobs are defined using this syntax:

<job name>= *<ODBC directory>*

WHERE(<extra WHERE clause>)

<job name> is an arbitrary name for the job.

<*ODBC directory*> is the DSN name or the SiteMinder User Directory name of an ODBC user directory (neither can have embedded spaces in this context) defined to SiteMinder through the Policy Interface. This must match up with the definition of a User Directory in the SiteMinder Policy Store (APSExpire will attempt to look up the directory using this value).

WHERE(<*extra WHERE clause*>) is another optional setting that allows a site to further refine a job. This clause is ANDed with any WHERE clause that APSExpire uses for its own operations. This is intended to segregate an ODBC directory into smaller jobs for performance purposes.

When using WHERE, it is the responsibility of the site to make sure that every user will be processed. APSExpire does not examine the sum of all defined jobs to ensure that all users will be processed.

General FPS Settings

All FPS settings appear in the APS configuration file after the keyword [FPS]:

Audit Log

Value: File Path

Default: off

Recommended: yes

Complexity Level: Basic

FPS can log all attempts, successful or failed, to an audit log. This log is written to a flat file in comma-delimited format (suitable for import into many database and spreadsheet applications).

To specify the location of this log file, use this setting.

There is no way to control the format or content of this log, nor is there provision for wrapping or deleting the file. If this setting does not appear in the configuration file, no audit log will be written. Please be sure that the user under which the SiteMinder Policy Server processes are running can create and write to this file.

This file is not terribly useful. A site should check its contents to determine if the information is worth keeping.

```
Audit Log=/usr/Netegrity/SiteMinder/Logs/FPS.log
```

Directory

Value: Server name or ip address for LDAP,

SN name for ODBC

Default: 127.0.0.1:389

Recommended: required

Complexity Level: Basic

This keyword tells FPS which directories to search when FPS is invoked.

Each instance specifies one or more SiteMinder user directories.

After the list of directories, an optional condition can be defined that tells FPS when that particular set of directories should be searched. This condition is surrounded by square brackets ("[" and "]") and can contain a partial URL (or *stem*) of the Forgot stub This URL must contain the port number, if other than 80, and cannot contain the "http:" or "https:" prefix. FPS will use the full URL of the FORGOT stub that invoked the process and, if the specified stem is included entirely, that directory or set of directories will be used.

If multiple lines exist with the Directory keyword and apply, all lines will be used.

If the same directory appears in more than one line, it will only be searched once.

Directories are specified as the ip address and port of an LDAP directory server or servers OR the DSN name of an ODBC directory.

If a nonstandard LDAP port (not 389) is used, it must be appended to each ip address in the list that it applies to, separated from the ip address by a colon.

```
Directory=127.0.0.1
Directory=DSN_CNA DSN_SCA [//www.acme-calif.com]
Directory=DSN_TX [//www.acme-texas.com]
```

If the user requesting FPS came in from www.acme-calif.com, FPS will search the local LDAP directory (127.0.0.1) and the ODBC directories with the DSN's of DSN_NCA and DSN_SCA.

If the user arrived from www.acme-texas.com, then the LDAP directory and DSN_TX will be searched.

The directories will be searched in the order that they appear.

FPS looks into the list of User Directories stored in the SiteMinder Policy Store, looking for an entry that references the server identified by this setting. FPS will use the first entry that matches to obtain administrator credentials and the search base (for LDAP directories).

Some sites will define multiple User Directory entries that reference the same physical LDAP server, each User Directory defining a different search base (or, in rare situations, credentials). This can confuse FPS, causing it to read the wrong portion of these LDAP directories.

To get around this problem, a site should trick SiteMinder (and FPS) into thinking that the multiple User Directory entries are implemented on *separate* servers. This is easily done using the hosts file on the Policy Server (it can be done using DNS as well, though that is a little more complicated) to define multiple names for the same physical IP address. Each User Directory should then reference a different alias for the same physical LDAP server. In the APS.cfg file, this setting, Directory, can then uniquely identify the proper User Directories.

Allowed

Value: Standard override expression

Recommended: If required

Complexity Level: Basic

The Allowed keyword is used to define which users in a directory are allowed to use FPS. If no Allowed keyword exists in this section, all users in the directory will be allowed to use FPS. If a single instance of this keyword exists, then rights must be explicitly granted to this functionality.

The Allowed keyword may appear as many times as necessary.

```
Allowed=true
```

FPS LDAP Settings

Under APS 3.0, there was a section called [LDAP] that was used to define the LDAP directory to be processed by FPS. This entire section has been eliminated from the APS Configuration File in APS Version 4.0.

The **Server** keyword has been replaced with the **Directory** keyword in the prior section. If APS finds the **Server** keyword, its value will be used as though presented in the **[FPS]** section using the **Directory** keyword and a warning will be issued.

The Administrator, Password, Search Base, ObjectClass and Lockout Group DN keywords are ignored. A warning is issued if these settings are encountered. APS now gets this information from the SiteMinder User Directory definition.

The Disallowed keyword is no longer supported. If encountered, it will be processed as though it appeared in the [FPS] section with the Allowed keyword, with a prefixed NOT operator and a warning will be issued.

The Allowed keyword is no longer supported in this section. If encountered, it will be processed as though it appeared in the [FPS] section and a warning will be issued.

FPS Identify Process

The first thing that FPS must do when invoked by a user is to attempt to identify the user. All of the information that FPS needs to do this is defined in a section headed by a single line in the FPS configuration file containing the text:

[FPS-Identify]

Everything appearing after this line and before the start of another section is considered part of the identification section.

Prior to APS Version 4.0, this section was called [Identify]. In Version 4, this was renamed to [FPS-Identify]. The old name is still recognized and will be processed correctly, but a warning will be issued about the use of the deprecated name.

This section specifies the form(s) required to identify the user, how the forms are to be used, and how to handle various typical error conditions.

Collecting Identification Information

URL

Value: URL

Default: none

Recommended: required

Code Description: URL (see page 307)

Complexity Level: Basic

When FPS is first invoked by a user, this is the page that FPS will first cause to be displayed. It is expected to present a form to the user. It must be a page that is *not* protected by SiteMinder. It need not be on the same directory as the Forgot stub.

```
URL=/FPS/Identify.jsp
```

Missing URL

Value: URL

Default: the value of URL above

Recommended: yes

Code Description: Missing URL (see page 308)

Complexity Level: Intermediate

If the user fails to fill in all of the required fields or one or more of the fields contain an invalid character, FPS will redirect the user to this URL. The URL will have a list of this missing or invalid fields appended to it, separated with ampersands ("&"). A question mark will be appended before the field names, if required.

There is currently no way to tell if the fields in error were empty but required, or if they contained invalid data.

```
Missing URL=/FPS/Identify-Missing.jsp
```

Multiple URL

Value: URL

Default: the value of Missing URL above, with ?MoreData appended

Recommended: yes

Code Description: Multiple URL (see page 308)

Complexity Level: Intermediate

If the user filled out the form presented by the URL, but multiple users were found matching the input criteria, this page is displayed. Depending upon your site, one of two cases may be in effect, in which case, this form must handle the correct case.

First, if the required fields on the identify form must absolutely define a single user, then you have a data or login error in your flow and this page must handle the error case.

If, on the other hand, your identify form has optional fields on it, this page might just request further information. For example, if your identification form requires first name and last name, but mail is optional, there may be more than one "Bill Smith" on file. This page could then request that one or more of the optional fields be supplied to further refine the search.

```
Multiple URL=/FPS/Identify.jsp?MoreData
```

Error Cases

Not Found URL

Value: URL

Default: the value of Missing URL above, with ?NotFound appended

Recommended: yes

Code Description: Not Found URL (see page 309)

Complexity Level: Intermediate

If *no* user matches the input criteria, this page will be presented to the user. This is a terminating case (the FPS process is done, since no user can be identified).

If not supplied, the value of Missing URL is used, with ?NotFound appended.

```
Not Found URL=/FPS/Unknown.jsp
```

Disabled URL

Value: URL

Default: none

Recommended: yes

Code Description: Disabled URL (see page 309)

Complexity Level: Basic

If the user is properly identified, but it turns out that the user is disabled for any reason, the user will be redirected to this page. This is a terminating case, since the FPS process cannot continue.

Note that this case is different, but related to, the Lockout case. Typically, this page refers the user to a customer server center.

```
Disabled URL=/FPS/Disabled.jsp
```

Disabled Mail

Value: Mail file(s)

Default: none

Recommended: if your site has email addresses on file

Complexity Level: Advanced

If the user is properly identified, but it turns out that the user is disabled for any reason, FPS can send mail. This setting specifies the files to send. If possible, mail should be sent to the actual user, since the user accessing FPS might not be the owner of the account.

This setting is optional and has no default.

```
Disabled Mail=Disabled.email
```

Form Processing

These settings describe the identification form, its fields, and its requirements. FPS uses these settings to ensure that the entered data is valid.

It may seem possible to perform these edits using JavaScript. In fact, it is highly recommended that a site do so to reduce round trips to the server. However, JavaScript processing can be turned off by some browsers and can be modified by a savvy user, so FPS performs these edits as well.

Required

Value: HTML field names

Default: none

Recommended: required

Complexity Level: Basic

This setting contains a list of the HTML fields on the identify (and missing and multiple) forms that are required. If any field identified by this setting is posted without a value, FPS will redirect the user to the page identified by the Missing URL setting above.

Names are not case sensitive, may appear in any order and are separated by semicolons. Multiple fields with the same name are not supported.

An FPS configuration error will occur if a field is posted that is neither required nor optional.

```
Required=FirstName;LastName
```

Optional

Value: HTML field names

Default: none

Recommended: if needed

Complexity Level: Intermediate

Some fields on the identification forms can be optional; that is, the user need not provide a value. Usually, these fields are only used to further refine the user identification process (see the example under the description for the Multiple URL setting).

This setting contains a list of HTML fields that are optional.

Names are not case sensitive, may appear in any order and are separated by semicolons. Multiple fields with the same name are not supported.

An FPS configuration error will occur if a field is posted that is neither required nor optional.

If the Submit button has a name (is renamed), then its value will be posted. In this case, its name must be defined as an Optional field.

```
Optional=Phone;City;State;UserID;phonenow;Mail
```

Lookup

Value: special (see text)

Default: none

Recommended: required

Complexity Level: Basic

Once the identify form is posted and all of the fields validated (required fields not blank and no fields contain invalid characters), this setting is used to determine how to look up the user.

This setting consists of HTML Field/attribute mappings, separated by either an equal sign ("=") or a equal sign/tilde ("=~"). Each pair is separated from the others by a semicolon. Names are not case-sensitive, but the order that they appear can have a significant performance impact.

A pair is only used if there is non-blank input for the HTML field.

For each combination, FPS builds a search string for the user, comparing the attribute/column with the value supplied on the form. If an equal sign is used, then the value must compare exactly (but may be subject to the comparison rules of the underlying Directory). If the tilde ("~") is used, then a case-insensitive match is performed.

Basically, you want the most specific items first in the list, with less specific items appearing later.

Multiple attributes can be specified for the same HTML field, separated from each other by commas. This indicates that the value of the field could be in *any* of the supplied attributes. This is typically used for phone numbers, where the supplied number could be in the telephone*,* homePhone*,* or mobile attributes. FPS will create an OR condition in this case that will be ANDed with all other clauses.

Not all fields need to appear in the **Lookup** setting, but a field in this setting that is neither **Required** nor **Optional** will never be used for lookup. A field that does not appear in this setting could be used in mail, for example (in some examples, the user can enter a "Phone number where you can be reached right now" that could be sent as mail to a customer service desk in the event of an error case).

```
Lookup=UserID=uid;FirstName=givenname;
                         LastName=sn;Mail=mail;
                         Phone=telephoneNumber,homePhone;
                         City=~l;State=st
```

Controlling Frequency of Use

Max Success Frequency

Value: number of days

Default: none

Recommended: 7

Complexity Level: Advanced

Some sites may wish to limit the use of FPS to prevent misuse. Some users may use FPS instead of their password, if the password content policies make memorable passwords too difficult. This setting can be used to prevent a user from successfully using FPS too frequently.

This setting specifies, in days, how much time must elapse after a successful use of FPS before a user can use FPS again.

For example, if the user uses FPS today to recover his password, he cannot use FPS again for a week if this value is seven. If he attempt to do so, he will be redirected to the Too Recently Used URL below.

For this feature to work with ODBC correctly, the FPS Logging queries (starting on page 107) must be defined.

```
Max Success Frequency=7
```

Too Recently Used URL

Value: URL

Default: none

Recommended: required if Max Success Frequency is set above

Code Description: Too Recently Used URL (see page 310)

Complexity Level: Advanced

If a user attempts to use FPS too soon after he has already recovered a password (based on the Max Success Frequency above), the user will be sent to this page.

For this feature to work with ODBC correctly, the FPS Logging queries (starting on page 107) must be defined.

```
Too Recently Used URL=/FPS/TRU.jsp
```

Too Recently Used Mail

Value: mail file(s)

Default: none

Recommended: ignored unless Max Success Frequency is set above

Complexity Level: Advanced

If the user is sent to the Too Recently Used URL above, any mail files specified in this setting are also sent, if possible.

For this feature to work with ODBC correctly, the FPS Logging queries (starting on page 107) must be defined.

```
Too Recently Used Mail=TRU.email
```

Max Attempts Frequency

Value: number

Default: none

Recommended: 1

Complexity Level: Advanced

FPS is a weak link in your site's security. A user's account can be attacked and entered without using a password using FPS. If a site uses questions to verify the user, the answers to those questions can often be socially engineered or obtained from other sources.

In order to improve security against such a hacker attack, this setting can control how often APS can be used by an account. It specifies, in days, how much time must elapse after any use of FPS (success or failure) before a user can use FPS again.

For example, if Jane Doe uses FPS today to recover John Smith's password today, but either fails to answer the question correctly or leaves the process without answering the question, nobody can use FPS against John Smith's account again for a day if this value is one. If someone attempts to do so, they will be redirected to the Too Recently Attempted URL below.

For this feature to work with ODBC correctly, the FPS Logging queries (starting on page 107) must be defined.

```
Max Attempts Frequency=1
```

Too Recently Attempted URL

Value: URL

Default: none

Recommended: required if Max Attempts Frequency is set above

Code Description: Too Recently Attempted URL (see page 310)

Complexity Level: Advanced

If a user attempts to use FPS too soon after any attempt to recover a password (based on the Max Attempts Frequency above), the user will be sent to this page.

For this feature to work with ODBC correctly, the FPS Logging queries (starting on page 107) must be defined.

```
Too Recently Attempted URL=/FPS/TRA.jsp
```

Too Recently Attempted Mail

Value: mail file(s)

Default: none

Recommended: ignored unless Max Attempts Frequency is set

Complexity Level: Advanced

If the user is sent to the Too Recently Attempted URL above, any mail files specified in this setting are also sent, if possible.

For this feature to work with ODBC correctly, the FPS Logging queries (starting on page 107) must be defined.

```
Too Recently Attempted Mail=TRA.email
```

Preventing Attacks

Lockout Count

Value: number

Default: none

Recommended: 3

Complexity Level: Intermediate

Adept hackers will attempt to access your system using FPS. Even if your site uses validation questions, a hacker can gain access by socially engineering the answers to questions.

This setting essentially sets a "3 strikes, you're out" policy on FPS. This setting identifies the number of failed attempts to use FPS that occur before FPS is locked out for that account.

Each time that a user fails to complete an FPS session, this counter is incremented by one. Once the count reaches Lockout Count, FPS is no longer available for this account.

This number is only reset upon successful completion of an FPS session. Note that this means that if the account is locked out, there is no way for the user to reset this value. The only way for this value to be reset is for an administrator to clear smfpsLockoutCounter using a User Administration tool such as APSAdmin or CA Identity Manager.

This setting can automatically be reset upon successful SiteMinder authentication if the Reset FPS (see page 99) Lockout setting is turned on.

```
Lockout Count=3
```

Lockout URL

Value: URL

Default: none

Recommended: 3

Code Description: Lockout URL (see page 310)

Complexity Level: Intermediate

If the user is locked out due to the Lockout Count, the user will be redirected to this page. The user will always be redirected to this page the first time that the count is reached. Further attempts to use FPS will also redirect the user to this page.

```
Lockout URL=/FPS/Lockout.jsp
```

Lockout Mail

Value: mail file(s)

Default: none

Recommended: if possible

Complexity Level: Advanced

If the user is locked out of FPS when attempting to recover a password, the files specified by this setting will be sent as email, if possible.

This is useful to detect intruders, in that the mail could be sent to an administrator account.

It is also useful to send mail to the user, if possible, since the user will not see the Lockout URL if the account is disabled by a hacker.

```
Lockout Mail=Lockout.email
```

Lockout Timer

Value: 0 to 365 * 24 * 60

Default: 0

Recommended: if possible

Complexity Level: Advanced

If the user is locked out of FPS when attempting to recover a password, FPS normally locks out the user permanently. This setting specifies a delay to use instead. The user will not be allowed to use FPS for this many minutes.

```
Lockout Timer=60
```

FPS Verify Process

At the end of the Identify phase, FPS knows the identity of the user. Some sites may wish to attempt to verify that the user is who he claims to be (this process is highly recommended). If this is desired, the settings in the FPS-Verify section should be used.

All of the information that FPS needs to do this is defined in a section headed by a single line in the FPS configuration file containing the text:

[FPS-Verify]

Everything appearing after this line and before the start of another section is considered part of the verify section.

Prior to APS Version 4.0, this section was called [Verify]. In Version 4, this was renamed to [FPS-Verify]. The old name is still recognized and will be processed correctly, but a warning will be issued about the use of the deprecated name.

This section specifies the form(s) required to verify the user, how the forms are to be used and how to handle various common error conditions.

Forms

These settings define the forms that FPS should use to query the user for verification information.

URL

Value: URL

Default: none

Recommended: yes

Code Description: URL (see page 311)

Complexity Level: Intermediate

If verification is to be used, this setting must be specified. This setting tells FPS where to send the user for verification. It is a form.

```
URL=/FPS/Verify.jsp
```

Missing URL

Value: URL

Default: the value of URL above

Recommended: yes

Code Description: Missing URL (see page 312)

Complexity Level: Intermediate

If the user fails to fill in all of the required fields or one or more of the fields contain an invalid character, FPS will redirect the user to this URL. The URL will have a list of this missing or invalid fields appended to it, separated with ampersands ("&"). A question mark will be appended before the field names, if required.

There is currently no way to tell if the fields in error were empty, but required, or if they contained invalid data.

```
URL=/FPS/Verify.jsp
```

Retry URL

Value: URL

Default: none

Recommended: no

Code Description: Retry URL (see page 313)

Complexity Level: Intermediate

If the user fails to answer the questions properly, he can be sent to one of two places. If this setting is not specified, the user will be sent to the page identified by the Invalid URL setting below, otherwise this setting will be used.

If set, the user will be sent to the Retry URL to re-attempt verification. Depending on the values in the Initial setting, the initial values may or may not differ. It is up to the site to control the number of times that retries can occur. Usually, this is controlled using the Restrict or Consume special instructions so that initial values are not reused and eventually run out (sending the user to the No Data URL below).

The use of this keyword is in no way affected by, nor affects, the Lockout Count settings, since Lockout Count only affects entry to the entire process, not looping within the process.

The use of this keyword significantly reduces the security of FPS, so we do not recommend its use.

```
Retry URL=/FPS/RetryVerify.jsp
```

Error Conditions

Invalid URL

Value: URL

Default: the value of URL above

Recommended: yes

Code Description: Invalid URL (see page 313)

Complexity Level: Intermediate

If the user answers the verification wrong, FPS will direct the user to this page. This is a terminal condition; the FPS process is terminated.

```
Invalid URL=/FPS/Unverified.jsp
```

Invalid Mail

Value: mail file(s)

Default: none

Recommended: yes

Complexity Level: Advanced

If the user answers the verification wrong, FPS will send these files, if possible, via email.

```
Invalid Mail=NoVerify.email
```

No Data URL

Value: URL

Default: none

Recommended: yes, if required

Code Description: No Data URL (see page 314)

Complexity Level: Intermediate

The Initial setting (described below) specifies the initial field settings that the URL is to receive before processing. In the Initial setting, some values may be marked *required*. If so, and no value (or not enough values) can be determined, the user will be sent to this page instead. This is a terminal condition (FPS processing terminates - the user cannot use FPS).

Typically, this is a page directing the user to a customer service representative.

```
No Data URL=/FPS/NoData.jsp
```

No Data Mail

Value: mail file(s)

Default: none

Recommended: yes, if required

Complexity Level: Advanced

If the user will be redirected to the No Data URL above, the file(s) specified by this setting can also be sent via email.

```
No Data Mail=NoData.email
```

Timeout URL

Value: URL

Default: none

Recommended: yes

Code Description: Timeout URL (see page 314)

Complexity Level: Intermediate

This setting specifies a page to send the user if the user failed to answer the questions within the period specified by the Timeout setting ("You did not answer quickly enough"). This reduces the ability of a hacker to socially engineer the answers to the verification question(s).

This is a terminal condition (FPS processing terminates - the user cannot use FPS at this time).

```
Timeout URL=/FPS/Timeout.jsp
```

Form Handling

These settings describe the verify form, its fields, and its requirements. FPS uses these settings to ensure that the entered data is valid.

It may seem possible to perform these edits using JavaScript. In fact, it is highly recommended that a site do so to reduce round trips to the server. However, JavaScript processing can be turned off by some browsers and can be modified by a savvy user, so FPS performs these edits as well.

Required

Value: HTML field names

Default: none

Recommended: required

Complexity Level: Basic

This setting contains a list of the HTML fields on the verify (and retry) forms that are required. If any field is posted without a value, FPS will redirect the user to the page identified by Missing URL setting above.

Names are not case sensitive, may appear in any order and are separated by semicolons. Multiple fields with the same name are only supported using special instructions (see the Initial setting below). If the special instruction field requires multiple values, the correct number of values must be specified by the user.

An FPS configuration error will occur if a field is posted that is neither required nor optional.

```
Required=SecretAnswer
```

Optional

Value: HTML field names

Default: none

Recommended: if needed

Complexity Level: Intermediate

Some fields on the verify form can be optional; that is, the user need not provide a value. For example, if the Submit button has a name (is renamed), then its value will be posted. In this case, its name must be defined as an Optional field.

This setting contains a list of HTML fields that are optional.

Names are not case sensitive, may appear in any order and are separated by semicolons.

An FPS configuration error will occur if a field is posted that is neither required nor optional.

```
Optional=SubmitButton
```

Lookup

Value: special (see text)

Defalt: none

Recommended: required

Complexity Level: Advanced

Once the verify form is posted and all of the fields validated (required fields not blank and no fields contain invalid characters), this setting is used to determine how to look up the user.

This setting consists of HTML Field/attribute mappings, separated by either an equal sign ("=") or an equal sign/tilde ("=~"). Each pair is separated from the others by a semicolon. Names are not case-sensitive, but the order that they appear can have a significant performance impact.

A pair is only used if there is non-blank input for the HTML field.

For each combination, FPS builds an evaluation expression for the user, comparing the attribute with the value supplied on the form. If an equal sign is used, then the value must compare exactly (in this case, the comparison defined by the underlying User Directory is irrelevant, FPS will require the exact match). If the tilde ("~") is used, then a case-insensitive compare is performed.

All such expression are combined with an AND operator, then checked against the user previously identified.

Multiple attributes can be specified for the same HTML field, separated from each other by commas. This indicates that the value of the field could be in any of the supplied LDAP attributes. This is typically used for phone numbers, where the supplied number could be in the telephone*,* homePhone*,* or mobile attributes. FPS will create an OR condition in this case that will be ANDed with all other clauses.

Not all fields need to appear in the Lookup setting, but a field in this setting that is neither Required nor Optional will never be used for lookup. A field that does not appear in this setting could be used in mail, for example (in some examples, the user can enter a "Phone number where you can be reached right now" that could be sent as mail to a customer service desk in the event of an error case).

If a field has special instructions (see the Initial setting below) that require multiple values, the query will be built appropriately (varies by special instruction).

```
Lookup=SecretAnswer=~personalAnswerLookup=SecretAnswer=QuestionList
```

Initial

Value: special (see text)

Deault: none

Recommended: as needed

Complexity Level: Intermediate

Usually, the verify page will need data from the user's record for display (see the chapter entitled *Presentation Forms* for details as to how this information is passed to the form). This setting controls what information is passed to the form. This is not required if the page does not need information about the user.

Sometimes, this is as simple as putting the identified user's name on the form. Other times, displaying the verification question selected by the user is required.

The format of this setting is as name/value pairs, separated by an equal sign ("="). Multiple pairs are separated by semicolons.

The name in each pair is the name that the page uses to identify the data element. It need not correspond to an HTML element. It is used inside the data cookie to name the field.

The second part of the pair identifies the name of the attribute from which FPS is to read the data value. Multiple values are not supported, except as defined by a *Special Instruction*.

If the name is prefaced by an asterisk, then the field *must* have a value. If, after looking up the attribute, the field does not have a value, the user will be redirected to the No Data URL.

```
Initial=*Name=cn;*Question=secretQuestion
```

Special Instructions

After the attribute name, *special instructions* can be specified for the field. Special instructions are surrounded by square brackets ("[" and "]") and *immediately* follow the attribute name.

Special instructions define special processing or formatting required for the field.

Special instructions consist of keywords separated by commas. Every special instruction requires a Format keyword. Other keywords vary by format. At the time of this writing, only Format=A and Format=B are supported. Additional formats may be supported in the future.

- **Format=A**

  Format A requires that the underlying attribute be multi-valued. Since only LDAP directories support multi-valued attributes, Format A is only supported on LDAP User Directories. However, Format B, which is virtually identical to Format A, is supported on ODBC directories.

  Format A indicates that the attribute is stored in the format:
  `<control data>|<question id>|<answer>`

  The following items exist as instances of the attribute:

  - <control data> can be blank, a date (in the format YYYYMMDDhhmm), the word DISABLED or the word DELETED. Any other value is treated as though it were DISABLED.
  - <question id> is the text of a question (it must not contain a pipe character ("|"), an asterisk ("*"), or a caret ("^")), or it may contain a question identifier. FPS does not care which.
  - <answer> is the answer to the corresponding question, in clear text. It *may* not contain a pipe ("|"), but must not contain an asterisk ("*") or a caret ("^").

  When FPS returns the value(s) from the attribute, it only returns the <question id>. When it compares the returned answer, it compares the combination of the <question id> with the supplied answer.

  Every time the question is asked, the <control data> is updated with the current date and time.

  Only questions that have never been asked (those with no <control data>) and valid dates in <control data> can be selected. Keywords can be specified to further control which questions can be asked.

  When using Format A, do not use approximate matching (the tilde sign). Since the question id is included in the match, almost anything that the user enters will come up as a match.

Pick=<number>

The Pick keyword tells FPS to select more than one value from the list of values stored in the attribute. If this keyword is not specified, only one value will be selected from the list.

If multiple values are specified, they will be supplied in separated by the pipe character ("|").

Note that if the name is prefixed by an asterisk and not enough values can be selected to satisfy the requirement, no values will be sent to the page. If the No Data URL is specified, the user will be sent to that page.

Questions are first qualified for selection using the <control data> and the keywords below. Once a qualified list has been produced, the questions are selected at random.

Restricted

If specified, this keyword indicates that no question can be selected if it has been used since the user last logged in. That is, if the date in <control data> is more recent than the last login date, the question cannot be selected. The use of this keyword requires that APS be installed and operational (since the last login date is tracked by APS).

Consume=<number>

This keyword indicates that a question cannot be selected if it has been asked within the last <*number*> days.

Consume

This keyword, which is mutually exclusive of the **Consume=<*number*>** option above, indicates that a question cannot be selected if it has ever been selected (<control data> is not blank).

Sorted

Once the possible values have been qualified (those items not available for selection have been removed from an in-memory list), the list is sorted. Those items that have already been selected today retain their full date/time for sorting. Those items selected more than 30 days ago have their date/time removed and the rest are sorted only by date (the time is removed). Please note that the adjustments are made internally for the purposes of sorting and are not saved to the directory.

The items are sorted in increasing order (oldest to newest). Those items without a date are sorted first.

FPS then looks down the list until it finds enough to satisfy the **Pick** requirement. If there is a "tie" on the effective date after reading enough items, FPS continues to count down the list until the "tie" is broken. From this list, FPS randomly picks the requisite number of items.

- **Format=B**

  Format B is identical in every way to Format A, with one exception: Instead of storing each of question as a separate value of a multi-valued attribute, all question strings are stored in a single instance of the attribute, separated by a caret ("^"). In all other respects, Format=B is identical to Format=A.

  ```
  Initial= *SecretQuestion=QuestionList [format=A,Pick=2,sorted]
  ```

ODBC Encrypt

Value: Attribute list, comma separated

Default: none

Recommended: as required

Complexity Level: Advanced

This setting specifies one or more attributes that must be encrypted or hashed before comparison. FPS will call the Compare FPS Answer (see page 147) query, if that query is defined.

If the Compare FPS Answer query is not defined, it will attempt to call a stored procedure on the User Directory called ODBCEncrypt_<attribute> with two parameters, the user name and the value for the attribute entered by the user. The procedure is expected to return a single Boolean (or integer) result. A non-zero return indicates that the value compared successfully.

```
ODBC Encrypt=SecretAnswer
```

Timer

Value: Number

Default: none

Recommended: 300

Complexity Level: Intermediate

This setting controls how much time (in seconds) is allowed to elapse before the user submits the verify form. If the user waits too long, the user will be redirected to the Timeout URL when the page is finally posted.

```
Timeout=300
```

FPS Change Password Process

Once the user is verified, you site may allow the user to set their password or may want to generate random passwords. If you wish to allow users to select their own passwords, specify this information in a section starting with:

[FPS-Change]

Everything appearing after this line and before the start of another section is considered part of the change password section.

Prior to APS Version 4.0, this section was called [Change]. In Version 4, this was renamed to [FPS-Change]. The old name is still recognized and will be processed correctly, but a warning will be issued about the use of the deprecated name.

This section specifies the form(s) required to gather a password change from the user, how the forms are to be used and how to handle various common error conditions.

URL

Value: URL

Default: none

Recommended: yes

Code Description: URL (see page 315)

Complexity Level: Basic

If the user will be allowed to select a password, this setting must be specified. This setting tells FPS where to send the user for the input form. It is a form.

```
URL=/FPS/ChangePassword.jsp
```

Timeout URL

Value: URL

Default: none

Recommended: yes

Code Description: Timeout URL (see page 315)

Complexity Level: Intermediate

This setting specifies a page to send the user to if the user fails to submit the form within the time period specified by the **Timeout** setting ("You did not answer quickly enough").

This is a terminal condition (FPS processing terminates - the user cannot use FPS at this time).

```
Timeout URL=/FPS/Timeout.jsp
```

Timeout

Value: Number

Default: none

Recommended: 300

Complexity Level: Intermediate

This setting controls how much time (in seconds) is allowed to elapse before the user submits the form. If the user waits too long, the user will be redirected to the Timeout URL when the page is finally posted.

```
Timeout=300
```

FPS Confirm Process

At the end of the FPS process, after the user has been identified and perhaps verified, FPS must give the user the information desired. This may include a new password and sometimes includes the user ID.

All of the information that FPS needs to do this is defined in a section headed by a single line in the FPS configuration file containing the text:

[FPS-Confirm]

Everything appearing after this line and before the start of another section is considered part of the confirm section.

Prior to APS Version 4.0, this section was called [Confirm]. In Version 4, this was renamed to [FPS-Confirm]. The old name is still recognized and will be processed correctly, but a warning will be issued about the use of the deprecated name.

This section specifies the form(s) required to confirm information to the user, how the forms are to be used and how to handle various common error conditions.

URL

Value: URL

Default: none

Recommended: yes, if required

Code Description: URL (see page 315)

Complexity Level: Basic

Under normal circumstances, this is the URL of a page to use to confirm the FPS process. Any data identified by the Initial setting (below) will be passed to this URL on its query string.

Some sites may consider this a security hole, so if this value is prefixed by an asterisk, FPS will display its own (internal) form for confirmation and will instead redirect the user to this URL upon completion. If this is the case, no query string will be used (since FPS can build the page dynamically).

If a password and user ID are to be recovered, only one should be displayed on this page (the other should be sent via mail), since both together open a larger security hole.

```
URL=/FPS/Confirm.jsp
URL=*/HomePage.jsp
```

Mail

Value: mail file(s)

Default: none

Recommended: yes, if required

Complexity Level: Advanced

At the completion of the FPS process, one or more files can be sent, via email, using this setting.

If the user will be redirected to the No Data URL above, the file(s) specified by this setting can also be sent via email.

If both a password and user id are to be recovered, only one should be sent via mail (the other should be displayed on a page), since both together opens a larger security hole.

There are several special macros available to this mail.

Macro Name	**Purpose**
Password	Clear text password that was randomly generated or that the user selected.
HalfPassword1	The first half of the new password, in clear text. Useful for mailing half and displaying half.
HalfPassword2	The second half of the new password, in clear text. Useful for mailing half and displaying half.
OneShotPassword	Only generated if the macro is requested, this is a random, 32-character password that can be used within 5 minutes (not-configurable) of generation to log this user in ONCE. Useful to automatically log in the user. Requires the APS Authentication Scheme to be installed.

```
Mail=Confirm.email
```

Initial

Value: special (see text)

Default: none

Recommended: as needed

Complexity Level: Intermediate

The confirm page needs the information that it will display (usually the password and/or uid). This setting identifies the information that should be passed to the confirm page.

The format of this setting is as name/value pairs, separated by an equal sign ("="). Multiple pairs are separated by semicolons.

The name in each pair is the name that the page uses to identify the data element. It need not correspond to an HTML element. It is used in the query string to name the field.

The second part of the pair identifies the name of the attribute from which FPS is to read the data value. Multiple values are not supported. You cannot use userPassword, as this is a hashed field. Use password instead.

All of the macros defined in the table under the Mail keyword are available as additional attributes.

```
Initial=User=uid;PWD=password
```

Force Change

Default: none

Recommended: yes

Complexity Level: Intermediate

When FPS sets the user's password, it can optionally set the force change password flag in the user's directory entry. FPS will only do this if this setting appears in the FPS configuration file.

```
    Force Change
```

New Password Length

Value: -32

Default: 8

Complexity Level: Intermediate

At the completion of the process, FPS can reset the user's password. This setting controls the length of the new password. If specified out of the valid range, a length of 8 will be used.

If the user is allowed to change her own password (as described in the [FPS-Change] section), this setting has no effect.

```
    New Password Length=10
```

Timeout

Value: 0 or 60-3000 seconds

Default: none

Recommended: 90 seconds

Complexity Level: Intermediate

If non-zero, FPS will set the **Must Login By** date and time to the current time plus this value. If the user does not login to your site within this period, the user will not be allowed to login.

```
    Timeout=90
```

Allowed Characters

Range: Character list

Default: none

Complexity Level: Advanced

The Allowed Characters keyword specifies a list of characters that are used in a generated password. Only characters listed with this keyword are used in generated passwords, subject to restriction by the Disallowed Characters and Force Case settings.

Each instance of this keyword can specify a list of characters. They may or may not be surrounded by double quotes. Since leading and trailing blanks in a setting value are ignored, these quotes may be necessary. If the value is surrounded by quotes, they will be removed from the list of allowed characters (though any contained quotes will be retained).

Multiple instances of this keyword may exist and may apply. APS will use the characters listed with *every* applicable instance of this setting.

If *no* Allowed Characters keyword is valid, then all characters will be used (subject to the Disallowed Characters setting below).

APS does not use characters that are both allowed and disallowed (they will be disallowed).

For example,

```
Allowed Characters=abcdefABCDEF01234
```

Disallowed Characters

Range: Character list

Default: none

Recommended: none

Complexity Level: Advanced

The Disallowed Characters keyword specifies a list of characters that are *not* allowed in a generated password. Characters listed with this keyword are not used in generated passwords.

Each instance of this keyword can specify a list of characters. They may or may not be surrounded by double quotes. Since leading and trailing blanks in a setting value are ignored, these quotes may be necessary. If the value is surrounded by quotes, they will be removed from the list of allowed characters (though any contained quotes will be retained).

Multiple instances of this keyword may exist and may apply. APS uses the characters listed with *every* applicable instance of this setting.

If *no* Disallowed Characters keyword is valid, then all characters are allowed (subject to the Allowed Characters setting above).

APS does not use characters that are both allowed and disallowed (they are disallowed).

```
Disallowed Characters=xyzXYZ56789
```

Force Case

Range: upper, lower, or none (default)

Default: none

Recommended: none

Complexity Level: Advanced

Controls whether alphabetic characters in generated passwords must be upper or lower case.

Default is "none" (characters may be either upper or lower case).

If Force Case is set to "upper" then a non-zero value for the Minimum Lower Case keyword cannot be satisfied. If Force Case is set to "lower" then a non-zero value for the Minimum Upper Case keyword cannot be satisfied.

For example:

```
Force Case=none
```

Minimum Length

Range: 0 to 32

Default: 4

Complexity Level: Intermediate

This setting determines the minimum length of the generated password. If specified out of the valid range, a length of 4 is used.

For example:

```
Minimum Length=8
```

Maximum Length

Range: 0 to 32

Default: 32

Complexity Level: Intermediate

This setting determines the maximum length of the generated password. If specified out of the valid range, a length of 32 is used.

For example:

```
Maximum Length=10
```

Minimum Length

Range: 0 to 32

Default: 0

Complexity Level: Intermediate

This setting determines the minimum number of upper case characters in a generated password.

Note: Any upper case character generated also contributes toward the required number of characters that are defined in the Minimum Letters and Minimum Alphanumeric keywords.

For example:

```
Minimum Upper Case=1
```

Minimum Lower Case

Range: 0 to 32

Default: 0

Complexity Level: Intermediate

This setting determines the minimum number of lower case characters in a generated password.

Note: Any lower case character generated also contributes toward the required number of characters that are defined in the Minimum Letters and Minimum Alphanumeric keywords.

For example:

```
Minimum Lower Case=4
```

Minimum Letters

Range: 0-32 characters

Default: 0

Complexity Level: Intermediate

This setting requires that the generated password contain a certain minimum number of alphabetic letters. Alphabetic characters are defined as the letters in the alphabet, regardless of case.

For example:

```
Minimum Letters=2
```

Minimum Digits

Range: 0-32 characters

Default: 0

Complexity Level: Intermediate

This setting requires that the generated password contain a minimum number of numeric digits ("0" to "9").

For example:

```
Minimum Digits=1
```

Minimum Alphanumeric

Range: 0-32 characters

Default: 0

Complexity Level: Intermediate

This setting specifies that a generated password contains a certain minimum number of alphanumeric characters ("A"-"Z" or "0"-"9").

Note: If this setting is used together with one of the Minimum Letters or Minimum Numbers settings, characters can satisfy both requirements. For example, if Minimum Digits is 4 and this setting is 4, the password 1234 satisfies both requirements.

For example:

```
Minimum Alphanumeric=1
```

Minimum Punctuation

Range: 0-32 characters

Default: 0

Complexity Level: Intermediate

This setting specifies that a generated password contain a certain minimum number of punctuation marks. These can be periods, commas, exclamation marks, and so on.

Note: If this setting is used together with the Minimum Other setting, punctuation characters satisfy both requirements.

For example:

```
Minimum Punctuation=1
```

Minimum Symbols

Range: 0-32 characters

Default: 0

Complexity Level: Intermediate

This setting specifies that a generated password contain a certain minimum number of symbol characters. Symbols are defined within APS as the following characters and all extended ASCII characters, including diacritical marks:

"~" (tilde)	"@" (at)	"#" (number)
"$" (dollar)	"%" (percent)	"^" (circumflex)
"&" (ampersand)	"*" (asterisk)	"(" (open parenthesis)
")"(close parenthesis)	"_" (underscore)	"-" (hyphen)

"+" (plus)	"=" (equals)	"{" (open brace)
"}" (close brace)	"[" (open bracket)	"]" (close bracket)
"<" (less than)	">" (greater than)	"/" (virgule)
"\" (back slash)	"|" (vertical bar)	

Note: If this setting is used together with the Minimum Other setting, symbol characters satisfy both requirements.

For example:

```
Minimum Symbols=1
```

Minimum Other

Range: 0-32 characters

Default: 0

Complexity Level: Intermediate

This setting specifies that a generated password contains a specified minimum number of non-alphanumeric characters. This includes punctuation marks and other symbols located on the keyboard.

For example:

```
Minimum Other=1
```

Maximum Repeat

Range: 0-32 characters

Default: 0

Complexity Level: Basic

This setting specifies maximum number of identical characters that can appear consecutively in a generated password. For example, if this setting is four, then aaaa should not appear anywhere in the password.

Note: This setting is advisory; the password generation algorithm makes every effort to satisfy this limitation but might not be able to, depending on the other settings. For example, if Maximum Repeat is set to 2, the password "A2bbc9j" would satisfy this guidance but "A2bbb9j" would not.

For example,

```
Maximum Repeat=3
```

FPS Errors

FPS may encounter an unexpected error during processing. This may be a missing configuration setting (one of the ìexpected errorî pages is not configured), or it may be an error which occurred when communicating with the User Directory. This section defines how such errors are to be reported to the user. Note that all errors arealso written to the SiteMinder Authentication Console Log. In most cases, the Console Log will contain more detailed information about the error that is not useful to the user (or should not be revealed to the user).

All of the information that FPS needs to do this is defined in a section headed by a single line in the FPS configuration file containing the text:

[FPS-Errors]

Everything appearing after this line and before the start of another section is considered part of the errors section.

Prior to APS Version 4.0, this section was called [Errors]. In Version 4, this was renamed to [FPS-Errors]. The old name is still recognized and will be processed correctly, but a warning will be issued about the use of the deprecated name.

URL

Value: URL

Default: none

Recommended: highly

Code Description: URL (see page 315)

Complexity Level: Basic

This is the page used to display errors. The error text will be in the query string (URL encoded).

```
URL=/FPS/Errors.jsp
```

Mail

Value: mail file(s)

Default: none

Recommended: highly

Complexity Level: Advanced

When such an error occurs, this setting indicates the file(s) to send via email. The actual error message can be included in the text using the message macro. Typically, this is used to notify an administrator that an error has occurred.

```
Mail=Errors.email
```

Handling FPS Mail Errors

This section defines how FPS is to handle the case where a mail file is specified but cannot be sent. This does not occur if the mail cannot be delivered. It only occurs if the mail file(s) cannot be found or if the mail server refuses to accept the message.

All of the information that FPS needs to do this is defined in a section headed by a single line in the FPS configuration file containing the text:

[FPS-No Mail]

Everything appearing after this line and before the start of another section is considered part of this section.

Prior to APS Version 4.0, this section was called [No Mail]. In Version 4, this was renamed to [FPS-No Mail]. The old name is still recognized and will be processed correctly, but a warning will be issued about the use of the deprecated name.

URL

Value: URL

Default: none

Recommended: highly

Code Description: URL

Complexity Level: Basic

This is the page used to display mailing errors. The error text will be in the query string (URL encoded).

```
URL=/FPS/NoMail.jsp
```

Mail

Value: mail file(s)

Default: none

Recommended: highly

Complexity Level: Advanced

When such an error occurs, this setting indicates the file(s) to send via email. The actual error message can be included in the text using the message macro. Typically, this is used to notify an administrator that an error has occurred.

There is no guarantee that this mail can be sent either.

```
Mail=NoMail.email
```

Chapter 4: User Directories: Schema, Storage and Capabilities

This chapter details how APS interfaces with each type of User Directory. This includes information about how APS stores its information within the directory, what "native" capabilities of the directory are supported (if any) and what APS features are not supported by specific User Directories.

General schema information is presented first, followed by details for each User Directory type.

This section contains the following topics:

Schema & Storage (see page 189)
Forgotten Password Services (FPS) Data (see page 208)
LDAP Directories (see page 211)
Microsoft Active Directories (see page 223)
Microsoft Windows Domains (see page 233)
ODBC (RDBMS) Directories (see page 237)
Password Replication/Synchronization (see page 252)

Schema & Storage

This section lists every attribute maintained and used directly by APS. APS assumes that all attributes exist and processing will fail if an attribute does not exist. Except for encrypted fields and those otherwise noted, your site can change the values of these attributes. However, you must take care to format the information in the attribute correctly so that APS can read it. For performance reasons, APS has little format verification.

Note that the fields used to handle FPS verification (such as secret questions) are not maintained by APS at all; they are *compared* by FPS and, in some cases updated (as in the control data). However, the field names and contents are entirely defined by the site and actually maintained by the site's user management tools, not APS.

The attributes listed in this section are in alphabetical order.

Date Format

All dates and times are in Greenwich (ZULU) time zone. This eliminates all complications of multiple policy/web servers in different time zones and daylight savings time. These values are stored in the format:

```
yyyymmddhhmmssZ
```

For example,

```
20010307164130Z
```

is Wednesday, March 7, 2001 at 11:41:30am Eastern time.

Attribute/Column Names

The Attribute/Column names used in this chapter can be changed on a per-site (or even per-directory server) basis. The names used here are the names that APS will use internally. If not remapped, APS will also use these names externally. To rename an attribute/column, use the [MAPPING] (see page 133) section of the APS.cfg file.

Some fields are *suppressible*, meaning that an entry can exist in the [MAPPING] section to map the field to a null name. In this case, APS will not store or use a value for that field. If an attribute is *not* suppressible, but a mapping to a null value exists in APS.cfg, the internal name of the attribute will be used. This will probably result in an error.

Note: Between APS Version 3 and Version 4, one attribute (smapsInactivityWarning) was dropped from use and a new attribute (smapsNextAction) was added. smapsOldBlob is also no longer used, except for sites upgrading from versions of APS prior to version 3.0.

smapsAccountInactivityDays

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.2
RDBMS Type:	character
Max Length:	24 (see note)
suppressible:	Yes

Format: *<integer> <reason>*

Examples:	0 Admin account 7 Wire Room Op 365

This attribute is used to override, on a per-*user* basis, the amount of time that can elapse between this user's logins before the user is disabled due to inactivity. As a general rule, this value should not be used; it is intended for a small number of users.

The first part of the value must be the number of days to use. The rest of the field is ignored and can be used to store any information (for example, *why* the override is there, who put it there, etc.).

Setting this value to zero tells APS that this user is never disabled due to inactivity, even if this conflicts with settings in the APS.cfg file.

If this field is null or contains no value, then the settings in the APS.cfg file will be used.

APS *never* writes to this field, except as part of APSAdmin.

Note on length: This field contains a number, followed by a comment, thus the length is truly variable. APSAdmin allows 3 characters for the integer value, a space, and up to 20 characters of user-entered comment. If your site has a custom interface to this field, you may want to make this field longer. APS itself, outside of APSAdmin, never updates this field.

smapsBaseDate

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.3
RDBMS Type:	character
Max Length:	36 (see note)
suppressible:	No

Format: *<date/time> <reason>*

Examples:	20010307164130Z Conversion

This is, essentially, the creation date of the user. It is used as the base date for all calculations if smapsLastPasswordChange or smapsLastLogin are not set or if this date is *later*. If not set, the current date/time is used. APSExpire will initialize this field, if necessary.

User creation utilities should set this value, but it is not required, since APSExpire will set it the next time that utility is run and APS will initialize a null value when the user authenticates.

If this field is later than smapsLastPasswordChange or smapsLastLogin, this field's value will be used instead. The reason for this is best demonstrated by an example: smapsLastLogin is used to calculate when an account expires. If a user's account has expired and the user is disabled, the user will immediately be expired again at next login, since smapsLastLogin is still too old. The old solution was to reset smapsLastLogin, but that caused the "real" date of last login to be lost. Instead, sites may reset this date instead in these cases (and in the similar case of password expiration).

Note on length: This field contains a date, followed by a comment, thus the length is truly variable. APSAdmin allows 15 characters for the date value, a space, and up to 20 characters of user-entered comment. If your site has a custom interface to this field, you may want to make this field longer. APS only writes the date/time (without a comment) to this field when it is initialized.

smapsDisableAfter

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.4
RDBMS Type:	character
Max Length:	36 (see note)
suppressible:	Yes

Format: *<date/time> <reason>*

Examples:	20010601000000Z End of semester
	20010601000000Z End of subscription

APS will not allow users to login after this date/time, regardless of activity. APS will not cancel an existing session when this time arrives, it will only prevent authentication after the specified date/time.

Sites may use this field freely. APS will never set or modify this value, but it will honor it if the date is readable. The *<reason>* is not used by APS, but may be passed to event redirection pages as the value of the DISABLEDREASON macro.

This field may need to be cleared to enable a user to authenticate.

Note on length: This field contains a date, followed by a comment, thus the length is truly variable. APSAdmin allows 15 characters for the date value, a space, and up to 20 characters of user-entered comment. If your site has a custom interface to this field, you may want to make this field longer.

smapsDisableUntil

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.5
RDBMS Type:	character
Max Length:	36 (see note)
suppressible:	Yes
Format:	*<date/time> <reason>*
	FOREVER *<reason>*
Examples:	20010207164130Z Failure Count
	FOREVER Failure Count

APS will not allow users to login until this date/time, regardless of whether the user successfully authenticates. APS uses this field to implement the Auto Reset Failure Count functionality.

If the Use Internal Disables is in effect for LDAP directories, APS will use this field to disable users and will use the word FOREVER instead of a date. For ODBC (RDBMS) directories, APS will always use this field for this purpose (regardless of the Use Internal Disables setting).

Sites may use this field freely. The reason is not used by APS, but may be passed to event redirection pages as the value of the DISABLEDREASON macro.

This field may need to be cleared to enable a user to authenticate.

Note on length: This field contains a date, followed by a comment, thus the length is truly variable. APSAdmin allows 15 characters for the date value, a space, and up to 20 characters of user-entered comment. If your site has a custom interface to this field, you may want to make this field longer. APS can write to this field, but never more than 36 characters.

smapsExpirePasswordDays

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.6
RDBMS Type:	character

Max Length:	24 (see note)
suppressible:	Yes

Format: *<integer> <reason>*

Examples:	0 Administrator
	7 Wire Room Op
	365

This attribute is used to override, on a per-user basis, the expiration period of a user's password. The performance impact of this value is nominal. As a general rule, it should not be used, since it creates maintenance overhead and thus can be an administrative nightmare.

The first part of the value must be the number of days to use. The rest of the field is ignored and can be used to store any information (for example, *why* the override is there, who put it there, etc.).

If the integer value is set to zero, then the user's password will never expire.

If this field is null or contains no value, then the settings in the APS.cfg file will be used.

APS *never* writes to this field, except as part of APSAdmin.

Note on length: This field contains a number, followed by a comment, thus the length is truly variable. APSAdmin allows 3 characters for the integer value, a space, and up to 20 characters of user-entered comment. If your site has a custom interface to this field, you may want to make this field longer. APS never writes to this field.

smapsFailureCount

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.7
RDBMS Type:	character
Max Length:	24
suppressible:	No

Format: *<integer> <date/time>*

Examples:	0 20010307170000Z
	2 20010307173022Z

This attribute is used by APS to track the current authentication failure count. When reset, this field should *not* cleared; it must be set to zero with a date and time. This is required because APS also keeps this value in memory so that a server outage will not open a security hole. APS will read the value from disk and compare the date/time against its in-memory value. The later value will be used.

The first part of the value is the counter, which is followed by the effective date of the most recent failure (or reset).

Note that just setting a value into this field is insufficient to disable a user (at the next login), since if the date/time is over Failure Count Timeout minutes old, the user will not be disabled.

This field may need to be updated to enable a user account.

smapsFailuresSinceLastLogin

LDAP Type:	cis/Single Values
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.8
RDBMS Type:	character
Max Length:	56 (see note)
suppressible:	Yes

Format: *<integer> <date/time> <IP address> <reason>*

Examples:	2 20010307170000Z 192.158.7.10 SiteMinder

This attribute is maintained by APS and is informational only. The *<integer>* component is the number of failed logins since smapsLastLogin (or smapsBaseDate, if the user has never logged in).

The *<date/time>* is when the most recent failure actually occurred.

The *<IP address>* is the reported tcp/IP address of the client. Note that this value is not trustworthy due to network address translation and spoofing.

The *<reason>* is why the login was rejected (if known).

APS copies the current value of this attribute to smapsFailuresSincePreviousLogin when the user successfully authenticates and then clears this attribute.

This value may be significantly different than smapsFailureCount, since the failure count times out and only includes password rejections. This is the actual total and includes all types of failures.

Note that if your site wishes to display failure count information on a user's screen, you should use smapsFailuresSincePreviousLogin, since the user just logged in and this value has been cleared.

Note on length: This field contains a number, followed by a comment, thus the length is truly variable. APSAdmin allows 3 characters for the integer value, a space, 15 characters for the date, another space, 15 characters for the IP address, yet another space and up to 20 characters of comment. If your site has a custom interface to this field, you may want to make this field longer.

smapsFailuresSincePreviousLogin

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.9
RDBMS Type:	character
Max Length:	56 (see note)
suppressible:	Yes

Format: *<integer> <date/time> <IP address> <reason>*

Examples:	2 20010307170000Z 192.158.7.10 SiteMinder

This attribute is maintained by APS and is informational only. This information is copied from smapsFailuresSinceLastLogin when the user successfully authenticates.

Note that if your site wishes to display failure count information on a user's screen, you should use smapsFailuresSincePreviousLogin, since the user *just* logged in and the other value has been cleared.

Note on length: This field contains a number, followed by a comment, thus the length is truly variable. APSAdmin allows 3 characters for the integer value, a space, 15 characters for the date, another space, 15 characters for the IP address, yet another space and up to 20 characters of comment. If your site has a custom interface to this field, you may want to make this field longer.

smapsGenerationalRedirects

LDAP Type:	cis/Multi-Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.10
RDBMS Type:	character
Max Length:	As long as possible (see note)
suppressible:	Yes

Format: *<key>=<integer> <date> <IP address>*

Examples:	LICENSE =1 20010307144130Z 10.2.2.1
	PROFILE=2 20010307144633Z 10.2.2.1

This attribute is used to store information about Generational Redirect (see page 119). If generational redirects are not used, this attribute can be omitted from the schema.

Each element (this is a multi-valued attribute under LDAP) tracks the last version of a page that the user was redirected to, when and what the IP address of the browser was (the IP address is not reliable).

Note on length: Under ODBC (RDBMS) directories, this is stored as a single-value, each "element" separated from others using a semicolon (";"). The length of each element can vary, since the length of the *<key>* name is essentially unlimited and the length of the *<integer>* value can vary greatly. The total length of this field can be calculated by a site, but if new generational redirects are added, additional space will have to be provided. Thus, a site should plan this length carefully.

smapsGraceLoginsUsed

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.11
RDBMS Type:	character
Max Length:	35
suppressible:	Yes

Format: *<integer> <date/time> <IP address>*

Examples:	2 20010307170000Z 192.158.7.10

APS uses this attribute to track the number of grace logins that the user has consumed since his/her password expired. The *<date/time>* and *<IP address>* are informational only. If the user changes their password, this value is reset to blank/null.

If grace logins are not used, then this attribute can be safely omitted from the schema.

smapsHistory

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.12
RDBMS Type:	character
Max Length:	25KB
suppressible:	Yes

Format: *<encrypted>*

The user's password history is stored in this attribute. APS will maintain this value. This value should never be modified, except by APS.

APS will only keep about 12KB worth of history, but limit is imposed *before* encryption. Encryption effectively doubles the length, so a site should provide storage for up to 25KB characters of data.

Since the information is encrypted, it is impossible to just truncate it; it must be stored in its entirety. Truncating the data may cause server failures.

To reserve the full amount of storage in an ODBC directory for every user row may be unreasonable. What many sites do is set the length of this column to some smaller value (for example 2k), then run *triggers* within the database to alert administrators when a certain percentage of this space starts to get used (say 90% - the field grows very slowly, usually less than 100 bytes per password change). If a single test user starts to consume too much, approaching the maximum length defined, the site clears the value *for that one account*. If multiple users start to approach the value, either determined by triggers or by periodic examination of the data, the column is enlarged.

smapsImmediateChange

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.13
RDBMS Type:	character
Max Length:	30 (see note)

suppressible:	Yes

Format: *<any non-blank value>*

Examples:	Initial Load

If this attribute is non-blank, APS will force the user to change their password the next time that the user logs in (if redirection, etc., is also configured).

APS clears/nulls this attribute when the user actually changes their password.

If sites do not wish to use the immediate change functionality, this attribute may be safely omitted from the schema.

Note on length: When set, this field contains any text comment, thus the length is truly variable. APSAdmin allows up to 30 characters of comment. If your site has a custom interface to this field, you may want to make this field longer.

smapsInactivityWarning

This field is no longer used by APS and it may safely be removed from existing schemas.

smapsLastLogin

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.15
RDBMS Type:	character
Max Length:	32
suppressible:	No

Format: *<date/time> <IP address>*

Examples:	20010307175245Z 192.168.42.10

This attribute holds the most recent login date and time (and IP address, if available, though it is not reliable).

Sites should not modify this value. It is used for inactivity calculations.

This field is required for proper APS operation.

smapsLastPasswordChange

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.16
RDBMS Type:	character
Max Length:	128 (see note)
suppressible:	No

Format: *<date/time> <comment>*

Examples:	20010307175245Z APS Interface

This attribute holds the date and time that the user's password last changed, if known. APS will update this field when the password is changed using any APS interface. It is up to the site to update this field if the password is changed in any other way.

This value is used for password expiration calculations and is required for APS operation.

Note on length: This field contains a date, followed by a comment, thus the length is truly variable. APSAdmin can set this to 15 characters for the date, a space, and the full length of the administrator's DN plus 12 characters. Thus, this field needs to accommodate the largest DN of your site. If your site has a custom interface to this field, you may want to make this field even longer.

smapsLoginHistory

LDAP Type:	cis/Multi-valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.17
RDBMS Type:	special, see description
suppressible:	Yes

Format: *<date/time> <result> <IP address> <comment>*

Examples:	20010307175245Z SUCCESS 10.2.3.2
	20010307175245Z FAILED 10.2.3.2 SiteMinder

APS uses this attribute to track all authentication activity for this user since the smapsPreviousLogin date. This includes both successes and failures.

This data is informational only. APS does not use it for anything.

In ODBC (RDBMS) directories, this information is kept in a separate table. This table should contain 2 columns: one to store the user's id and another to store the log entry. The names of the fields are discussed in the section on ODBC queries later in this chapter. The maximum length of the Log Entry column is about 60 characters (15 for date, a space, 7 for status, a space, 15 for the IP address, another space, and a comment).

If this field is suppressed, no authentication history will be kept. To suppress its use, be sure to specify this attribute in the **[MAPPINGS]** section of the APS.cfg file.

smapsMaxFailures

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.18
RDBMS Type:	character
Max Length:	60 (see note)
suppressible:	Yes

Format: *<integer> <date/time> <IP address> <reason>*

Examples:	12 20010307170000Z 192.158.7.10 SiteMinder

This attribute is maintained by APS and is informational only. This information is *identical* to the information maintained in smapsFailuresSinceLastLogin except that it contains the *highest* value that the account ever reached.

If this field is suppressed, no such information will be kept. To suppress its use, be sure to specify this attribute in the [MAPPINGS] section of the APS.cfg file.

Note on length: This field contains a number, followed by a comment, thus the length is truly variable. APSAdmin allows 3 characters for the integer value, a space, 15 characters for the date, another space, 15 characters for the IP address, yet another space and up to 20 characters of comment. If your site has a custom interface to this field, you may want to make this field longer.

smapsMustLoginBy

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.19
RDBMS Type:	character

Max Length:	36 (see note)
suppressible:	Yes

Format: *<date/time> <reason>*

Examples:	20010307171211Z Forgotten Password
	20010307171211Z Admin Reset

If set, if the user fails to login by the specified date and time, the authentication attempt will be rejected. If the user successfully authenticates *before* this date/time, APS will clear the value.

If the writable directory server is down when the user authenticates, APS will be unable to clear this attribute and the user may not be able to login at a later date.

APS will never set this field, but it will clear it upon a successful authentication prior to the specified date/time.

This field may need to be cleared to enable a user account.

Note on length: This field contains a date, followed by a comment, thus the length is truly variable. APSAdmin allows 15 characters for the date value, a space, and up to 20 characters of user-entered comment. If your site has a custom interface to this field, you may want to make this field longer.

smapsNextAction

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.20
RDBMS Type:	character
Max Length:	45
suppressible:	No

Format: *<date/time> <reason>*

Examples:	20010307171211Z ACCOUNT INACTIVITY EXPIRE
	20010307171211Z ACCOUNT PURGE

This field is used by APSExpire to trigger activity for the account. It is only concerned with the date/time value; the reason is ignored.

APS will update (or at least check for updates to) this field *every time* that it modifies any value in the user's entry.

APSExpire will process any entries that have a null value for this field, so if a site performs maintenance on a record that may change this value, it should null out the field so that APSExpire will process this account.

This field is required for APS operation and may not be omitted.

smapsOldBlob

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.21
RDBMS Type:	not used
suppressible:	Yes

Format: *<32-bit checksum>*

Examples:	2165

This attribute is only used by sites that have converted from a pre-version 3.0 release of APS and, even then, only for a short period of time, in order to detect old copies of APS utilities at their site.

This attribute is not used or needed in any other case.

smapsPassword

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.22
RDBMS Type:	not used
suppressible:	Yes

Format: *<encrypted>*

Note: This description applies to LDAP directories only. ODBC (RDBMS) directories uses a stored procedure to obtain the functionality described (the stored procedure will probably need a column to store its data). See the ODBC Queries section later in this chapter).

When a user changes their own password, APS will attempt to read back the password (after it has been changed) to obtain the hashed value that the Directory has stored and store it back into this attribute.

If the Auto Force Change setting is in effect, APS will compare this value with the value stored in the user's password attribute. If they differ, APS assumes that the user's password changed using an interface other than one provided by APS and the user will be forced to change their password.

Note that this functionality requires that users have the right to read their own password back and that the administrator credentials specified in the SiteMinder User Directory also have this ability. If not, then APS cannot read the current value of the user's password. Some LDAP implementations (such as Microsoft Active Directory) will not allow the hashed value to be read back in any case.

smapsPreviousLogin

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.23
RDBMS Type:	character
Max Length:	32
suppressible:	Yes

Format: *<date/time> <IP address>*

Examples:	20010307175245Z 192.168.42.10

This attribute holds the login date and time prior to the current login date and time (and IP address, if available, though it is not reliable).

If the last login date is to be displayed on user screens, this value should be used rather than smapsLastLogin, since that value will reflect the current login date and time.

Sites should not modify this value. It is informational only. APS does not use this value for any reason.

If this field is suppressed, no such information will be kept. To suppress its use, be sure to specify this attribute in the [MAPPINGS] section of the APS.cfg file.

smapsTotalFailures

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.24

RDBMS Type:	character
Max Length:	9
suppressible:	Yes

Format: *<number>*

Examples:	12

This attribute is informational only. APS will maintain it, but does not use it for any calculations.

This attribute contains the total number of failed logins (for any reason) for this user. It is often displayed on Help Desk panels to give the administrator an idea of the level and pattern of usage for the particular user record.

Sites should not modify this value.

If this field is suppressed, no such information will be kept. To suppress its use, be sure to specify this attribute in the **[MAPPINGS]** section of the APS.cfg file.

smapsTotalLogins

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.25
RDBMS Type:	character
Max Length:	9
suppressible:	Yes

Format: *<number>*

Examples:	25

This attribute is informational only. APS will maintain it, but does not use it for any calculations.

This attribute contains the total number of successful logins for this user. It is often displayed on Help Desk panels to give the administrator an idea of the level and pattern of usage for the particular user record. In other words, a support desk operator can differentiate, using the value in this field, between a new user, an occaisional user, or a user that authenticates to the site on a regular basis.

Sites should not modify this value.

If this field is suppressed, no such information will be kept. To suppress its use, be sure to specify this attribute in the [MAPPINGS] section of the APS.cfg file.

smfpsLockoutCounter

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.26
RDBMS Type:	character
Max Length:	5
suppressible:	Yes

Format: *<number>*

Examples:	2

This attribute is used by the Forgotten Password Services component of APS to track the number of failed attempts (for lockout purposes).

Sites should not modify this value.

This field can be completely omitted if FPS is not used by a site.

If this field is to be suppressed at a site using FPS, be sure to specify this attribute in the [MAPPINGS] section of the APS.cfg file.

smfpsLog

LDAP Type:	cis/Multi-valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.27
RDBMS Type:	special, see description
suppressible:	Yes

Format: *<date/time> <comment>*

Examples:	20010401173244Z Requesting verification

This attribute is used by the Forgotten Password Services component of APS to record the use of FPS. It *is* used by FPS to track successful and failed uses of FPS in order to enforce the Max Success Frequency and Max Attempts Frequency settings.

In ODBC (RDBMS) directories, this information is kept in a separate table. This table should contain 2 columns: one to store the user's id and another to store the log entry. The names of the fields are discussed in the section on ODBC queries later in this chapter. The maximum length of the Log Entry column is about 60 characters (15 for date, a space, and a comment).

Sites should not modify this value.

This field can be completely omitted if FPS is not used by a site.

If this field is to be suppressed at a site using FPS, be sure to specify this attribute in the [MAPPINGS] section of the APS.cfg file.

smfpsOneShot

LDAP Type:	cis/Single Valued
LDAP OID:	1.3.6.1.4.1.2552.1.1.9.28
RDBMS Type:	character
Max Length:	72
suppressible:	Yes

Format: *<encrypted>*

If the OneShotPassword capability (Unsupported "Page" Cross-Reference) of FPS is to be used, APS uses this attribute to store encrypted information about this use. This value will be cleared when used, but there is no security problem if it is retained.

The maximum length refers to the amount of space required to store a 32 character encrypted password.

Sites should not modify this value.

This field can be completely omitted if FPS is not used by a site or the OneShotPassword capability is not in use.

Forgotten Password Services (FPS) Data

The section above describes the attributes that "belong" to APS; that is, the fields that APS needs and maintains for its own purposes or for auditing purposes.

The Forgotten Password Services (FPS) component of APS allows users to change (or recover) their password *without* knowing their old password. This is done by requiring the user to enter information, usually the answers to a series of questions, and matching that information with data stored in the user record.

Unlike the above attributes, the information used in the FPS process does not "belong" to APS and is not maintained by it. The FPS process merely uses this information in the way that it is configured to do so. APS/FPS does not care what this information is, it merely needs to know the attribute names and the format (in some cases) in which it is stored.

There are three FPS-related attributes described in the previous section. Those attributes are used for control purposes and are not end-user visible, nor are they typically modified by a site outside of the FPS process. This section talks about the fields/attributes that are used by the FPS process to interact with the user.

This information is defined and maintained by whatever user management tools used by the site. APS/FPS generally does not do anything with this data except to match it up against user input (with the possible exception of certain control information used internally by APS, described later).

Most of the FPS configuration process involves the mapping of input fields (from the input forms) to the user attributes and describes how the comparison is performed. For example, the configuration might indicate that an input field called "FirstName" is compared, case-insentively, to a attribute called "givenname". This comparison is generally performed during the "Identify" phase of the FPS process in order to figure out who the user is.

The second (optional, but commonly used) phase of the FPS process involves verification; the attempt by the user to prove that they are who they say they are. This is typically done by presenting the user with one or more questions, gathering the user's answers, then comparing those answers with the ones stored in the user record.

APS does not provide any mechanism for storing these questions in the user record, for updating it, or even for displaying it. This should all be part of the User Management Application. While APS does not require specific names for such attributes, it does need to understand how that information is stored so that it can perform its operations correctly.

The most trivial example involves a single question and answer. In these cases, two attributes can be used: a Question and an Answer. These attributes can be called anything the site wants to call them and are expected to contain character data. FPS is configured to pass whatever is in the Question attribute (the name of the attribute is identified in the APS.cfg file) to the Verify form and the posted data is compared to the data stored in the Answer attribute (also named in the APS.cfg file).

There is actually an even simpler case, where the question is fixed for the entire site, so only an answer is needed. This configuration is generally not recommended, since it is trivial to socially engineer an answer to break into the user account.

Most sites prefer to use the more complex case where a number of questions and their answers are stored in the user record, allowing FPS to select a subset of those questions for display. This is a much more complicated case.

The issue is not one of where the information is stored in the user record, it is an issue of how the information is formatted within the user record.

With multiple questions and answers, the answers must be tightly coupled with the question that they belong to; the two must be stored together, rather than in separate attributes (especially on LDAP directories, where the order in which multi-valued attributes are returned is not guaranteed).

Thus, FPS expects such data to contain a question and answer, stored together, separated by a "pipe" or "vertical bar" character ("|").

FPS will also select a subset of multiple questions. Because of this, FPS needs to "remember" which questions have already been asked and when they were asked. This information **is** updated by FPS and is associated with each question/answer pair by placing it in front of the question/answer pair and separating it with another pipe character.

Thus, a single question/answer should be stored in the form:

```
<control data>|<question>|<answer>
```

FPS does not care about the format of the question. Whatever is stored between tne two pipes will be passed to the verify page. That page can do whatever translations that it must in order to display the correct user question (challenge). Typically, this is *not* the actual text of the question (this is discussed in great detail elsewhere in this document). Usually, it is a *code* (in FPS parlance, a Question ID, or QID) that the Verify page uses to lookup the actual text of the question. This method accomplishes several things:

- Reduces the amount of disk space required to store questions, since only the QID is stored, not the full text of the question.
- Questions can be internationalized, by providing locale-specific lookup tables for QIDs.
- Questions can be easily "repaired" for typos and wording by changing the text in the lookup table rather than the entire directory.

The control data portion of the data should be left blank at initialization (in other words, the entire field should just start with the pipe). FPS will write a date/time or the word DELETED into this area, depending on how it is configured.The answer is the text that FPS will match against the user input. This information is generally not encrypted, but can be, either using SmAPSEx or, in the case of ODBC directories, using stored procedures. The issue here is that the User Management Tool must be able to encrypt the information when it is stored in the user record, so both the User Management Tool and FPS must agree on the encryption scheme.

If the word DELETED appears, then this question has been asked and FPS is configured to consume questions. In these cases, FPS will never ask the question again. User Management Tools should keep such information in the user record and not allow the user to re-select the same question, because it has already been used. The question should remain in the user record to prevent the question from ever being selected again by the user.

If a date appears in the control field, it is important that the User Management Tool maintain this value, otherwise questions may not be selected correctly in the future.

The above discusses how each question/answer pair is stored in the user record. There is also an issue of how all of the question/answer pairs are stored together. FPS recognizes two different methods.

- When Format=A is specified, each question/answer pair, with its control data, is stored as a separate value in a multi-valued attribute (LDAP).
- When Format=B is used, all of the question/answer pairs (with the control data) is stored as a single string, each pair separated from all others with a caret ("^").

The name of the attribute used and the storage format is defined in the APS.cfg.

LDAP Directories

Note: APS does support Microsoft Active Directory and this support is provided using its LDAP interface. However, because Active Directory deviates so extensively from the LDAP specification, APS contains a significant amount of special processing and thus Active Directory is discussed in its own section.

Starting with Version 3.0, APS no longer stored its per-user information in a single LDAP attribute (the "Blob"); all information is stored in individual attributes. This section describes these attributes, the data that they are expected to contain (and the format) and how to set up your schema.

Sites converting from APS versions prior to version 3 should contact CA for a utility to convert their old style APS Blob to the new attributes.

Schema Updates

There is no "standard" way to update an LDAP schema. Each vendor implements their own way to do it. In addition, most sites have policies or requirements for how schemas are updated, naming conventions and data layout issues.

APS tries to place as few restrictions as possible for the schema that it needs. The attributes must exist for every user entry in the User Directory that will be maintained by APS, with the exception of the attributes marked as optional and suppressed in the [MAPPINGS] (see page 133) section of the APS Configuration File.

How sites accomplish this is entirely up to the administrators at the site. There are a number of choices available; no tools are provided to do this for you (there are some sample files provided for iPlanet LDAP directories, however). CA Professional Services has considerable experience in this area and can help or advise you, if you so desire.

There are three steps required to ensure that the attributes exist for every user:

1. The attributes must be defined as valid attributes for the directory. This can be done manually using the LDAP console or similar utility.
    - For iPlanet LDAP Server Version 4.x User Directories, the APS installation kit includes a file called smaps.user_at.conf that can be used to define these attributes. This file cannot be used directly, but can be merged with an existing schema definition to update the directory. Please refer to the iPlanet documentation for details.
    - For iPlanet LDAP Server Version 5.x User Directories, the APS installation kit includes a file called 56NetegrityAPS.ldif that can be used, with some editing, to define these attributes.
    - Other LDAP implementation will have different requirements to define these attributes and they may have to be defined manually through a vendor-supplied user interface.

Attributes need not have the names defined above. If the attributes are to be renamed, the names must be remapped as described in the [MAPPINGS] (see page 133) section.

The LDAP RFI requires that each attribute have an OID (a globally unique identifier). Some implementations do not require this or build them automatically. Other implementations not only require an OID, but that it also conform to OID formatting standards. CA has obtained and assigned the following OIDs for this use (these are also listed in the Schema section above):

Schema Object	**OID**
smapsInfo (objectClass)	1.3.6.1.4.1.2552.1.1.9.1
smapsAccountInactivityDays	1.3.6.1.4.1.2552.1.1.9.2
smapsBaseDate	1.3.6.1.4.1.2552.1.1.9.3
smapsDisableAfter	1.3.6.1.4.1.2552.1.1.9.4
smapsDisableUntil	1.3.6.1.4.1.2552.1.1.9.5
smapsExpirePasswordDays	1.3.6.1.4.1.2552.1.1.9.6
smapsFailureCount	1.3.6.1.4.1.2552.1.1.9.7
smapsFailuresSinceLastLogin	1.3.6.1.4.1.2552.1.1.9.8
smapsFailuresSincePreviousLogin	1.3.6.1.4.1.2552.1.1.9.9
smapsGenerationalRedirects	1.3.6.1.4.1.2552.1.1.9.10
smapsGraceLoginsUsed	1.3.6.1.4.1.2552.1.1.9.11
smapsHistory	1.3.6.1.4.1.2552.1.1.9.12
smapsImmediateChange	1.3.6.1.4.1.2552.1.1.9.13
smapsInactivityWarning	1.3.6.1.4.1.2552.1.1.9.14
smapsLastLogin	1.3.6.1.4.1.2552.1.1.9.15
smapsLastPasswordChange	1.3.6.1.4.1.2552.1.1.9.16
smapsLoginHistory	1.3.6.1.4.1.2552.1.1.9.17
smapsMaxFailures	1.3.6.1.4.1.2552.1.1.9.18
smapsMustLoginBy	1.3.6.1.4.1.2552.1.1.9.19
smapsNextAction	1.3.6.1.4.1.2552.1.1.9.20
smapsOldBlob	1.3.6.1.4.1.2552.1.1.9.21
smapsPassword	1.3.6.1.4.1.2552.1.1.9.22
smapsPreviousLogin	1.3.6.1.4.1.2552.1.1.9.23
smapsTotalFailures	1.3.6.1.4.1.2552.1.1.9.24
smapsTotalLogins	1.3.6.1.4.1.2552.1.1.9.25

smfpsLockoutCounter	1.3.6.1.4.1.2552.1.1.9.26
smfpsLog	1.3.6.1.4.1.2552.1.1.9.27
smfpsOneShot	1.3.6.1.4.1.2552.1.1.9.28

2. The attributes need to be assigned to an objectClass. The choice of objectClass is entirely up to the site.

   - If the site already has a custom user objectClass, the attributes should probably be assigned to it (as optional attributes).
   - The site could define a new custom objectClass. The APS installation kit includes smaps.user_oc.conf and 56NetegrityAPS.ldif files (for iPlanet LDAP Directories, versions 4.x and 5.x, respectively) that define the attributes in terms of a new custom object class called smapsInfo. However, the site can use any name they so desire; APS never even sees it.

     The LDAP RFI requires that custom objectClasses have an OID (a globally unique identifier). Some implementations do not require this or build them automatically. Other implementations not only require an OID, but that it also conform to OID formatting standards. CA has obtained and assigned "1.3.6.1.4.1.2552.1.1.9.1" as the OID for the smapsInfo objectClass.
   - It may be possible to add the new attributes to an existing, standard, objectClass, such as inetOrgPerson. This is usually not recommended by LDAP vendors, but it may be possible. iPlanet, for example, forbids it from within its Console, but it can be done by editing its configuration files. This method is generally not a good idea, but it is possible.

   An attribute can appear in multiple objectClasses.

3. The objectClass containing the attributes must be assigned to every user in the User Directory. For LDAP directories, this amounts to adding the objectClass to the list of classes stored in the object's objectClass attribute.

   Note that the selection of where to put the attributes (described in the previous step) can have a considerable impact on this step. For example, if the attributes were added to a standard objectClass, this step is already done. If an existing custom objectClass was used, then this step may already be done. Using a new objectClass will require that this step be performed.

   Also, consider new users. As new users get added to the User Directory, they will need to have access to the attributes as well, so the objectClass will need to be added to their list as well. This will probably impact any user management system implemented at the site.

To use 56NetegrityAPS.ldif on iPlanet LDAP Directories Only:

- Edit the 56NetegrityAPS.ldif file using your favorite editor
- Locate the two instances of smapsInfo (case-insensitive). Rename this class to the name of the class that you want to use. Note that if you want to add the APS attributes to an existing class, you will have to remove this class definition and add the attributes to your chosen class in the file in which it is defined (probably 99user.ldif).
- Save the file.
- Locate the file 99user.ldif in your iPlanet directory. Make sure that you locate the one associated with the desired LDAP directory instance.
- Put the modified 56NetegrityAPS.ldif file into the same directory.
- Restart the LDAP Directory Server.

Note: These steps only need to be performed on a single Master Directory. The changes will be propagated to other masters and consumers (though the changes will be recorded in subsidiary servers in the 99user.ldif file).

Special Setup

There is no special setup for standard LDAP User Directories.

Native Password Policy Support

There is no standard for LDAP native password policies. However, many vendors have implemented them using extended services (how extended services are accessed is defined in the LDAP standard, but the particular details of such extensions is vendor-specific).

Expiration policies are enforced by LDAP directories when the user attempts to authenticate to the directory. This authentication is actually performed by SiteMinder, before APS is invoked. Any failure, even if accompanied by extended information such as Password Expired, is seen by SiteMinder as an authentication failure and so reported to APS. Thus, APS is unable to support such policies.

Password content policies are enforced during the password change process. Most LDAP implementations will refuse a password change if it does not conform to a native password policy. However, this refusal is returned as a generic error code and (vendor-specific) extensions need to be used to determine the specific reason for the refusal. APS does not support native password content policies on standard LDAP user directories.

This second restriction is relatively easy to avoid. APS will apply its own password content policies *before* attempting to update the password in the underlying directory. If the APS content policy is at least as restrictive as the Directory's policy, then the password will already conform the the directory's rules. Bear in mind that this method cannot accommodate the password history restrictions.

Expiration policies are considerably more difficult to accommodate. Most sites simply turn off any native expiration policies and use SiteMinder (with APS) to perform all authentications. If LDAP is to be used directly by some applications, those applications should be modified to update APS' control information on every authentication.

Disabling/Enabling User Accounts

If the user satisfies the conditions for any Ignore keyword in the APS.cfg file, APS will not perform any authentication time processing for this user, including detecting if the user is disabled or detecting disabling conditions (such as password expired).

Recognizing a Disabled User

APS will *recognize* that a user is disabled if any of the following are true (this is not necessarily the order in which APS actually does detection). Note that the Use Internal Disables setting does not affect detection, only how APS actually disables users.

- If the user is a member of a standard LDAP (forward) group whose name starts with "DISABLED". APS searches for this membership using the following information:
  - The Search Base as defined in the associated SiteMinder User Directory entry.
  - A group objectclass of groupOfUniqueNames (can be remapped).
  - The group name attribute of GroupCN (by default, maps to cn, but can be remapped to another name).
  - A value for its uniqueMember (also remappable) attribute that contains this user's full DN.

  The Disabled Reason is derived from the text following "DISABLED" in the group's name (any additional hyphen is removed).

  This search cannot be suppressed.
- If the user is a member of a standard LDAP (reverse) group whose name starts with "DISABLED".
  - APS uses the attribute memberof (remappable) in the current user's entry. This attribute is assumed to contain the DN's of the (reverse) groups in which the user is a member.
  - The DN is assumed to start with the group's name attribute, which is remapped from GroupDN (by default, maps to cn, but can be remapped to another name).

  - If the group's DN starts with "DISABLED", the user is considered disabled.

  The Disabled Reason is derived from the text following "DISABLED" in the group's name (any additional hyphen is removed).

  This search can be suppressed by remapping "memberof" to a null value.

- If the user satisfies any of the conditions defined in APS.cfg using the Disable, the user will be considered disabled. The keyword defines the Disabled Reason code.
- If the smapsDisabledUntil field starts with the word "FOREVER", the user account will be considered disabled. The rest of the text (after "FOREVER") in the attribute will be used as the Disabled Reason code.
- If the smapsDisabledUntil field contains a date/time that is in the future, the user account will be considered disabled. The rest of the text (after the date/time) in the attribute will be used as the Disabled Reason code.
- If the smapsDisabledAfter field contains a date/time that is in the past, the user account will be considered disabled. The rest of the text (after the date/time) in the attribute will be used as the Disabled Reason code.
- If the smapsMustLoginBy field contains a date/time that is in the past, the user will be considered disabled. The rest of the text (after the date/time) in the attribute will be used as the Disabled Reason code.

The account may *become* disabled (as described in the next section) during authentication, but this will set one of the above conditions.

Disabling a User

APS itself can disable a user for only four reasons:

- Exceeded Failure Count. This can only occur during the authentication process.
  - If Auto Reset Failure Count is in effect:

    smapsDisabledUntil is set to the current date and time, plus the number of minutes indicated by the Max Failures On Change followed by the text "Failure Count".
  - Otherwise:
    - If the Use Internal Disables is in effect, smapsDisabledUntil is set to "FOREVER Failure Count".
    - If the Use Internal Disables is **not** in effect, the user is added to the Disabled-FailureCount group. APS will search for this group using the GroupCN attribute (defaults to cn, can be remapped). If the group does not exist, it will be created in the OU defined by the LDAP Disabled Groups setting.

- Password Expired (any grace period expired and all grace logins expended). This can be done either at authentication time *or* by APSExpire.
  - If the Use Internal Disables setting is in effect, smapsDisabledUntil is set to "FOREVER Password Expired".
  - If the Use Internal Disables setting is **not** in effect, the user is added to the Disabled-PasswordExpired group. APS will search for this group using the GroupCN attribute (defaults to cn, can be remapped). If the group does not exist, it will be created in the OU defined by the by the LDAP Disabled Groups setting.
- Account Expired due to inactivity. This can be done either at authentication time *or* by APSExpire.
  - If the Use Internal Disables setting is in effect, smapsDisabledUntil is set to "FOREVER User Expired".
  - If the Use Internal Disables setting is **not** in effect, the user is added to the Disabled-UserExpired group. APS will search for this group using the GroupCN attribute (defaults to cn, can be remapped). If the group does not exist, it will be created in the OU defined by the by the LDAP Disabled Groups setting.
- Locked out from FPS misuse. This can only be done by the FPS process.
  - If the Use Internal Disables setting is in effect, smapsDisabledUntil is set to "FOREVER FPS Lockout".
  - If the Use Internal Disables setting is **not** in effect, the user is added to the Disabled-FPSLockout group. APS will search for this group using the GroupCN attribute (defaults to cn, can be remapped). If the group does not exist, it will be created in the OU defined by the by the LDAP Disabled Groups setting.

Enabling a User Account

APS never really enables an account itself. Some of the mechanisms used by APS to maintain the user status have built-in reset capabilities (dates). APS does not actually update a user record to re-enable it.

To re-enable an account, a site must ensure that all of the above criteria that APS uses to detect that a user account is already disabled are not true. In other words, the user account cannot be a member of a disabled group, smapsDisableUntil must be set correctly, etc.

Even then, it may appear that a user remains disabled. The usual cause is that while the user account does get enabled, the conditions that caused the account to become disabled remain, thus causing APS to just disable the account again. These reasons are:

- The password remains expired. Either set smapsBaseDate (recommended) or smapsLastPasswordChange (not recommended) to a more recent value.
- The account remains expired. Either set smapsBaseDate (recommended) or smapsLastLogin (not recommended) to a more recent value.
- smapsFailureCount shows that the user remains locked out (see the description of the field at the beginning of this chapter for details on how to reset this field - it is not sufficient to merely delete its value).

Special APS Processing/Limitations

For the most part, all APS features are supports by standard LDAP directories (of course, "standard" is the operative word).

The IsLDAP() function (used in settings overrides) returns TRUE if the current user is stored in an LDAP directory (of any type).

The LDAP Disabled Groups setting controls where APS will *create* any disabled groups within the directory, if the Use Internal Disables setting is **not** in effect.

The Check LDAP Password Existence and Auto Force Change settings may or may not be supported, depending on whether the implementation allows an LDAP client to read the (hashed) user password.

LDAP Reverse Groups

When APS was originally designed, LDAP supported groups in one way: what are now call forward groups.

A *forward* group is a group object, physically stored, that contains the DN of all of its members in a multi-valued attribute (usually uniqueMember).

Forward groups do not scale well. In order to provide more scalability, the LDAP standards committee defined reverse groups.

A reverse group is a group object, physically stored, whose DN is contained in its member's records (usually memberof or groups). In other words, each user contains a list of the groups of which it is a member in a multi-valued attribute. Reverse groups scale much better than forward groups.

APS supports forward groups by default, but can support reverse groups, if they are used at the site. It can also support a mix of forward and reverse groups.

You must tell APS which groups will be reverse groups (since there is no standard, quick, way to tell the difference). You configure this in the [Mappings] (see page 133) section of the APS Configuration File.

To have APS treat *all* LDAP groups as reverse groups, use:

```
LDAP.ReverseGroups=*
```

An asterisk should not be considered a wildcard. It can only be used in the above manner to indicate that *all* groups are reverse groups.

To specify a single group as a reverse group:

```
LDAP.ReverseGroups=cn=Disabled-NoCredit, o=nds.com
```

Multiple such lines can exist, one for each reverse group.

Note: Some LDAP vendors has also implement something that they call virtual groups. APS does not support virtual groups as of this writing, but, for most purposes, they can be easily simulated using override expressions or using the Disable setting.

Forgotten Password Services (FPS)

All FPS functionality is supported for standard LDAP implementations, with the exception of question/answer encryption. This can still be accomplished, however, using the SmAPSEx library.

APSExpire

APSExpire has special settings for LDAP directories so that the user directory can be partitioned into smaller "chunks" for processing. See the chapter on Daily Processing (APSExpire) (see page 297), for details.

APSAdmin

APSAdmin fully supports LDAP User Directories. However, APSAdmin only supports the entry of the full user DN on the user selection panel.

Fault Tolerance & Performance

A considerable amount of work has gone into APS to optimize performance when dealing with LDAP directories.

APS reads the entire user entry into memory before handling any request. This means that all settings overrides and attribute references required by email templates canbe handled without another request to the directory server. The additional overhead of this single read is nominal, the performance of the read is the lookup and network latency, not the actual data transfer.

When writing to the directory, APS "gangs" all of the updates into a single request to the directory server. This is a huge performance gain. While suppressing individual attributes can improve storage resource utilization (disk space), it has little impact on the update performance; the performance overhead for the entire update far outweighs the incremental performance cost of additional fields (as long as the updates are submitted together).

A side-effect of the ganged updates is improved fault tolerance. LDAP guarantees that all writes submitted as a single request either succeed or fail together. No single field will be updated without the others.

What this means, for example, is that the password cannot be updated without the password change date attribute. They either both update or neither updates.

In a multi-mastering situation, this is even more critical. Parts of the update cannot go to each of two servers. All of the updates will be posted to the same place at the same time.

APS does *not* use SiteMinder's Enhanced Referrals. It does not keep pools of LDAP handles, pinging them periodically to determine connection health and to keep the connections from timing out. It *does* use SiteMinder handles for reading (and, in some cases, writing) to the directory server, but requests normal handles from SiteMinder.

This means that APS fully supports multi-mastering. With no writeback settings (discussed below), APS follows write referrals to any or all masters that it needs to accomplish its task. This is entirely handled within the LDAP SDK and is subject to its own performance issues.

LDAP supports master/consumer servers, where one or more directory servers are *masters* and others are *consumers*. Consumer servers are, by definition, *read only* and are tuned for high performance reads. Typically, SiteMinder is configured to deal with consumers and all writes should go to the master(s). When a change goes to the master, it replicates that change to other masters and any consumers.

However, LDAP implements such relationships using *referrals*. When the write is made to a consumer server, the server responds to the LDAP client with a special message that says "I am a consumer, please resubmit your request to *<list of master servers>*". The LDAP client is expected to (possibly open a new connection and) resubmit the request to each of the master servers in the list until the request is satisfied.

The LDAP SDK can be told to accomplish this automatically and, in fact, when SiteMinder Enhanced Referrals are turned *off*, it does so. APS uses such handles when it uses SiteMinder handles.

However, even this mechanism has a fairly significant performance penalty when doing a large number of writes: every write is at least two round trips to directory servers; the first to the consumer which then returns the referral.

Sites can bypass this referral process within APS by specifying LDAP Write Back Information*. This tells APS that one or more master servers exist and to do all writes to that list instead of submitting the update request to the SiteMinder-supplied handle.*

Known Directory Idiosyncrasies

A few implementation-dependent "features" have created some idiosyncrasies for APS support:

- Some LDAP implementers have disallowed the reading of the hashed user password. In these cases, APS cannot support the Check LDAP Password Existence and Auto Force Change settings. This is implementation-dependent.
- Some LDAP implementations (Websphere among them) only recognize the first *n* characters of attribute names as being unique. Some APS attributes may be too long or identical out to this limit, causing issues with the directory. This problem is easily solved by remapping the names of the attributes to shorter names.
- If the LDAP Disabled Groups setting is not defined and APS needs to create a new disabled group, it attempts to create the group at the root of the directory tree. It determines this root by using the *last* component of the user's DN. Some implementations of LDAP default the last two components of the default DN to domain components ("dc") rather than organization ("o"). In these configurations, there is no container for the last component; the last two components must be used instead. APS is not aware of this and will fail to create the group.
- This is easily solved in one of two ways: Either use the LDAP Disabled Groups setting to explicity specify where groups should be created or pre-create all of the groups so that APS does not need to create them.
- The same physical LDAP directory is sometimes configured to SiteMinder multiple times, either with different search criteria or with different administrator credentials. Since APS tries to match up the current user's LDAP directory (as specified by SiteMinder as the server's IP address or network name) with that specified within the SiteMinder Policy Store, these cases can cause problems when APS selects the *wrong* SiteMinder User Directory (thus either using the wrong credentials or the wrong search base).

  This problem can be solved by defining multiple *names* for the same LDAP directory within the Policy Server's host file, all with the same IP address (alternatively, multiple DNS aliases could also be used). Then the multiple User Directories can be specified within the SiteMinder Policy Store, each with a unique name. APS can then determine the correct entry to use.

- Some LDAP implementations impose security within the directory, most notably iPlanet with its Access Control Information (ACIs). This is not part of the LDAP specification at all and can make troubleshooting APS very difficult. CA generally recommends using administrator credentials with full rights to user entries. In the case of iPlanet, the cn=Directory Manager administrator bypasses ACI checking altogether.
- For iPlanet User Directories, the cn=Directory Manager administrator has some special processing on the directory server. Specifically, it bypasses all security (ACIs), it has no limits on its resource limitations (timeouts and size limits), and it receives priority processing.

  APS is very careful of its resource utilization, so its use of the Directory Manager can be considered "safe"; it will not consume unreasonable amounts of server resources. However, the use of this account must be determined by the site.
- LDAP directories must be writable for APS to operate at full function. Advanced Password Services saves information to each user's record in the LDAP directory (in particular, the new password), so the directory must not prevent active writing.

  Backup LDAP servers, as of this writing, are, by the LDAP specification, read-only servers. If SiteMinder is configured to use a backup LDAP server, APS will be unable to write to the directory. Errors will be logged to the console indicating this state.

  This problem can sometimes be avoided using the writeback settings in the APS Configuration File, but only if the Primary LDAP Directory Server is operational at all times.

APS supports LDAP multi-mastering in two ways:

- It will automatically follow any write referrals sent back as the result of a write to a consumer server.
- The LDAP Write Back Information allow the specification of more than one master server.

  Some implementations of LDAP are actually LDAP interfaces to other data stores. Many of these implementations provide multi-mastering on their own. Examples are X.500, ISOCOR and Oracle's LDAP implementation. CA does not recommend any particular implementation.

  While APS does generally support the same consumer failover as SiteMinder and fully supports LDAP multi-mastering, the FPS subsystem does not currently support consumer failover.

Microsoft Active Directories

Note: APS does support Microsoft Active Directory and this support is provided using its LDAP interface. However, because Active Directory deviates so extensively from the LDAP specification, APS contains a significant amount of special processing and thus Active Directory is discussed in its own section.

APS supports Microsoft Active Directories running in LDAP mode only.

Schema Updates

APS tries to place as few restrictions as possible for the schema that it needs, in part because many sites have policies or requirements for how schemas are updated, naming conventions and data layout issues. The attributes must exist for every user entry in the User Directory that will be maintained by APS, with the exception of the attributes marked as optional and suppressed in the [MAPPINGS] section of the APS Configuration File.

How sites accomplish this is entirely up to the administrators at the site. There are a number of choices available; no tools are provided to do this for you. CA Professional Services has considerable experience in this area and can help or advise you, if you so desire.

There are three steps required to ensure that the attributes exist for every user:

1. The attributes must be defined as valid attributes for the directory. This can be done manually using the LDAP console or similar utility.

   Attributes need not have the names defined above. If the attributes are to be renamed, the names must be remapped.

   Microsoft Active Directory requires that each attribute have an OID (a globally unique identifier). CA has obtained and assigned the following OIDs for this use (these are also listed in the Schema section above):

Schema Object	**OID**
smapsInfo (objectClass)	1.3.6.1.4.1.2552.1.1.9.1
smapsAccountInactivityDays	1.3.6.1.4.1.2552.1.1.9.2
smapsBaseDate	1.3.6.1.4.1.2552.1.1.9.3
smapsDisableAfter	1.3.6.1.4.1.2552.1.1.9.4
smapsDisableUntil	1.3.6.1.4.1.2552.1.1.9.5
smapsExpirePasswordDays	1.3.6.1.4.1.2552.1.1.9.6
smapsFailureCount	1.3.6.1.4.1.2552.1.1.9.7
smapsFailuresSinceLastLogin	1.3.6.1.4.1.2552.1.1.9.8

smapsFailuresSincePreviousLogin	1.3.6.1.4.1.2552.1.1.9.9
smapsGenerationalRedirects	1.3.6.1.4.1.2552.1.1.9.10
smapsGraceLoginsUsed	1.3.6.1.4.1.2552.1.1.9.11
smapsHistory	1.3.6.1.4.1.2552.1.1.9.12
smapsImmediateChange	1.3.6.1.4.1.2552.1.1.9.13
smapsInactivityWarning	1.3.6.1.4.1.2552.1.1.9.14
smapsLastLogin	1.3.6.1.4.1.2552.1.1.9.15
smapsLastPasswordChange	1.3.6.1.4.1.2552.1.1.9.16
smapsLoginHistory	1.3.6.1.4.1.2552.1.1.9.17
smapsMaxFailures	1.3.6.1.4.1.2552.1.1.9.18
smapsMustLoginBy	1.3.6.1.4.1.2552.1.1.9.19
smapsNextAction	1.3.6.1.4.1.2552.1.1.9.20
smapsOldBlob	1.3.6.1.4.1.2552.1.1.9.21
smapsPassword	1.3.6.1.4.1.2552.1.1.9.22
smapsPreviousLogin	1.3.6.1.4.1.2552.1.1.9.23
smapsTotalFailures	1.3.6.1.4.1.2552.1.1.9.24
smapsTotalLogins	1.3.6.1.4.1.2552.1.1.9.25
smfpsLockoutCounter	1.3.6.1.4.1.2552.1.1.9.26
smfpsLog	1.3.6.1.4.1.2552.1.1.9.27
smfpsOneShot	1.3.6.1.4.1.2552.1.1.9.28

2. The attributes need to be assigned to an objectClass. The choice of objectClass is entirely up to the site.

   - If the site already has a custom user objectClass, the attributes should probably be assigned to it (as optional attributes).
   - The site could define a new *custom* objectClass.
   - Active Directory requires that custom objectClasses have an OID (a globally unique identifier). CA has obtained and assigned "1.3.6.1.4.1.2552.1.1.9.1" as the OID for the smapsInfo objectClass. If a different name is used, a different OID should also be obtained.
   - It may be possible to add the new attributes to an existing, standard, objectClass, such as inetOrgPerson. This is usually not recommended by LDAP vendors. This method is generally not a good idea, but it is possible.

   An attribute can appear in multiple objectClasses.

3. The objectClass containing the attributes must be assigned to every user in the User Directory. For LDAP directories, this amounts to adding the objectClass to the list of classes stored in the object's objectClass attribute.

   Note that the selection of where to put the attributes (described in the previous step) can have a considerable impact on this step. For example, if the attributes were added to a standard objectClass, this step is already done. If an existing custom objectClass was used, then this step may already be done. Using a new objectClass will require that this step be performed.

   Also, consider new users. As new users get added to the User Directory, they will need to have access to the attributes as well, so the objectClass will need to be added to their list as well. This will probably impact any user management system implemented at the site.

Special Setup

Certain remappings are required in the APS.cfg file to identify and process Active Directories. The following three lines are *the* minimum required to support an Active Directory as a user store (in the [Mapping] section of the APS.cfg file):

```
userPassword={IsInDirectory("<dirname>")} unicodePwd
inetOrgPerson={IsInDirectory("<dirname>")} user
smapsPassword ={ IsInDirectory("<dirname>") }
```

Note that <*dirname*> is the name of the User Directory entry in your Policy Store.

If the only User Directory is an Active Directory, then the following three lines can be used:

```
userPassword=unicodePwd
inetOrgPerson=user
smapsPassword=
```

Native Password Policy Support

APS does not support Active Directory password content policies (the policies that define the required format for a password). Active Directory will refuse a password change if it does not conform to its defined native password policy. However, this refusal is returned as a generic error code.

This restriction is relatively easy to avoid. APS will apply its own password content policies before attempting to update the password in the directory. If the APS content policy is at least as restrictive as the Active Directory's policy, then the password will already conform the the directory's rules. Bear in mind that this method cannot accommodate password history restrictions.

APS does support some of Active Directory's native *expiration* policies.

- When calculating password expiration dates, it uses the PasswordExpirationDate attribute (can be remapped) and compares it to its own password expiration date. APS will use whichever value is sooner. The Password Warning date will be calculated backwards from the resulting password expiration date.
- When it needs the date/time that the password was last changed, APS uses the pwdLastSet (remappable) attribute or its own value, whichever is earliest.
- When calculating account expiration dates, it uses the accountExpires attribute (can be remapped) and compares it to its own account expiration date. APS will use whichever value is sooner. The Account Inactivity Warning date will be calculated backwards from the resulting account expiration date.
- When calculating the last login date, it uses the lastLogon attribute (can be remapped) and compares it to its own last login date. APS will use whichever value is later.
- APS honors Active Directory's immediate change flag.

Disabling/Enabling User Accounts

If the user satisfies the conditions for any Ignore keyword in the APS.cfg file (page 55), APS will not perform any authentication time processing for this user, including detecting if the user is disabled or detecting disabling conditions (such as password expired).

Recognizing a Disabled User

APS will *recognize* that a user is disabled if any of the following are true (this is not necessarily the order in which APS actually does detection). Note that the Use Internal Disables setting does not affect detection, only how APS actually disables users.

- If the user is a member of a standard LDAP (forward) group whose name starts with "Disabled". APS searches for this membership using the following information:
  - The Search Base as defined in the associated SiteMinder User Directory entry.
  - A group objectclass of groupOfUniqueNames (can be remapped).
  - The group name attribute of GroupCN (by default, maps to cn, but can be remapped to another name).
  - A value for its uniqueMember (also remappable) attribute that contains this user's full DN.

  The Disabled Reason is derived from the text following "Disabled" in the group's name (any additional hyphen is removed).

  This search cannot be suppressed.

- If the user is a member of a standard LDAP (reverse) group whose name starts with "Disabled".
  - APS uses the attribute memberof (remappable) in the current user's entry. This attribute is assumed to contain the DN's of the (reverse) groups in which the user is a member.
  - The DN is assumed to start with the group's name attribute, which is remapped from GroupDN (by default, maps to cn, but can be remapped to another name).
  - If the group's DN starts with "Disabled", the user is considered disabled.

  The Disabled Reason is derived from the text following "Disabled" in the group's name (any additional hyphen is removed).

  This search can be suppressed by remapping memberof to a null value.
- If the user satisfies any of the conditions defined in APS.cfg using the Disable setting, the user will be considered disabled. The keyword defines the Disabled Reason code.
- If the smapsDisabledUntil field starts with the word "FOREVER", the user will be considered disabled. The rest of the text (after "FOREVER") in the attribute will be used as the Disabled Reason code.
- If the smapsDisabledUntil field contains a date/time that is in the future, the user will be considered disabled. The rest of the text (after the date/time) in the attribute will be used as the Disabled Reason code.
- If the smapsDisabledAfter field contains a date/time that is in the past, the user will be considered disabled. The rest of the text (after the date/time) in the attribute will be used as the Disabled Reason code.
- If the smapsMustLoginBy field contains a date/time that is in the past, the user will be considered disabled. The rest of the text (after the date/time) in the attribute will be used as the Disabled Reason code.
- The account may become disabled (as described in the next section) during authentication, but this will set one of the above conditions.

Disabling a User

APS itself can disable a user for only four reasons (APS will never set the native account status bits):

- Exceeded Failure Count. This can only occur during the authentication process.
  - If Auto Reset Failure Count is in effect:

    smapsDisabledUntil is set to the current date and time, plus the number of minutes indicated by the Max Failures On Change setting followed by the text "Failure Count".
  - Otherwise:
    - If the Use Internal Disables setting is in effect, smapsDisabledUntil is set to "FOREVER Failure Count".
    - If the Use Internal Disables setting is **not** in effect, the user is added to the Disabled-FailureCount group. APS will search for this group using the GroupCN attribute (defaults to cn, can be remapped). If the group does not exist, it will be created in the OU defined by the LDAP Disabled Groups setting.
- Password Expired (any grace period expired and all grace logins expended). This can be done either at authentication time *or* by APSExpire.
  - If the Use Internal Disables setting is in effect, smapsDisabledUntil is set to "FOREVER Password Expired".
  - If the Use Internal Disables setting is **not** in effect, the user is added to the Disabled-PasswordExpired group. APS will search for this group using the GroupCN attribute (defaults to cn, can be remapped). If the group does not exist, it will be created in the OU defined by the LDAP Disabled Groups setting.
- Account Expired due to inactivity. This can be done either at authentication time *or* by APSExpire.
  - If the Use Internal Disables setting is in effect, smapsDisabledUntil is set to "FOREVER User Expired".
  - If the Use Internal Disables setting is **not** in effect, the user is added to the Disabled-UserExpired group. APS will search for this group using the GroupCN attribute (defaults to cn, can be remapped). If the group does not exist, it will be created in the OU defined by the LDAP Disabled Groups setting.
- Locked out from FPS misuse. This can only be done by the FPS process.
  - If the Use Internal Disables setting is in effect, smapsDisabledUntil is set to "FOREVER FPS Lockout".
  - If the Use Internal Disables setting is **not** in effect, the user is added to the Disabled-FPSLockout group. APS will search for this group using the GroupCN attribute (defaults to cn, can be remapped). If the group does not exist, it will be created in the OU defined by the LDAP Disabled Groups setting.

Enabling a User Account

APS never really enables an account itself. Some of the mechanisms used by APS to maintain the user status have built-in reset capabilities (dates). APS does not actually update a user record to re-enable it.

To re-enable an account, a site must ensure that all of the above criteria that APS uses to detect that it is already disabled are not true. In other words, the account cannot be a member of a disabled group, smapsDisableUntil must be set correctly, etc.

Even then, it may appear that an account remains disabled. The usual cause is that while the account *does* get enabled, the conditions that caused it to *become* disabled remain, thus causing APS to just disable it again. These reasons are:

- The password remains expired. Either set smapsBaseDate (recommended) or smapsLastPasswordChange (not recommended) to a more recent value.
- The account remains expired. Either set smapsBaseDate (recommended) or smapsLastLogin (not recommended) to a more recent value.
- smapsFailureCount shows that the user remains locked out (see the description of the field at the beginning of this chapter for details on how to reset this field - it is not sufficient to merely delete its value).

Special APS Processing/Limitations

Most APS features are supports by Active Directory, but there is some special processing in addition to the native support above.

Active Directory requires that passwords be updated using LDAPS (LDAP over SSL) connections. If writebacks are used, this means that the LDAP Writeback Use SSL setting may be required.

If not using writebacks, the "Use Secure Connections" checkbox must be selected in the SiteMinder User Directory entry.

The IsLDAP() function (used in settings overrides) returns TRUE if the current user is stored in an LDAP directory (of any type).

The LDAP Disabled Groups setting controls where APS will create any disabled groups within the directory, if the Use Internal Disables setting is **not** in effect.

The Check LDAP Password Existence and Auto Force Change setting will not function under Active Directory.

LDAP Reverse Groups

When APS was originally designed, LDAP supported groups in one way: what are now call forward groups.

A *forward* group is a group object, physically stored, that contains the DN of all of its members in a multi-valued attribute (usually uniqueMember).

Forward groups do not scale well. In order to provide more scalability, the LDAP standards committee defined reverse groups.

A reverse group is a group object, physically stored, whose DN is contained in its member's records (usually memberof or groups). In other words, each user contains a list of the groups of which it is a member in a multi-valued attribute. Reverse groups scale much better than forward groups.

APS supports forward groups by default, but can support reverse groups, if they are used at the site. It can also support a mix of forward and reverse groups.

You must tell APS which groups will be reverse groups (since there is no standard, quick, way to tell the difference). You configure this in the [Mappings] section of the APS Configuration File.

To have APS treat *all* LDAP groups as reverse groups, use:

```
LDAP.ReverseGroups=*
```

An asterisk should not be considered a wildcard. It can only be used in the above manner to indicate that all groups are reverse groups.

To specify a single group as a reverse group:

```
LDAP.ReverseGroups=cn=Disabled-NoCredit, o=nds.com
```

Multiple such lines can exist, one for each reverse group.

Note: Some LDAP vendors also implement something that they call virtual groups. APS does not support virtual groups as of this writing, but, for most purposes, they can be easily simulated using override expressions or using the Disable setting.

Forgotten Password Services (FPS)

All FPS functionality is supported for Active Directories, with the exception of question/answer encryption. This can still be accomplished, however, using the SmAPSEx library.

APSExpire

APSExpire has special settings for LDAP directories so that the user directory can be partitioned into smaller "chunks" for processing. See the chapter entitled Daily Processing (APSExpire) (see page 297) for details.

APSAdmin

APSAdmin fully supports LDAP User Directories. However, APSAdmin only supports the entry of the full user DN on the user selection panel.

Fault Tolerance & Performance

A considerable amount of work has gone into APS to optimize performance when dealing with LDAP directories and this optimization applies directly to Active Directory implementations.

APS reads the entire user entry into memory before handling any request. This means that all settings overrides and attribute references required by email templates can be handled without another request to the directory server. The additional overhead of this single read is nominal, the performance of the read is the lookup and network latency, not the actual data transfer.

When writing to the directory, APS "gangs" all of the updates into a single request to the directory server. This is a huge performance gain. While suppressing individual attributes can improve storage resource utilization (disk space), it has little impact on the update performance; the performance overhead for the entire update far outweighs the incremental performance cost of additional fields (as long as the updates are submitted together).

A side-effect of the ganged updates is improved fault tolerance. LDAP guarantees that all writes submitted as a single request either succeed or fail together. No single field will be updated without the others.

What this means, for example, is that the password cannot be updated without the password change date attribute. They either both update or neither updates.

In a multi-mastering situation, this is even more critical. Parts of the update cannot go to each of two servers. All of the updates will be posted to the same place at the same time.

APS does not use SiteMinder's Enhanced Referrals. It does not keep pools of LDAP handles, pinging them periodically to determine connection health and to keep the connections from timing out. It *does* use SiteMinder handles for reading (and, in some cases, writing) to the directory server, but requests normal handles from SiteMinder.

This means that APS fully supports multi-mastering. With no writeback settings (discussed below), APS follows write referrals to any or all masters that it needs to accomplish its task. This is entirely handled within the LDAP SDK and is subject to its own performance issues.

LDAP supports master/consumer servers, where one or more directory servers are masters and others are consumers. Consumer servers are, by definition, *read only* and are tuned for high performance reads. Typically, SiteMinder is configured to deal with consumers and all writes should go to the master(s). When a change goes to the master, it replicates that change to other masters and any consumers.

However, LDAP implements such relationships using referrals. When the write is made to a consumer server, the server responds to the LDAP client with a special message that says "I am a consumer, please resubmit your request to *<list of master servers>*". The LDAP client is expected to (possibly open a new connection and) resubmit the request to each of the master servers in the list until the request is satisfied.

The LDAP SDK can be told to accomplish this automatically and, in fact, when SiteMinder Enhanced Referrals are turned off, it does so. APS uses such handles when it uses SiteMinder handles.

However, even this mechanism has a fairly significant performance penalty when doing a large number of writes: every write is at least two round trips to directory servers; the first to the consumer which then returns the referral.

Sites can bypass this referral process within APS by specifying LDAP Write Back Information*. This tells APS that one or more master servers exist and to do all writes to that list instead of submitting the update request to the SiteMinder-supplied handle.*

Known Directory Idiosyncrasies

A few Active Directory "features" have created some idiosyncrasies for APS support:

- Active Directory requires that passwords be updated using LDAPS (LDAP over SSL) connections. If writebacks are used, this means that the LDAP Writeback Use SSL setting may be required. If not using writebacks, the "Use Secure Connections" checkbox must be selected in the SiteMinder User Directory entry.
- If the LDAP Disabled Groups setting is not defined and APS needs to create a new disabled group, it attempts to create the group at the root of the directory tree. It determines this root by using the last component of the user's DN. Some implementations of LDAP default the last two components of the default DN to domain components ("dc") rather than organization ("o"). In these configurations, there is no container for the last component; the last *two* components must be used instead. APS is not aware of this and will fail to create the group.

  This is easily solved in one of two ways: Either use the LDAP Disabled Groups setting to explicitly specify where groups should be created or pre-create all of the groups so that APS does not need to create them.
- The same physical LDAP directory is sometimes configured to SiteMinder multiple times, either with different search criteria or with different administrator credentials. Since APS tries to match up the current user's LDAP directory (as specified by SiteMinder as the server's IP address or network name) with that specified within the SiteMinder Policy Store, these cases can cause problems when APS selects the wrong SiteMinder User Directory (thus either using the wrong credentials or the wrong search base).

  This problem can be solved by defining multiple names for the same LDAP directory within the Policy Server's host file, all with the same IP address (alternatively, multiple DNS aliases could also be used). Then the multiple User Directories can be specified within the SiteMinder Policy Store, each with a unique name. APS can then determine the correct entry to use.
- Some LDAP implementations impose security within the directory, most notably iPlanet with its Access Control Information (ACIs). This is not part of the LDAP specification at all and can make troubleshooting APS very difficult. CA generally recommends using administrator credentials with full rights to user entries. In the case of iPlanet, the "cn=Directory Manager" administrator bypasses ACI checking altogether.
- APS does not support Active Directory Servers that have a space embedded in their name.

Microsoft Windows Domains

APS does provide support for Windows NT domain directories. However, this support is severely limited because the schema in such directories cannot be extended. On the other hand, APS does support several native capabilities of such directories.

Schema Updates

Windows NT domain directories do not support schema changes, so none of the schema attributes described in the first section of this chapter are possible.

Special Setup

There is a small amount of additional setup required for APS to support Windows NT domain directories.

- The system security policy on the Policy Server must allow users to change their passwords without logging in.
- The Policy Server must be trusted by the domain controller.

When APS sends email to a user, it needs access to the user's email address. However, there is no place to store this information in the schema. APS can, however, still do this.

For Windows NT users, APS looks at the Description field in the user's entry. Within that field, APS looks for the string Email:. The text immediately following, up to the end of the description field or the next space, will be used as the user's email address.

Native Password Policy Support

APS does not support Windows Domain Directory password content policies (the policies that define the required format for a password). The Domain Controller will refuse a password change if it does not conform to its defined native password policy. However, this refusal is returned as a generic error code.

This restriction is relatively easy to avoid. APS will apply its own password content policies before attempting to update the password in the directory. If the APS content policy is at least as restrictive as the Domain Controller's policy, then the password will already conform the the directory's rules. Bear in mind that this method cannot accommodate password history restrictions, since the directory's password history is not accessible.

APS has limited support for native expiration policies. In fact, APS has little of its own such policies with this type of directory because it cannot maintain its own information within the user record.

APS cannot keep track of the last password change date. Normally, it would use this date to calculate the password expiration date. Instead, APS will use the actual password expiration date stored natively in the user record. However, APS will perform no password expiration unless it has a password expiration delay defined. This is important to note. The password expiration delay so configured is completely ignored by APS, but is needed to trigger APS to check the user record for password expiration.

APS honors the account expiration date stored natively on the user record. It does not calculate its own date at any time. As is the case with password expiration, this date is only checked if there is an account inactivity delay configured in APS.cfg. The actual setting is ignored, but instead triggers the APS activity.

APS will check and honor the immediate password change flag stored in the user record.

APS honors the native account disabled flags. It will also set the flag, if it needs to disable the user.

Disabling/Enabling User Accounts

If the account satisfies the conditions for any Ignore (see page 79) keyword in the APS.cfg file, APS will not perform any authentication time processing for this user, including detecting if the account is disabled or detecting disabling conditions (such as password expired).

Recognizing a Disabled Account

APS recognizes that an account is disabled if the native account disabled flag is set in the user record. In this case, the reason code will be Windows.

If the account satisfies any of the conditions defined in APS.cfg using the Disable setting, it will also be considered disabled. The keyword defines the Disabled Reason code.

Disabling a User Account

If APS needs to disable an account for any reason, it will set the native account disabled flag. Note that the reason that the account was disabled is lost in this case, since there is no place to store such information.

Enabling a User Account

To enable an account, merely reset the native account disabled flag and ensure that any conditions defined using the Disable setting are not satisfied.

However, this can still prevent the user from authenticating.

APS tracks failure counts for Windows Domain users in memory (there is no place in the user record to store it). If the user is locked out because of a maximum failure count, then the account may be disabled immediately when they make their next authentication attempt, subject to the following rules:

- If Auto Reset Failure Count is set, the user will not be allowed to authenticate until Max Failures On Change minutes have passed.
- Even if Auto Reset Failure Count is not in effect, the user will not be allowed to login for Failure Count Retention minutes (or 5 minutes, if no setting is specified).

There are two caveats here:

1. If the SiteMinder Authentication Service is restarted, the in-memory counters are lost.
2. If multiple SiteMinder Policy Servers exist, each keeps its own in-memory counter.

Special APS Processing/Limitations

APS is extremely limited in what it can do with users in a Windows Domain Directory because of the lack of any place to store control information in the user record.

- Disable Until, Disable After and Must Login By are not supported.
- Multiple disabling methods and reason codes are extremely limited (as described above).
- Generational Redirects are not supported.
- The external reset of failure counters is not supported.
- All of the historical data (total logins, last login, login history, et. al.) is not supported.
- Password History is not supported.
- Checking of new passwords against profile attributes is not supported.
- Password reuse checking is not supported.
- Case-insensitive passwords are not supported.
- Grace Periods and Grace Logins are not supported.
- Since APSExpire does not support Windows Domains, offline password and account expiration warnings are not supported.
- User-specific overrides for password and account expiration are not supported in the APS.cfg file. The actual password and account expiration dates in the user record are used instead.

Forgotten Password Services (FPS)

FPS does *not* support users stored in Windows Domain Directories.

APSExpire

APSExpire does not support users stored in Windows Domain Directories. This is generally not needed, Windows provides its own tools.

APSAdmin

APSAdmin does not support the maintenance of users stored in Windows Domain Directories. However, administrators (the online users) can use APSAdmin to administer other users.

Fault Tolerance & Performance

Fault tolerance is entirely implemented by the Windows Domain. APS has no special processing for these directories.

Known Directory Idiosyncrasies

APS supports Windows Domain Directories for backwards compatibility. However, due to the very nature of the directory mechanism, it is extremely limited with what it can do

ODBC (RDBMS) Directories

There is no "standard" way to update an ODBC schema. Each vendor implements their own way to do it. In addition, most sites have policies or requirements for how schemas are updated, naming conventions and data layout issues.

APS tries to place as few restrictions as possible for the schema that it needs. The attributes described in the first section of this chapter must exist for every user entry in the User Directory that will be maintained by APS, with the exception of the attributes marked as optional and suppressed in the [MAPPINGS] section of the APS Configuration File.

How sites accomplish this is entirely up to the administrators at the site. There are a number of choices available; no tools are provided to do this for you. CA Professional Services has considerable experience in this area and can help or advise you, if you so desire.

Schema Updates

The simplest way to organize the schema for an ODBC directory is to just add all of the non-multiple-valued attributes to the existing user table as columns, then add two new tables for the two multi-valued attributes.

However, this is not always possible due to naming policies and possible table size constraints.

Columns (attributes) need not have the names defined below. If the columns are to be renamed, the names must be remapped as described in the section starting on Unsupported "Page" Cross-Reference.

Unlike SiteMinder (which reads each column as a separate query), APS "wants" to read an entire row to retrieve the user. The contents of the row is cached so that it is available for settings overrides and mail replacement values. This means that the entire user row is retrieved. This may have both capacity and security implications.

This can be easily tuned by defining a stored *view* of the user that restricts APS to "seeing" only those columns that the site wishes APS to have access to. A site can use the query overrides in the [ODBC] section of the APS.cfg file to define APS-specific versions of these queries that use these stored views. The APS view of a user *can* be different from SiteMinder's view of the data.

Some sites have quite successfully stored the APS-specific information in a separate table from the rest of the user information. Those sites merely used a stored query to join the APS table with the user table to get the views that they want.

Attribute Length

Many of the attributes described below are listed as "variable length". In each case, the length listed is the maximum length used by APS itself. However, these columns typically contain a "comment" that can be placed on the information by a site. The comment can be of any length allowed by that site.

Some attributes, specifically smapsGenerationalRedirects and smapsHistory, can become extremely large, depending on site usage.

smapsGenerationalRedirects contains a list of generational redirect information. Normally, this is actually a relatively small (or non-existent) amount of information. However, if sites use a large number of generational redirects, the data storage requirements of this column can increase. A site should review its expected use of this feature to determine the amount of storage required.

The smapsPassword column contains encrypted data containing information about previously used passwords. APS places a hard restriction on the maximum length of this data. That hard limit is listed with the attribute description. However, the reality is that it would take an automated password changer a considerable amount of time to enlarge the data to that size.

Since the information is encrypted, it is impossible to just truncate it; it must be stored in its entirety.

To reserve the full amount of storage for every user row may be unreasonable. What many sites do is set the length of this column to some smaller value (for example 2k), then run triggers within the database to alert administrators when a certain percentage of this space starts to get used (say 90% - the field grows very slowly, usually less than 100 bytes per password change). If a single test user starts to consume too much, approaching the maximum length defined, the site clears the value for that one account. If multiple users start to approach the value, either determined by triggers or by periodic examination of the data, the column is enlarged.

Native Data Formats

APS does not support native ODBC data formats at this time. This is currently under review by the product team for a future enhancement.

Number of Rows

Two attributes are implemented as separate tables that will grow. The Login History table is maintained by APS and will be pruned as each user authenticates to prevent infinite growth. The FPS History table will grow indefinitely, records written each time a user runs FPS.

Special Setup (Queries)

Wherever possible, APS will use the queries defined in the SiteMinder ODBC Query Scheme associated with the User Directory. However, there are some additional queries that APS needs and there will be times that it is not appropriate for APS to use queries defined to SiteMinder.

In addition to the APS-specific queries, every query defined in the SiteMinder Query Scheme can be overridden for use by APS.

Each query has replacement parameters, or placeholders, embedded within them that indicate where APS (or SiteMinder) is to place values before executing the query. Each query will replace these parameters with specific values in the order defined by the query. Thus, the first parameter for a given query might be replaced by the User's ID. It is not possible to change the order of the parameters (the choice of placeholders and their order is defined by SiteMinder's Query Schemes). Placeholders are indicated in a query by "%s".

All of these queries are defined or overridden in the APS.cfg file in the [ODBC] section.

Enumerate

The Enumerate query overrides the query by the same name defined to SiteMinder. APS does not use this query at this time.

Get Object Info

The Get Object Info query overrides the query by the same name defined to SiteMinder. APS does not use this query at this time.

Lookup

The Lookup query overrides the query by the same name defined to SiteMinder. APS does not use this query at this time.

Init User

The Init User query overrides the query by the same name // defined to SiteMinder. APS does not use this query at this time.

Authenticate User

The Authenticate User query overrides the query by the same name defined to SiteMinder. APS uses this query to determine if the old password entered during a password change is valid. The default query is:

```
SELECT Name FROM SmUser
        WHERE Name='%s' AND Password='%s'
```

The first parameter is the user's name (entered to SiteMinder) and the second value is the (clear-text) old password as entered during the change password process.

This query can be a stored procedure, but the replacement parameters must retain their order and meaning. If the password is encrypted, then this query almost must be a stored procedure.

Get User Property

The Get User Property query overrides the query by the same name defined to SiteMinder. APS uses this query to retrieve the attribute values defined for the user.

The default query is:

```
    SELECT %s FROM SmUser WHERE Name='%s'
```

The first parameter is always replaced by an asterisk (retrieve all defined columns); the second parameter is the user's ID. If all columns should not be returned, this query should be overriden to reference a stored view in the database that returns fewer columns. Note that only columns returned on this query can be used in overrides or as macros in mail or redirections.

The first parameter is always replaced as a constant asterisk. A query could be defined with this, but then APS could not use the query defined to SiteMinder (which contains a replacement parameter in this position).

This query can be a stored procedure (if the underlying RDBMS supports rows returned from stored procedures), but the replacement parameters must retain their order and meaning.

Set User Property

The Set User Property query overrides the query by the same name defined to SiteMinder. APS uses this query to set attribute values for the user. The default query is:

```
UPDATE SmUser SET %s='%s' WHERE Name='%s'
```

The first parameter is the (mapped) name of the attribute (column) to modify, the second is the value to set it to. The third parameter is the user's name.

APS does special processing when using this query. If the query defined to APS (or SiteMinder) uses UPDATE, then APS will build a query to update *all* columns at once, using standard SQL syntax.

If, however, a stored procedure is used for this query, APS will call the stored procedure for each change. Parameters must appear in the same order for stored procedures.

The use of the UPDATE query is for higher performance.

If column access is to be restricted, create a stored VIEW in the database and use an UPDATE query to the stored view.

Get User Properties

The Get User Properties query overrides the query by the same name defined to SiteMinder. APS does not use this query at this time.

User Properties

The User Properties setting overrides the setting by the same name defined to SiteMinder. APS does not use this query at this time.

Lookup User

The Lookup User query overrides the query by the same name defined to SiteMinder. FPS and APSExpire use this query to locate users in the directory. The default query is:

```
SELECT Name, 'User' AS Class FROM SmUser WHERE %s
```

The parameter is the WHERE clause built up by FPS or APSExpire.

Get User Groups

The Get User Groups query overrides the query by the same name defined to SiteMinder. APS uses it to return the list of group in which the current user is a member. The default query is

```
SELECT SmGroup.Name
        FROM SmGroup, SmUser, SmUserGroup
        WHERE SmUser.Name='%s' AND
        SmUser.UserID=SmUserGroup.UserID
        AND
        SmGroup.GroupID=SmUserGroup.GroupID
```

The parameter is the user's name.

Each row returned represents a group name that the user is a member of.

Is Group Member

The Is Group Member query overrides the query by the same name defined to SiteMinder. APS uses it to determine if the user is in a specific group, both for internal purposes and to determine the result of IsInGroup() calls in an override expression. The default query is:

```
SELECT ID FROM SmUserGroup
                WHERE UserID=
                        (SELECT UserID FROM SmUser
        WHERE Name='%s')
                AND GroupID=
                        (SELECT GroupID FROM SmGroup
         WHERE Name='%s')
```

The first parameter is the user's name, the second is the name of the group of interest.

If the result of the query contains any rows or data, the user is considered a member of the group.

Set Password

The Set Password query overrides the query by the same name defined to SiteMinder. APS uses it to actually change the user's password. Typically, it is a stored procedure, but it need not be. The default query is:

```
    UPDATE SmUser SET Password='%s' WHERE Name='%s'
```

The first parameter is the new password, the second is the name of the user.

Note that the password may be encrypted, if SmAPSEx encrypted it.

This query can be a stored procedure (and should be), but the replacement parameters must retain their order and meaning.

Passwords will always be set using this query, never the Set User Properties query above.

Get Login History

Note: The Get Login History query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Login History is to be maintained.

Login History contains an entry for each login attempt by a user. Typically, it will be a separate table containing two fields, the history entry and the user's name.

The next three queries are also used to manipulate login history.

The query has one parameter that is the user's name.

The Get Login History query must return a single column with the login history entry. Its name is irrelevant. The entries should be returned in date order.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
SELECT smapsLoginHistory
                FROM LoginHistory
                WHERE Name='%s'
                ORDER BY smapsLoginHistory
```

Set Login History

Note: The Set Login History query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Login History is to be maintained.

Login History contains an entry for each login attempt by a user. Typically, it will be a separate table containing two fields, the history entry and the user's name.

The query has two parameters. The first is the Login History value and the second is the user's name.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
INSERT INTO LoginHistory (smapsLoginHistory, Name)
VALUES ('%s','%s')
```

Delete Login History

Note: The Delete Login History query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Login History is to be maintained.

Login History contains an entry for each login attempt by a user. Typically, it will be a separate table containing two fields, the history entry and the user's name.

Delete Login History query is used by APS to clean out old history values (before a specific date and time).

The query has two parameters. The first is the date and time to delete before and the second is the user's name.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
DELETE FROM LoginHistory
                  WHERE LEFT(smapsLoginHistory, 15)<'%s'
        AND Name='%s'
```

Clear Login History

Note: The Clear Login History query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Login History is to be maintained.

Login History contains an entry for each login attempt by a user. Typically, it will be a separate table containing two fields, the history entry and the user's name.

The Clear Login History query is used by APSAdmin to clean out all of the login history for a specific user.

The query has a single parameter used to identify the user.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
    DELETE FROM LoginHistory WHERE Name='%s'
```

Get FPS Log

Note: The Get FPS Log query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if FPS Logging is to be maintained.

The FPS Log contains an entry for each FPS usage attempt by a user. Typically, it will be a separate table containing two fields, the log entry and the user's name.

The next two queries defined are also used to manipulate the FPS Log.

The query has one parameter that is the user's name.

The Get FPS Log query must return a single column with the log entry. Its name is irrelevant. The entries should be returned in date order.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
SELECT smfpsLog FROM FPSHistory
                        WHERE Name='%s'
                        ORDER BY smfpsLog
```

Add FPS Log

Note: The Add FPS Log query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if FPS Logging is to be maintained.

The FPS Log contains an entry for each FPS usage attempt by a user. Typically, it will be a separate table containing two fields, the log entry and the user's name.

The query has two parameters. The first is the FPS Log value and the second is the user's name.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
    INSERT INTO FPSHistory (smfpsLog, Name) VALUES ('%s','%s')
```

Clear FPS Log

Note: The Clear FPS Log query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if FPS Logging is to be maintained.

The FPS Log contains an entry for each FPS usage attempt by a user. Typically, it will be a separate table containing two fields, the log entry and the user's name.

The Clear FPS Log query is used by APSAdmin to clean out all of the FPS Log entries for a specific user.

The query has a single parameter used to identify the user.

The actual names of the table and columns are defined by the query and do not matter to APS.

An example query might be:

```
DELETE FROM FPSHistory WHERE Name='%s'
```

Set Password Checksum

Note: The Set Password Checksum query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Auto Force Change functionality is required.

The Auto Force Change keyword in this file tells APS to check for password changes outside of APS and, if detected, force the user to change their password at next login. It is used to automatically treat external (administrative) password changes as Force Immediate Change situations without requiring changes to the administration utility.

Under LDAP, APS uses the smapsPassword attribute to handle this functionality. Under ODBC, this is not necessarily possible. Databases are typically protected so that passwords cannot be read back.

If Auto Force Change is to be used, this and the following query must be defined. They are almost always stored procedures.

The query has a single parameter used to identify the user (the user's password has already been changed).

Typically, the implementation of this functionality uses some special attribute to store the checksum (or the entire password value, for that matter).

An example query might be:

```
CALL SetPasswordChecksum('%s')
```

Test Password Checksum

Note: The Test Password Checksum query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if Auto Force Change functionality is required.

The Auto Force Change keyword in this file tells APS to check for password changes outside of APS and, if detected, force the user to change their password at next login. It is used to automatically treat external (administrative) password changes as Force Immediate Change situations without requiring changes to the administration utility.

Under LDAP, APS uses the smapsPassword attribute to handle this functionality. Under ODBC, this is not necessarily possible. Databases are typically protected so that passwords cannot be read back.

If Auto Force Change is to be used, this and the previous query must be defined. They are almost always stored procedures.

The query has a single parameter used to identify the user. The function should return a numeric or boolean value, where non-zero indicates that the checksum is invalid.

Typically, the implementation of this functionality uses some special attribute to store the checksum (or the entire password value, for that matter).

An example query might be:

```
?=CALL TestPasswordChecksum('%s')
```

Compare FPS Answer

Note: The Compare FPS Answer query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if FPSí **ODBC Encrypt** functionality is required.

The answers to FPS questions can be encrypted in an ODBC database. This is indicated to APS using the ODBC Encrypt keyword in the [FPS-Verify] section of the APS.cfg file. This query is one way that sites can implement this encryption.

There is no default for this query. If not defined and ODBC Encrypt was specified in APS.cfg, FPS will generate an error, since it won't know how to compare an encrypted answer (this is assuming that the encryption is not being performed by SmAPSEx).

The query must be a stored procedure that takes three arguments (or, at least, three substitution parameters) and return a numeric or boolean value (non-zero indicates that the compare is true).

The first parameter is the user name. The second parameter is the name of the attribute, as configured to APS. The third parameter is the user-entered answer (in clear text).

The implementation of this function usually encrypts (or hashes) the user supplied value (the third parameter) and compares it to the value stored in the user entry.

An example query might be:

```
?=CALL CompareFPSAnswer('%s', '%s', '%s')
```

Add To Group

Note: The Add To Group query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if the APSAdmin API is to be used to maintain group memberships.

The APSAdmin API functions use this query to add users to an existing group (creation of groups is not supported).

There is no default query for this purpose. If not specified and a user must be added to a group, an error will be logged and the update will fail.

An example query might be:

```
INSERT INTO SmUserGroup (UserID, GroupID) VALUES ('%s', '%s')
```

Remove From Group

Note: The Remove From Group query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if the APSAdmin API is to be used to maintain group memberships.

The APSAdmin API functions use this query to remove users from an existing group.

There is no default query for this purpose. If not specified and a user must be removed from a group, an error will be logged and the update will fail.

An example query might be:

```
DELETE FROM SmUserGroup WHERE UserID='%s' AND GroupID='%s'
```

Admin Translation

Note: The Admin Translation query does not exist in the SiteMinder Query Scheme and has no default. Therefore, a query must be defined if the APSAdmin API is to be used in certain scenarios.

The APSAdmin APS functions use this query to translate input user identity (on the APSAdmin user selection form) to a user name that can be used on future queries.

The query is expected to return a single record with at least one column that is the user's name (the one used in all of the other queries).

If not specified, then the query is not used and the data entered is expected to be the user's name. If multiple records or no records are returned, a "User record could not be found" error is displayed to the administrator. If multiple records are returned, an error is issued to the console log as well.

This query is typically used if the administrator is expected to identify users to APSAdmin using something other than userid, such as membership number.

An example query might be:

```
SELECT Name FROM Users WHERE MemberNumber='%s'
```

Disabling/Enabling User Account

If the account satisfies the conditions for any Ignore keyword in the APS.cfg file, APS will not perform any authentication time processing for this user, including detecting if the account is disabled or detecting disabling conditions (such as password expired).

Recognizing a Disabled Account

APS will *recognize* that an account is disabled if any of the following are true (this is not necessarily the order in which APS actually does detection):

- If the account satisfies any of the conditions defined in APS.cfg using the Disable setting, it will be considered disabled. The keyword defines the Disabled Reason code.
- If the smapsDisabledUntil field starts with the word "FOREVER", the account will be considered disabled. The rest of the text (after "FOREVER") in the attribute will be used as the Disabled Reason code.
- If the smapsDisabledUntil field contains a date/time that is in the future, the account will be considered disabled. The rest of the text (after the date/time) in the attribute will be used as the Disabled Reason code.
- If the smapsDisabledAfter field contains a date/time that is in the past, the account will be considered disabled. The rest of the text (after the date/time) in the attribute will be used as the Disabled Reason code.
- If the smapsMustLoginBy field contains a date/time that is in the past, the account will be considered disabled. The rest of the text (after the date/time) in the attribute will be used as the Disabled Reason code.
- The account may *become* disabled (as described in the next section) during authentication, but this will set one of the above conditions.

Disabling a User Account

APS itself can disable an account for only four reasons (APS will never set the native account status bits):

- Exceeded Failure Count. This can only occur during the authentication process.

  If Auto Reset Failure Count setting is in effect:

  smapsDisabledUntil is set to the current date and time, plus the number of minutes indicated by the Max Failures On Change setting followed by the text "Failure Count".
- Otherwise:

  smapsDisabledUntil is set to "FOREVER Failure Count".
- Password Expired (any grace period expired and all grace logins expended). This can be done either at authentication time *or* by APSExpire.
- smapsDisabledUntil is set to "FOREVER Password Expired".
- Account Expired due to inactivity. This can be done either at authentication time or by APSExpire.
- smapsDisabledUntil is set to "FOREVER User Expired".
- Locked out from FPS misuse. This can only be done by the FPS process.
- smapsDisabledUntil is set to "FOREVER FPS Lockout".

Enabling a User Account

APS never enables an account. Some of the mechanisms used by APS to maintain the user status have built-in reset capabilities (dates). APS does not actually update a user record to re-enable it.

To re-enable an account, a site must ensure that all of the above criteria that APS uses to detect that it is already disabled are not true. In other words, the account cannot be a member of a disabled group, smapsDisableUntil must be set correctly, etc.

Even then, it may appear that an account remains disabled. The usual cause is that while the account does get enabled, the conditions that caused it to become disabled remain, thus causing APS to just disable it again. These reasons are:

- The password remains expired. Either set smapsBaseDate (recommended) or smapsLastPasswordChange (not recommended) to a more recent value.
- The account remains expired. Either set smapsBaseDate (recommended) or smapsLastLogin (not recommended) to a more recent value.
- smapsFailureCount shows that the account remains locked out (see the description of the field at the beginning of this chapter for details on how to reset this field - it is not sufficient to merely delete its value).

Special APS Processing/Limitations

APS does not support native ODBC (RDBMS) data formats, such as dates and binary formats. However, some sites have quite successfully used temporary tables and triggers to convert the APS maintained data into and out of native formats. CA Professional Services has some experience in this area and may be able to help, if this is required.

The Auto Force Change setting makes no sense for ODBC directories and is not support.

Disabled groups are not supported.

Forgotten Password Services (FPS)

FPS fully supports ODBC (RDBMS) directories. There is special processing for handling encrypted FPS answers (See ODBC Encrypt.).

APSExpire

APSExpire has special settings for ODBC directories so that the user directory can be partitioned into smaller "chunks" for processing. See the chapter entitled Daily Processing (APSExpire) for details.

APSAdmin

APSAdmin fully supports ODBC User Directories. Using the Admin Translation (see page 248) query, users can be selected using arbitrary identification.

Fault Tolerance & Performance

APS does a few things that significantly improve performance and fault tolerance when dealing with ODBC User Directories. Note that if improperly configured, performance can actually be *worse*; some fine-tuning is always desirable.

- APS ignores the list of user columns (properties) defined in the query scheme. It will, instead, attempt to read all of the columns in the user view (using "*" for the column selection). APS will then cache the entire row (for the duration of the request only). This means that any columns used by email templates or settins overrides will be available to APS without performing another user query. It also simplifies updates to APS configuration in that a reference to a new column by an override or email template need not be accompanied by an update to the list of available columns.

However, this may come at a cost. If there are a large number of columns in the user table that APS should not access, they will be returned anyway. In this case, it may be desirable to use a stored view instead of accessing the raw table (this is generally considered good practice for SQL databases anyway).

- When updating the underling directory, APS looks at the Set User Property query. If it looks like a standard SQL update (as opposed to a stored procedure), an SQL UPDATE statement will be used to perform ALL of the updates on a single query (except for certain special updates, such as the actual password). This reduces network round trips to the directory server.
- APS turns off auto-commit processing for the directory and performs its own COMMIT and ROLLBACK operations. Thus, if multiple updates need to be performed, such as password change and related data, they will either all be committed together or rolled back.

Known Directory Idiosyncrasies

Stored procedures are extremely difficult to implement and debug, since they are so implementation-dependent. APS uses the "standard" ODBC mechanism for invoking stored procedures. Please see the documentation for the directory vendor for any special formatting or escapes required to implement stored procedures or embedded function calls within queries. CA cannot provide support for writing these queries, as ever vendor differs in their implementation.

However, CA Professional Services may be contracted to help a site develop and troubleshoot such queries.Some ODBC implementations limit the number of characters allowed for a column name (or the number of significant characters). These implementations may require that the default attribute names be remapped to shorter names.

Password Replication/Synchronization

APS, out of the box, does not support password replication or synchronization between multiple user directories.

This is not a simple subject, but deserves some attention in this document.

There are many technical problems with password replication. Here are a few of the bigger examples:

- Fault tolerance. What happens if one of the directories involved is not available when a password changes?
- Compatibility. What happens if a given password is *allowed* by the first directory, but not allowed by a subsequent directory (due to native password policies)? How does the process "roll back" the change already made on the first directory?

- Coverage. Unless implemented at the directory level, *every* application that changes user passwords must trigger the replication.
- Standards. Most directories do not implement a "Is This Password Valid" function so that applications can verify a password *before* saving it.
- Network Security. Passwords will have to be passed around, multiple times, on the network. Often in clear text, depending on directory support.
- Application Security. The password will only be as secure as the least secure data storage mechanism. Imagine an LDAP database, which usually stores passwords in hashed format: fairly secure. Now a new application using Microsoft Access is added and all password changes to the LDAP directory are replicated to the Access Database, which might store it in clear text. Now all of the applications at the site are compromised. Even if the Access application deems its security as sufficient for its own purposes, it may not be for the other applications at the site.

There are a number of reasons that sites consider password replication/synchronization. Some are better implemented in other ways, thus avoiding many of the problems listed above.

Shared Directories Some people are under under the impression that "Password Synchronization" must occur between "applications", even if the "applications" share a common user directory. This is actually a trivial case, since the credentials are only stored once and can (usually, but not always) be shared between the two applications (presumably, one of the applications is SiteMinder). With SiteMinder's broad User Directory support, this case of "synchronization" is non-existent (or trivial, depending on how you look at it). This, of course, depends on whether the applications require their own operational user attributes or can share the "native" ones.

- Hidden Directories—There are many cases where SiteMinder is protecting a web application that must, in turn, log into an underlying system, with its own (unsharable) User Directory. However, the user's interaction with that underlying system is exclusively through SiteMinder. Many jump to the conclusion that this requires synchronization; that the credentials passed to SiteMinder need to be used to log into the underlying application and therefore the credentials always need to be kept in sync.

  If the underlying system cannot be accessed through any other mechanism than SiteMinder, this synch is *not* needed, in fact. An alternate solution for this case is presented in the chapter on Best Practices for Storing Legacy/Back-End Credentials (see page 423).
- Single Sign On to the Desktop —The Site wants users to have a single set of credentials for both the desktop and SiteMinder.

  Since SiteMinder supports both Windows Domain Directories and Active Directory, this is better accomplished using a *shared* directory than by replication. iPlanet (and possibly others) also has a replicator available to perform password synchronization at the directory level.

- Authentication vs. Data Directories—Some sites look at synchronization as a necessity when they have multiple data stores, all protected by SiteMinder, the data stores *cannot* be merged (for whatever reason), but they want to pull different data out of different directories for different applications. The example is a customer that defines two different User Directories, but the users are the same in each. Directory A is used for Application A policies and Directory B is used for Application B policies. They do this so that they can make *authorization* decisions or provide responses to resources from the correct directory.

  This case is trivially solved using SiteMinder's Auth/AZ mapping. In fact, without such mapping, no single sign-on is possible, since when the user will authenticates against different directories, SiteMinder will consider them *different* users.

- Multiple Directories—There are real cases where two (or more) user stores are used as User Directories by different applications and all of the applications are accessible through their own interfaces (not through common access control, such as SiteMinder).

  This case only occurs when two (or more) data stores are accessed independently, directly by the user and both are used for authentication.

  In these cases, sites cannot get Single Sign-On (because of the different access controls), but they want each user to have a single set of credentials that they can use to log into each application. When the credentials change in one application, they should change in all others. Things like forced password change, password expiration, etc, should be enforced according to common (or at least negotiated) rules.

  This is the only true Password Synchronization case and is actually quite rare. All of the other cases are better handled using other mechanisms.

  Note: The two directories involved are often otherwise unrelated and no other provisioning is required.

  Such replication is, however, still possible, even though it is not supported out of the box by APS. APS calls a specific function *every time* a user's password changes, passing the user identity and the new password. This function can be hooked to capture the change, and then the function can write the changes to any or all of the other places that it is needed.

  This function is part of the SmAPSEx library (described on Unsupported "Page" Cross-Reference), but is not publicly available in the SDK (imagine the possibility of Trojan horses capturing password changes). However, it is available to CA Professional Services to provide custom solutions.

Chapter 5: Administration and Operations

This section contains the following topics:

Introduction (see page 255)
APS Processing during User Authentication (see page 256)
Password Lifetime (see page 260)
Backup Policy Servers (see page 263)
Special Case: Three Strikes, You're Out (see page 263)
SiteMinder's [Basic] Password Services (see page 265)
Using Persistent Cookies (see page 266)
Authorization Mapping and AZRedirect (see page 267)
FPS Process Flow (see page 267)

Introduction

Advanced Password Services (APS) runs in conjunction with the SiteMinder Policy Server. After SiteMinder has attempted each authentication of a user against a directory, APS is consulted to verify the authentication. Each login attempt, successful or failed, is presented to APS behind the scenes and APS is asked to verify that the user is valid and not disabled. Advanced Password Services checks its own records of that user's activity, including the number of failed login attempts that user has accumulated and the length of time since the user last successfully authenticated. APS then checks whether the user has been disabled, finally updates its records and either allows an authenticated user access, or halts a successful authentication, rejecting the login and (possibly) disabling the user's account.

Advanced Password Services also allows users to change their own passwords through the Change Password Interface, which can be customized for your organization, or the API, which can be called by your own or third party tools. Users must be configured through the SiteMinder Policy Server to have access to the Change Password Interface, but in most circumstances, this will not require periodic maintenance.

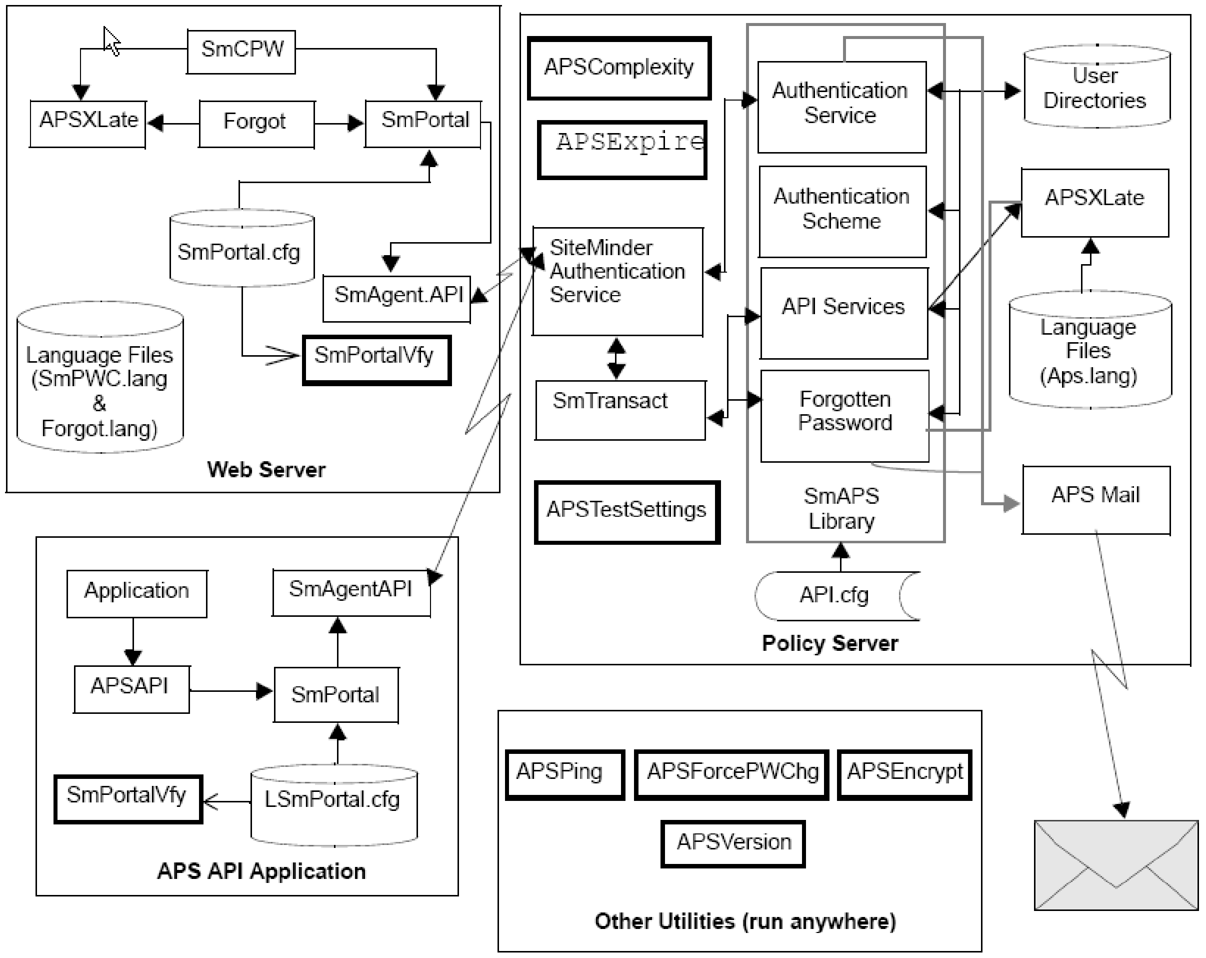
Password content policies apply only to passwords changed through the Change Password Interface and the API. They are not applied to existing passwords nor password changes made using other interfaces (such as Windows NT or iPlanet's Directory Server).

APS includes Forgotten Password Services (FPS), which can allow users to interactively recover forgotten passwords or user ids in as secure a manner as possible.

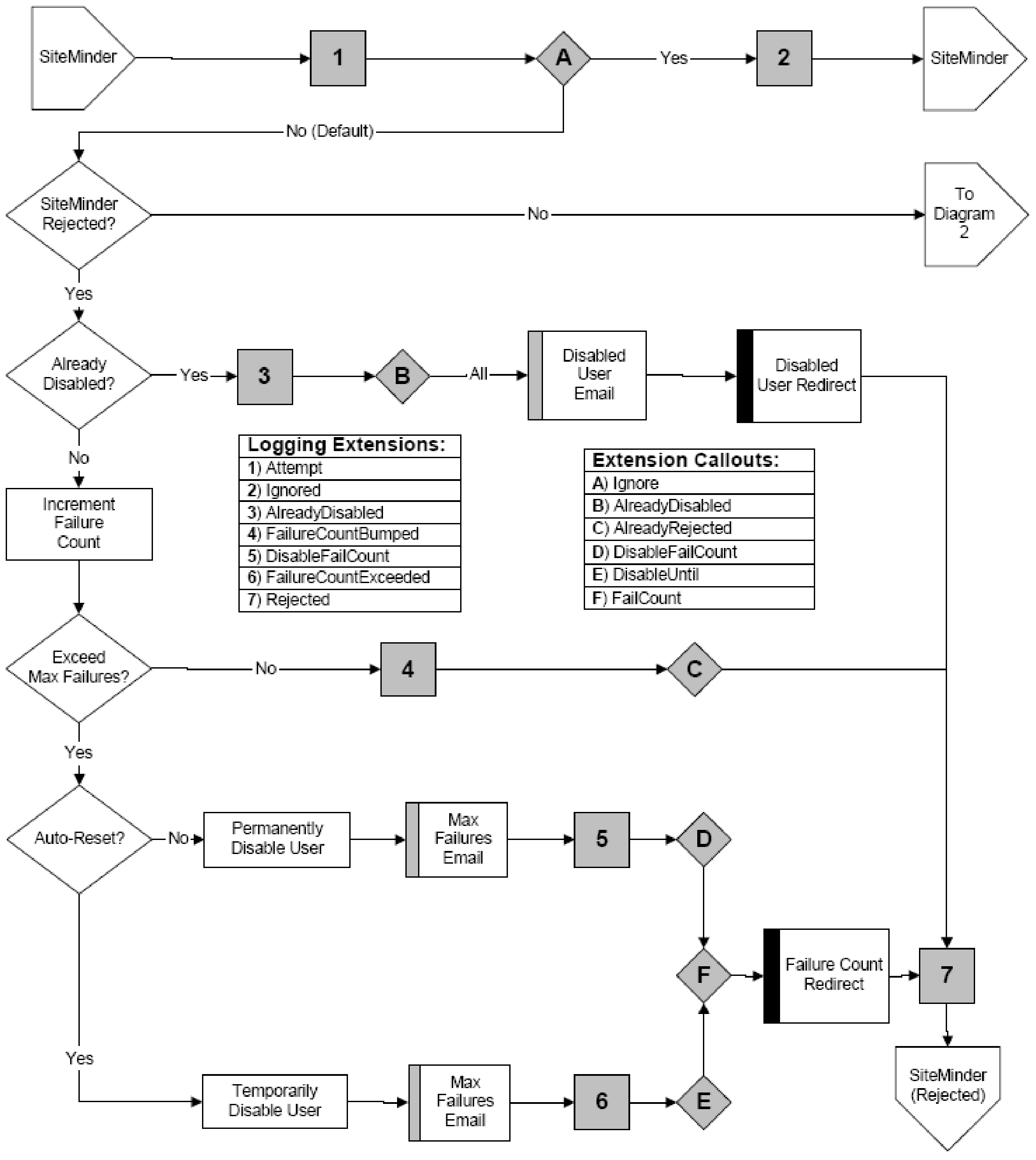
There is also a Help Desk interface (not shown in diagram above) called APSAdmin, which allows sites to quickly set up simple user account enable/disable/password reset operations.

APS Processing during User Authentication

The following graphic shows the processing that occurs when SiteMinder calls APS during User Authentication. SiteMinder passes the result of *its* authentication to APS (success or failure). It shows what APS does if SiteMinder has already rejected the login (almost always because the user's password is bad):



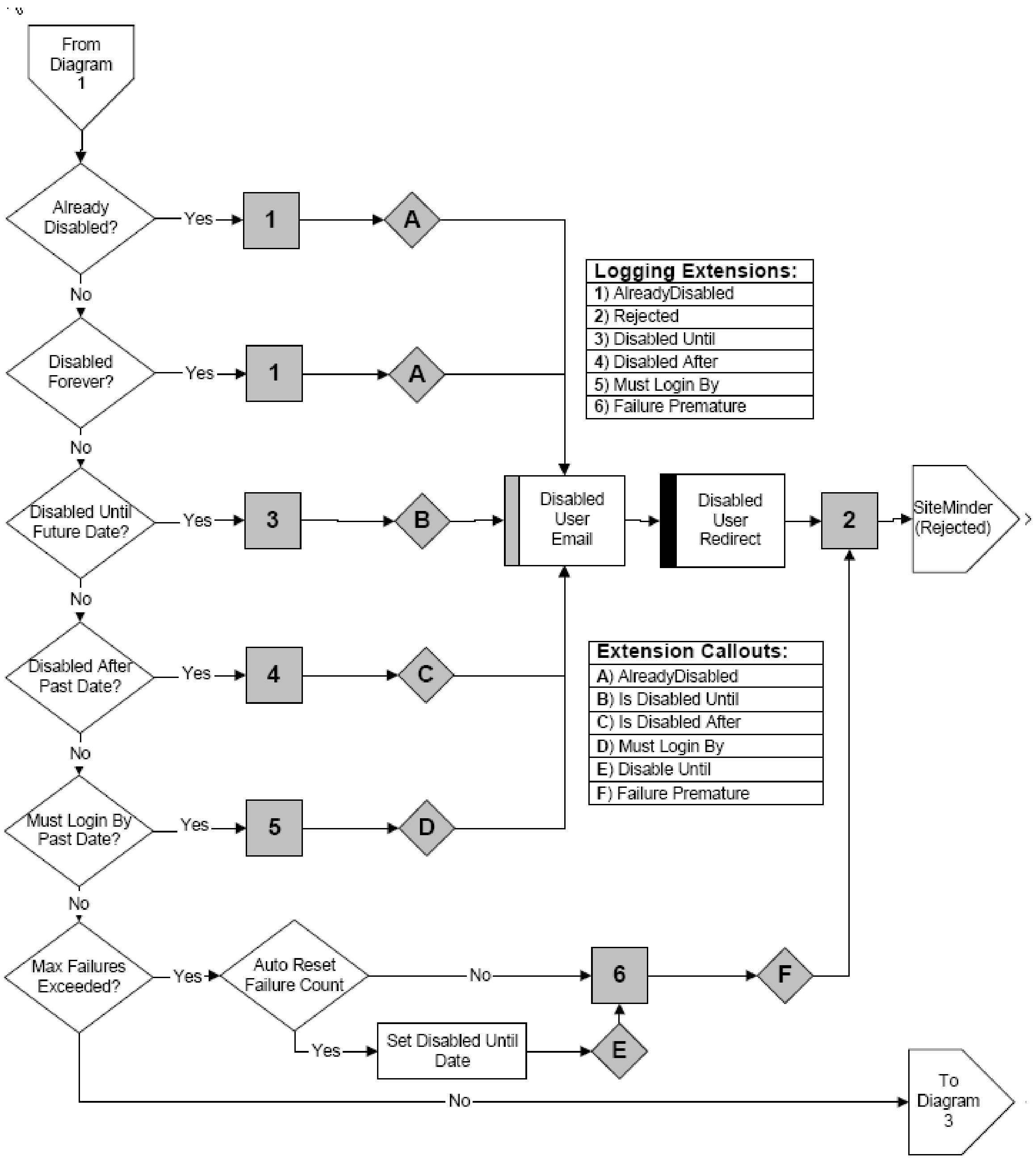
The process description continues in the following graphic to explain what APS processing occurs when SiteMinder has indicated a successful authentication (at least until the APS invocation):



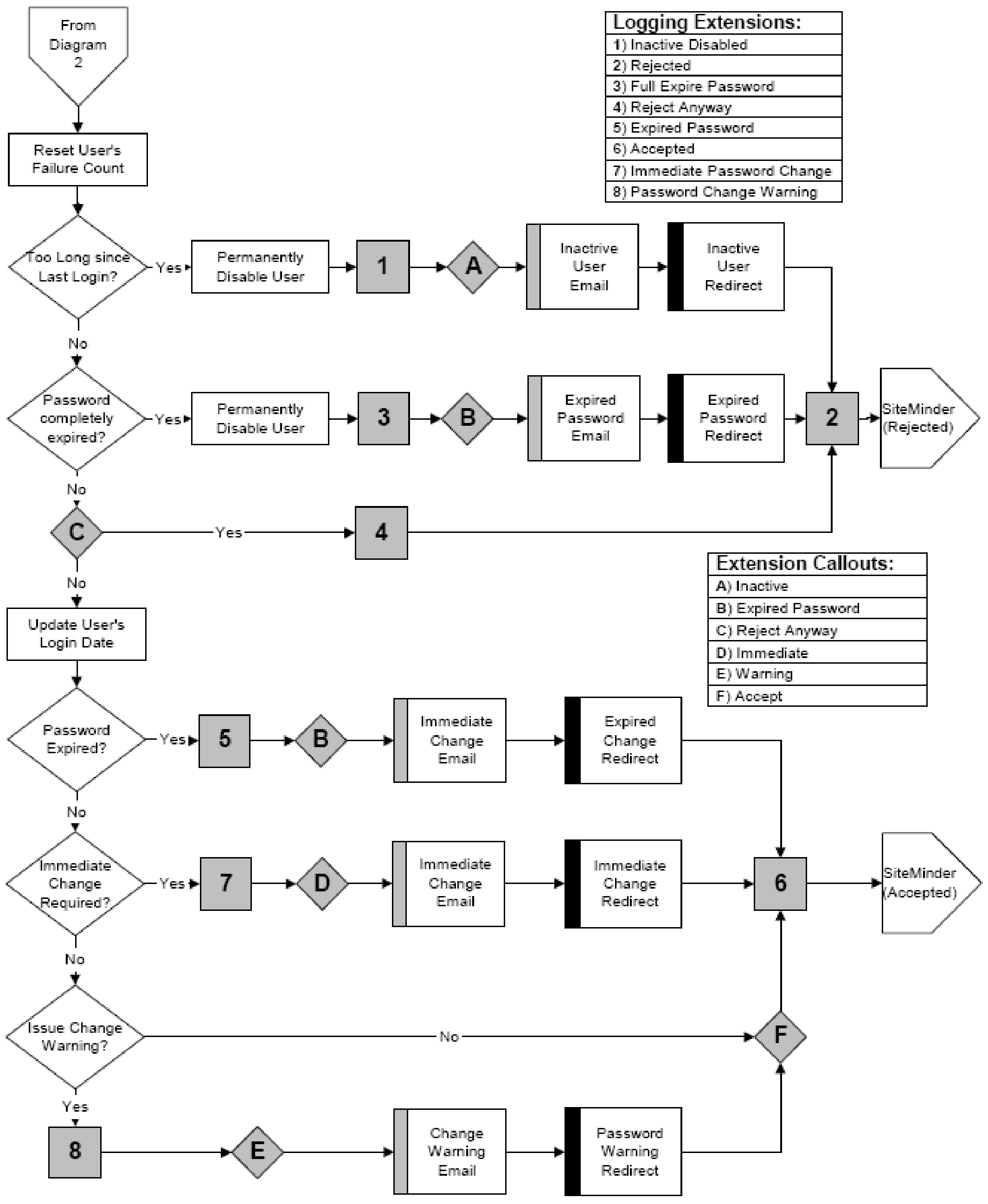
Logging Extensions:
1) Attempt
2) Ignored
3) AlreadyDisabled
4) FailureCountBumped
5) DisableFailCount
6) FailureCountExceeded
7) Rejected

Extension Callouts:
A) Ignore
B) AlreadyDisabled
C) AlreadyRejected
D) DisableFailCount
E) DisableUntil
F) FailCount

The following graphic shows how APS checks whether a user is disabled. Since SiteMinder does not "know" about APS disabling, it has already approved the authentication. If the user has already been approved by SiteMinder, APS must check the disabled status first:



If the user passes all of these tests, then conditions like user inactivity and password expiration must be checked, as shown in the following graphic:

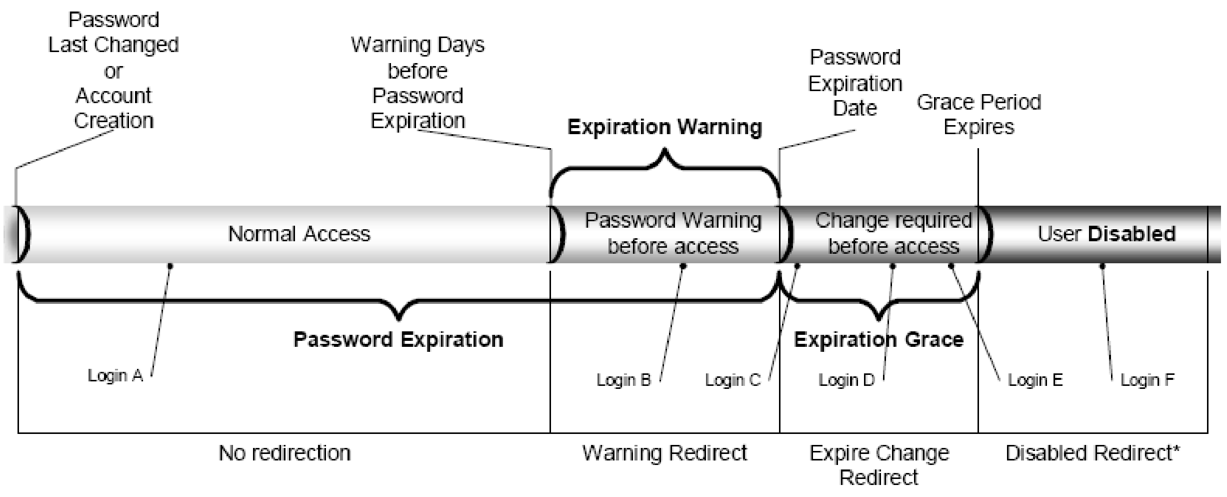


Logging Extensions:
1) Inactive Disabled
2) Rejected
3) Full Expire Password
4) Reject Anyway
5) Expired Password
6) Accepted
7) Immediate Password Change
8) Password Change Warning

Extension Callouts:
A) Inactive
B) Expired Password
C) Reject Anyway
D) Immediate
E) Warning
F) Accept

Password Lifetime

The following graphic demonstrates how certain APS settings affect the lifetime of a password.



There are three settings which affect password lifetime.

- Password Expiration controls the maximum amount of time that can pass between password changes (or since the user account was created).

  If Password Expiration is not set (or zero), the user's password has an infinite lifetime (never expires).

  Password Expiration can be overridden at the user level using the smapsExpirePasswordDays attribute in an LDAP entry.

- Expiration Warning controls how long before actual expiration we start to warn the user that their password will *expi*re.

  If Expiration Warning is zero or not set, no password warnings will be issued.

- Expiration Grace controls the amount of time *after* the password has actually expired that we will allow the user to still login, though the user will have to change their password before accessing the system.

  Once the Expiration Grace period has expired without a password change, the user will be disabled when they login.

  If Expiration Grace is not set or zero, the user account will be disabled immediately upon password expiration (subject to the Grace Logins setting, as described below).

A fourth setting, Grace Logins, can also affect this process.

The following tables describe what happens at each of the "Login Points" shown in the diagram.

Note that APS may redirect the user or take other actions that have nothing or little to do with password lifetimes (i.e. Force Password Change, User Expiration). These actions are not reflected in the tables below.

Also, if a particular redirect is undefined, APS will not perform the redirect and may ignore the lifetime setting (e.g. if there is no Expire Change Redirect, then APS will ignore the Expiration Grace period).

Expiration Grace *set*
no* Grace Login *Setting

Login	Action
A	Normal login. APS does not redirect.
B	User will be redirected to the Warning Redirect setting.
C	User will be redirected to the Expire Change Redirect page. If AZRedirect is configured, user cannot access site without changing password.
D	User will be redirected to the Expire Change Redirect page. If AZRedirect is configured, user cannot access site without changing password.
E	User will be redirected to the Expire Change Redirect page. If AZRedirect is configured, user cannot access site without changing password.
F	User will be redirected to the Disabled Redirect page. If the Reset Password setting is in effect, further attempts will be rejected without any APS redirect (bad credentials).

Expiration Grace *set*
Grace Login *set to 3*

Login	Action
A	Normal login. APS does not redirect.
B	User will be redirected to the Warning Redirect setting.
C	User will be redirected to the Expire Change Redirect page. Even if AZRedirect is configured, user *will* be allowed to access the site without changing password, since this is *not* the last Grace Login.
D	User will be redirected to the Expire Change Redirect page. Even if AZRedirect is configured, user *will* be allowed to access the site without changing password, since this is *not* the last Grace Login.
E	User will be redirected to the Expire Change Redirect page. Even if AZRedirect is configured, user *will not* be allowed to access the site without changing password (since this is the last allowed Grace Login)

F	The user will be disabled and redirected to the Disabled Redirect page. If the Reset Password setting is in effect, further attempts will be rejected without any APS redirect (bad credentials).

Expiration Grace *set*
Grace Login *set to 4*

Login	**Action**
A	Normal login. APS does not redirect.
B	User will be redirected to the Warning Redirect setting.
C	User will be redirected to the Expire Change Redirect page. Even if AZRedirect is configured, user *will* be allowed to access the site without changing password, since this is *not* the last Grace Login.
D	User will be redirected to the Expire Change Redirect page. Even if AZRedirect is configured, user *will* be allowed to access the site without changing password, since this is *not* the last Grace Login.
E	User will be redirected to the Expire Change Redirect page. Even if AZRedirect is configured, user *will* be allowed to access the site without changing password, since this is *not* the last Grace Login.
F	User will be disabled and redirected to the Disabled Redirect page. If the Reset Password setting is in effect, further attempts will be rejected without any APS redirect (bad credentials). Note that even though the user has another Grace Login remaining, the Expiration Grace period has expired, so the user will be disabled.

Expiration Grace *NOT set (or zero)*
Grace Login *set to 3*

Login	**Action**
A	Normal login. APS does not redirect.
B	User will be redirected to the Warning Redirect setting.
C	User will be redirected to the Expire Change Redirect page even though there is no Expiration Grace (since there is a Grace Login defined). Even if AZRedirect is configured, user *will* be allowed to access the site without changing password, since this is *not* the last Grace Login.
D	User will be redirected to the Expire Change Redirect page even though there is no Expiration Grace (since there is a Grace Login defined). Even if AZRedirect is configured, user *will* be allowed to access the site without changing password, since this is *not* the last Grace Login.

E	User will be redirected to the Expire Change Redirect page. Even if AZRedirect is configured, user *will not* be allowed to access the site without changing password (since this is the last allowed Grace Login)
F	On the FOURTH authentication attempt, the user will be disabled and redirected to the Disabled Redirect page. If the Reset Password setting is in effect, further attempts will be rejected without any APS redirect (bad credentials). Note that since the password has expired, there is no Expiration Grace and all Grace Logins are used, the user will be disabled.

Backup Policy Servers

Advanced Password Services will operate simultaneously on any number of Web Servers using a single set of SiteMinder Policy Servers. In case of a failure of the SiteMinder server, APS can switch over to secondary servers in the same way that a Web Agent will. Primary and backup Policy Servers are configured using the SmPortal.cfg file (see the SmPortal/SmTransact Administration Guide). They can be configured either in failover or in round-robin mode.

Each SiteMinder Policy Server must have Advanced Password Services installed. Each such server is configured separately and independently. It is important that the configuration of all such machines be identical, though APS does not enforce this. There are some cases where they should differ between machines, but this always involves specific performance tuning when the Policy Servers are geographically dispersed.

See the description of the Max Failures setting for details about how multiple Policy Serves impact that functionality.

Special Case: Three Strikes, You're Out

SiteMinder, with APS installed, supports *two* different "Three Strikes, You're Out" options. These options are basically unrelated and can be used together, if care is taken with the configuration. It is fairly typical for a site to confuse users (or themselves) by improperly configuring the two options.

APS keeps a failure count on a per user basis. This failure count is kept in memory **and** written to disk (for LDAP users). The information is kept both on disk and in memory for LDAP users, so that the feature will still work even if the Primary LDAP server is unavailable. APS will use the setting in memory or the setting on disk, whichever has a later date/time. Each failure count is maintained with a timer (controlled by the Failure Count Timeout setting). If no attempts are made for this amount of time, the failure count is discarded. The count includes all attempts, regardless of which browser supplied the authentication credentials, but only those that apply to the specific user.

Most implementations of HTML form-based login keep a *retry count*. In the form login mechanisms supplied with SiteMinder, this count is kept in a transient cookie called RETRYNO, SMRETRYNO, TRYNO or SMTRYNO. This count is kept for the duration of the browser session. It includes all failures, *regardless* of the user id entered. Usually, when this value hits a threshold, further attempts are prevented and an error message (or page) is displayed.

Please note that Basic Authentication (where the browser presents a dialog box for authentication) works the same way, but the count is not configurable; it is always three. In this case, it is consecutive retries, but, like above, it is independent of the user id entered.

The counters kept at the browser level should only be considered "speed bumps". A savvy user knows that he need only stop, and then restart the browser and further attempts can be made.

The APS tracking will, eventually, *lock down* (disable) the user. Regardless of what the user does, the account is no longer available.

Some sites will use one capability or the other. For the most security, with the minimum user distress, both should be used with care, as described below:

The "speed bump" (browser-side) options should be set to a value of three (note that Basic Authentication is not configurable). The APS Max Failures setting should be set to 5.

At the end of three attempts, send the user to an error page that reads something like "You've apparently forgotten your user id or password. Please call our Help Desk for assistance." or refer the user to your Forgotten Password Support page.

A normal user will typically do exactly what is asked; they will call the help desk for assistance. The account is not disabled, since the APS count will only be 3.

If, however, the user is intent on breaking into the site, they will stop the browser, then restart it and continue to attempt a break-in. After two more attempts, APS will intercede and the account will be disabled. Further attempts will be denied. This will only happen in two cases:

- The user shut down the browser and explicitly continued to make attempts.
- The user is accessing through multiple browsers.

In either case, it is a real attempt at intrusion, not just a mistaken user.

APS should *not* be configured to redirect the user in this case (nor in the disabled case). Instead, an email should be sent to an internal security administrator, possibly through a pager gateway, telling the administrator that an intrusion attempt is in progress. If possible, email should be sent to the owner of the user id as well, describing that the account is disabled and why.

The "speed bump" will continue to operate, so the hacker will never know that the account has been disabled.

SiteMinder's [Basic] Password Services

Starting with Version 4.1, SiteMinder included Password Services (often referred to as *Basic* Password Services). The functionality included is essentially a subset (with a few extensions) of the Advanced Password Services Version 1.1 functionality.

In the process, CA modified SiteMinder so that Password Services can be enabled or disabled on a per-authentication scheme basis. This is done using a checkbox on the authentication scheme setup pages in the Policy Management interface. If this flag is turned *off* for a given authentication scheme, neither Password Services *nor* Advanced Password Services will be invoked. It is critical for Advanced Password Services operation that this flag be turned *on* for all authentication schemes (there are probably reasons that it might be turned off in special cases, but we haven't seen any yet).

Password Services and Advanced Password Services, as of this writing, are not compatible. You can use only one or the other, not both. In fact, at this time, there is no way to convert from one to the other (though such conversion could be written; contact CA Professional Services if this is required).

Because of this, when APS is used, all Password Services settings (except the authentication scheme setting above) must be disabled. If not, unpredictable results may occur.

Using Persistent Cookies

SiteMinder Web Agents can be configured to use *persistent cookies*. When this option is turned on, the user authenticates *once* from a particular machine/browser and a permanent record of that authentication is stored in a cookie. Whenever the user accesses the site from the same machine/browser, the user is not authenticated again (though the user is re-validated to ensure that the cookie is still valid).

APS is not involved in the validation process, only in the authentication process. If persistent cookies are to be used, the following functionality of APS will not work properly, since the user's last login date will not be recorded (last login implies authentication). None of these options can be used with persistent cookies.

Account Expiration

Since users don't authenticate, there is no last login date upon which to base the calculation.

Account Inactivity Warning

Since users don't authenticate, there is no last login date upon which to base the calculation.

Password Expiration Warnings

The detection of this situation occurs during the authentication process, which is not invoked.

True password expiration

The detection of this situation occurs during the authentication process, which is not invoked. Users will not be disabled at the end of their grace period.

APS is not invoked during user validation (which occurs when a persistent cookie is presented), only during authentication. Thus, the above functionality is not operable.

However, Force Change Password and Expired Password (as during the grace period) can still be handled using the AZRedirect capability, since AZRedirect is invoked in all cases. If the site does not use AZRedirect, then APS functionality will only occur during the initial authentication (when the persistent cookie is created) and will never be invoked again.

Of course, voluntary password changes will work correctly, since it does not involve the authentication process.

Authorization Mapping and AZRedirect

Some strange things can happen when using Authorization (AZ) mapping and AZRedirect in the same Policy Domain. The problem arises because the *authenticating* user is checked during the authentication process, but the *authorizing* user is used during AZRedirect. The problem occurs when the two users are **not** the same.

APS checks the Force Password Change flag, Generational Redirects, and several other user settings both at authentication time and during AZRedirect. If the two users are not the same, APS will not be looking at the same set of flags.

In addition, the change password process needs to update the *authenticating* user rather than the *authorizing* user. Thus, you should **never** perform AZ mapping within the change password domain. This also means that the Immediate Password Change setting will only be reset in the *authenticating* user.

FPS Process Flow

The Forgotten Password Services (FPS) component of APS is essentially a *finite state machine* (FSM). An FSM is a standard computer software construct that make decisions based on a finite number of *states* and the transition conditions between these states.

FPS defines the following states:

State	Condition	Terminal State?
0	Initial Entry	No
10	Identify Form	No
11	Display Missing Form	No
12	More than one user found	No
13	No user found	Yes
14	User is disabled	Yes
15	Too recent success	Yes

16	Too recent attempt	Yes
17	Failure Count (Lockout)	Yes
18	Missing required data from LDAP	Yes
20	Display Verify Form	No
21	Missing/Invalid data from Verify form	No
22	Retry verify form	No
30	Change Password form	No
31	Change Password invalid	No
80	Display confirmation form	Yes
90	Error	Yes
91	Incorrect verification	Yes
92	Timeout	Yes
99	Error state (internal errors)	Yes

Only states ending in zero are *external* states. That is, ones that will exist whenever FPS is invoked. FPS can then convert to an internal state, resulting in a redirection that is, for all intents and purposes, equal to another external state.

The state transitions are defined as:

Enter	**To**	**Exit**	**Cause**
0	10	10	Initial entry
0	99	90	Identify form not configured
10	11	10	Invalid/missing data
10	12	10	More than one user found
10	13	90	No user found
10	14	90	User is disabled
10	15	90	Too recent success
10	16	90	Too recent attempt
10	17	90	Lockout count exceeded
10	18	90	Insufficient/missing data from LDAP

10	20	20	User identified, verify form defined
10	80	80	User identified, no verify form
10	99	90	Unconfigured fields posted
20	21	20	Invalid/missing data posted
20	22	20	Verification failed, retry defined
20	30	30	Verification successful, change next
20	80	80	Verification successful, no change
20	91	90	Verification failed
20	92	90	User did not answer in time
20	99	90	Unconfigured fields posted
30	80	80	Password change successful
30	31	30	Password change failed

Whenever Forgot (Forgot.exe on Windows NT) is executed, it communicates with the SmAPS library on the SiteMinder server. All logic is actually contained in the SmAPS library; Forgot is merely a communications stub. Basically, the only logic in Forgot is for handling communications errors.

FPS first determines the initial state. This will always be one of the external states (number ending in zero). This is done using the referrer (the page that sent the user to Forgot) and cookies.

Once the initial state is determined, FPS can determine what the expected POST data is supposed to be (if any), validate it, and determine the next state. It can then determine the next place that the user is to be redirected and whatever setup is required for that page.

If debugging is turned on (using the DEBUG statement in the SmPortal.cfg file **and** the DEBUG setting in the FPS configuration file), state changes are recorded in the log. Problems can often be diagnosed using this information.

Chapter 6: Change Password Interface (SmCPW)

This chapter discusses how to use and customize the Change Password Interface (SmCPW). There is a default form supplied with APS for changing passwords. This form can, within limits, be customized by a site with no HTML programming. If further customization is required, the entire form can be replaced easily with minimal Web programming.

This section contains the following topics:

Change Password Forms (see page 271)
Query String Options (see page 273)
Setup (see page 275)

Change Password Forms

While APS provides a default form for doing so, custom forms can be written for password changes. They can be written in any language that can display a form. The form must POST to the SmCPW program (SmCPW.EXE on Windows) and may pass any or all command line arguments to it. If command line arguments are passed, the **TARGET=**<*targetURL*> should be passed, so that SmCPW knows where to send the user when the process is complete.

Note: Tests have shown that Netscape Communicator Version 4.70 may not support HTTP_REFERER correctly, so that HTML forms may not work correctly. No other browsers have yet been identified that have this same failing.

Prior to APS version 2.1, custom forms had to pass their query string to SmCPW during the posting process. Thus, such custom forms had to be written in a server-side scripting language, such as JSP or ASP, or as a CGI program. Since version 2.1, this is no longer required; custom forms can be written in HTML, if desired. If SmCPW receives a POST without any query string (the HTML form posts just to SmCPW), it will use its caller's (referrer's) query string.

SmCPW, when presented with an HTTP GET request, will generate an HTML form so that a user may enter password information.

SmCPW need not be used to produce the form. Instead, provide your own HTML for the form, if special processing is desired.

Any form that you desire can be used, as long as it provides three input fields: OldPassword, NewPassword and VerifyPassword. The last two must contain identical values. Set up your form to post its data to SmCPW or to whatever you have renamed that file.

An easy way to start is to run SmCPW through a browser and to select **View Source** from the browser’s menu. This action displays the raw HTML used by the form. You can save this HTML into a file and modify it as desired.

You can also run SmCPW from the command line and save its output. When run from the command line with no arguments (other than command line switches), SmCPW outputs the HTML to produce the form. From the command line, you can specify the -L and -C switches to make SmCPW produce the HTML in other languages; however, this approach requires that you have set up the proper translation files.

A single change password page may not be sufficient for your site. There aretwo types of password changes: Required and Optional. You can detect the difference using query string values (which is how SmCPW identifies the case) or you can use completely different forms. The different forms can be linked into your site separately through the APS configuration file and/or your home page (for voluntary password changes).

Additionally, password change *warnings* can invoke the page. You may wish tobuild an entirely separate page for this case that displays the number of days left to change the password (the number of days is available using the *<DaysLeft>* macro).

Customization without Custom Forms

SmCPW gets some of its text from the SmCPW.lang files located in the language directories on your Web Server. This includes not only the text to display for each prompt, but limited control of HTML tags as well.

- Additional attributes wanted inside the BODY tag, such as background colors, using the BODY key in the LANG file.
- An arbitrary block of HTML to appear immediately after the body tag can be specified using the LOGO key in the LANG file.
- A block of arbitrary HTML to appear *after* the FORM tag is closed but *before* the JavaScript using the COPYRIGHT key in the LANG file.

You can modify these values in the lang file to perform some substantial customization of the default form without writing any web code.

Handling Errors

By default, SmCPW displays any error message (including content violations and communications errors) and forces the browser *back* to the form (using JavaScript). You can easily override this behavior, but it will require more work and more maintenance for your site. If you wish to use custom forms, no need exists to override this error handling, nor do you need to use a custom form to override this behavior.

All content and LDAP error messages are stored in files called APS.lang, located in the language directories on your Policy Server. Any of these messages can be modified using a standard text editor.

The password change confirmation and all communications error messages are located in the SmCPW.lang files stored in the language directories on your Web Server.

If any message to be displayed starts with http:, https: or /, SmCPW assumes that the message is a URL. After performing macro substitution (including replacing macros arriving in SmCPW's query string), SmCPW redirects the user to the URL instead of displaying the message.

An easy test is to put the URL into the language file as desired, but put the text msg: in front of it. When you invoke the error, SmCPW displays the message with all macros replaced within the normal message box. Remove the msg: and SmCPW redirects the user to the page.

You can selectively redirect for some errors and use the standard messages for others.

If you are using internationalization as well, duplicate the URLs or specify different URLs for each supported language.

Query String Options

When running under a Web Server, the syntax of SmCPW is as follows. Note that APS can and will automatically construct these arguments if the redirection events are properly configured (see Chapter 3, APS Configuration File (APS.CFG).

```
SmCPW{.exe}?
        [Optional]
        [&][DaysLeft=<daysleft>]
        [&][GraceLogins=<grace logins left>]
        [&][DaysRemaining=<days remaining>]
        [&][CancelTo=<cancelURL>]
        [&][Target=<targetURL>]
        [&][<additional arguments>]
```

The **.exe** component is required on Windows.

If **Optional** is specified, then the default form displayed by SmCPW has a **Cancel** button. If the user clicks this button, the browser is redirected back to the page that invoked the form (or <*cancelURL*>, if specified). The **Cancel** button is dependent on JavaScript support by the browser. If the Browser does not support JavaScript, a standard hyperlink labeled **Cancel** appears instead.

DAYSLEFT=<*daysleft*> may be supplied to indicate how long the password has left before it expires. There are two cases:

- The argument is not supplied at all. SmCPW will consider that the password *must* be changed. By default, no additional messages are displayed a and no **Forget It** or **Cancel** buttons appear.
- **DAYSLEFT=**<*daysleft*> is supplied. The password *may* be changed and no time limit exists. The **Forget It** or **Cancel** button is displayed (as if **Optional** were specified) in the default form. A message indicating that the password must be changed within "<*daysleft*> days" is displayed by the default form.

  http://www.xyz.com/CPW/SmCPW.exe?DAYSLEFT=3

The **TARGET=**<*targetURL*> argument identifies the page to which the user should be directed upon completion of the password change. The <*targetURL*> should be passed in a URL-encoded format to avoid problems with embedded characters.

GRACELOGINS and DAYSREMAINING are used when the password has expired, but a grace period or a number of grace logins is configured. SmCPW will modify its displayed text to communicate these values to the user.

Additional key/value pairs may also be supplied, each set separated from its predecessor by an ampersand ("&"). These key/value pairs will be parsed and saved by SmCPW.

When SmCPW redirects to the next page, either by redirecting the specified target or to a custom message page, these key/value pairs will be used to replace macros in the target of the redirection. The URL is examined for text between < and > (angle brackets). The text and the angle brackets are replaced by the value associated with the text.

For example, if the SmCPW invocation is:

```
SmCPW?Target=%2FMyPage%2Easp%3F<lang>&Lang=EN
```

The user is redirected to:

```
/MyPage.asp?EN
```

When APS builds the redirection because of an event, it can construct complex URLs with such query strings and can include user attributes in that query string. See the section about Redirections later in this document.

Setup

SmCPW must be placed into a directory that is visible to a Web Server tha is protected by a SiteMinder Web Agent. This directory must have *execute* privileges. To create a CGI-bin directory if the web server is Microsoft's Internet Information Server, use the Microsoft Management Console. If the web server is Netscape Enterprise Server, modify the obj.conf file to create this directory. The installation program usually places this file into this directory:

SiteMinder Web Agent/Bin/Web/CPW.

It is not necessary that the program actually be called SmCPW.EXE (though for troubleshooting purposes, this name is desirable). You may rename the program as needed; however, include the **.exe** extension on Windows. The remainder of this document refers to the program as SmCPW.

A response attribute may be required to be passed when posting to SmCPW. This attribute explicitly identifies the user changing the password. With normal Web Agent configuration, this response is unnecessary (before APS version 4, this response was required). If SmCPW issues a message that it cannot identify the user, this response is required. If this response exists, SmCPW uses it, so no disadvantage exists to setting up the response even it is unnecessary.

See SiteMinder documentation for how to set up Response Attributes. The active expression required is as follows:

```
<@ lib="smaps" func="SMCPW" param="" @>
```

This is set up as a standard HTTP-Variable type of attribute, though you do not specify an attribute name. You must select Active Expression as the attribute type, then select the Manual Entry page in order to create this Active Response, since there is no variable name.

This attribute is only required on the POST rule for the SmCPW program. The GET rule does not need it.

Chapter 7: Help Desk Interface (APSAdmin)

APSAdmin refers to both the Help Desk Interface application program that can be set up to run on your Web Site and the part of the APS API that it calls to perform its operations. This chapter primarily deals with the program itself, though it will discuss the configuration of the API as well. The term APSAdmin, as used in this chapter, will refer to the CGI program unless otherwise indicated.

APSAdmin is an on-line user maintenance application. It is designed to deal with the APS-specific data in a user record, but has limited capacity for maintaining other user fields. It is not intended as a general purpose user maintenance or CRM utility. However, it *is* designed to *augment* existing systems, such as DMS.

This chapter assumes that APSAdmin is already installed and operational according to the instructions in the Installation Checklists (and also discussed in Installation). This chapter discusses how the program works and various options available to control its functionality, look and feel, localization issues and which fields are available for reading and writing.

APSAdmin requires JavaScript support on the browser.

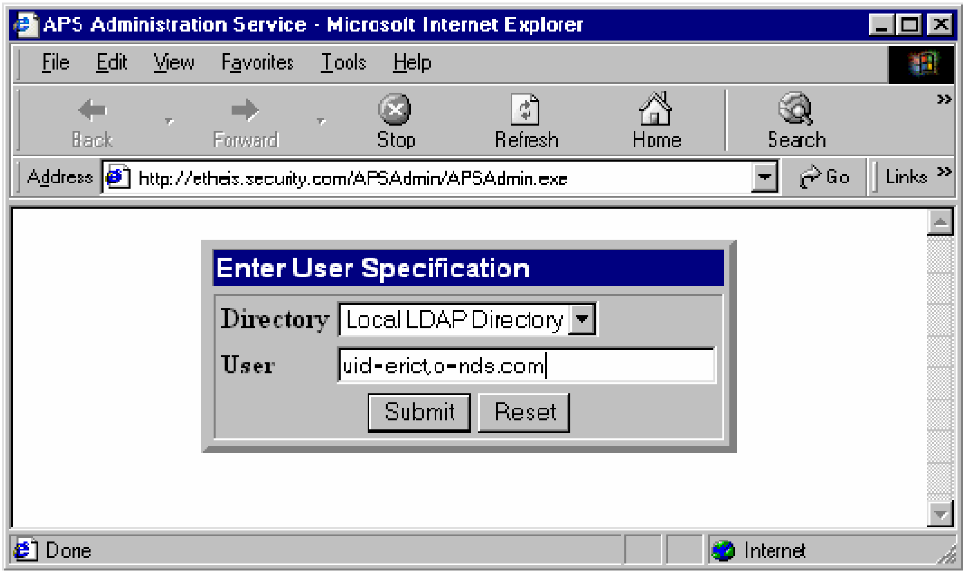
This section contains the following topics:

User Selection (see page 278)
User Display (see page 282)
User Update (see page 285)
Specifying Form Content and Access Control (see page 285)
Running APSAdmin from the Command Line (see page 295)

User Selection

If APSAdmin is invoked without specifying a user (discussed later), it will present a user selection panel.

By default, the user selection panel asks for the User Directory in a drop-down box (if more than one directory is available) and asks for the user path in a standard text box. The format of the user path is directory-type specific: for example, LDAP directories will require a full LDAP DN, whereas an ODBC directory will require just a user id.


APS Administration Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address http://etheis.security.com/APSAdmin/APSAdmin.exe
Go Links
Done
Internet

Enter User Specification

Directory: Local LDAP Directory

User: uid=eric,o=nds.com

Submit Reset

When the user hits the Submit button, APSAdmin will communicate with the APS library on the Policy Server to retrieve the record.

Customization Options

APSAdmin supports a number of options in this mode. Some options can be specified in the query string, some options are specified using SiteMinder Responses.

Restricting the Directory or Directory Type

The Query String (that part of the URL after a question mark) can specify all or part of a user directory specification. This can be used to restrict the directory from which the online user can select. APSAdmin will only display those directories that qualify. If only one directory qualifies, then no directory needs to be selected, only the user.

This can be used by setting up SiteMinder rules that restrict the URL that the administrator can access. In these cases, the link (from wherever the user accesses APSAdmin) should contain the query string restricting the directory.

The directory is specified using in the query string as "DIR=<directory>". The <directory> is compared, from left to right, with the directory list that exists in the SiteMinder Policy Store.

For example, to restrict the user to only LDAP directories, (on Windows) use:

http://server/APSAdmin/APSAdmin.exe?DIR=LDAP:

For ODBC directories only, (on Solaris) use:

http://server/APSAdmin/APSAdmin?DIR=ODBC:

The actual directory can be restricted as well, by specifying the directory itself, such as:

http://server/APSAdmin/APSAdmin?DIR=LDAP%3A%2F%2F127.0.0.1

In this case, only the one directory will be available.

Partial directory specifications are also supported.

If the directory specification has unusual characters in it, they should be URL-encoded.

Making the User Display Read Only

If READONLY appears in the Query String, the user, when displayed, will be in read-only mode; that is, all allowed fields will be displayed, no data entry will be possible.

If multiple query string options are used, READONLY must be separated from other options with an ampersand ("&"), such as:

http://server/APSAdmin/APSAdmin?DIR=LDAP:&READONLY

The READONLY option has no effect on user selection. It will be passed to the user display to control operation.

Showing a Help Button

Some sites may wish to display a Help button on this panel. APSAdmin can display a Help button that will display a site-supplied URL.

To make APSAdmin provide a Help button, define a static response (using the SiteMinder Policy Server User Interface) called HELPURL. The value of this attribute should be the URL containing the help text to be displayed.

Different help URLs can be specified for different APSAdmin panels by using different rules based on the query string (this will be complicated because of the many query string options, but it is possible --- you will have to use regular expressions when defining the Rule).

Showing a Cancel Button

Some sites may wish to display a Cancel button on this panel. APSAdmin can display a Cancel button that, when pressed, will return the user to a specified URL.

To make APSAdmin provide a Cancel button, pass the desired URL in the query string using the CancelTo option, such as:

http://server/APSAdmin/APSAdmin?CancelTo=%2F%2Fsvr/abc.htm

Note that the target is URL-encoded.

The CancelTo URL is passed to the user display panel.

Suppressing the Reset Button

To suppress the Reset button, use the APSAdmin.lang file (located on the Web Server) to translate the key USERSELECT_RESET to a null value. The Reset button will not be displayed.

Changing the Prompts

All text on the User Selection page is translated using the APSAdmin.lang file stored on the Language directory on the Web Server. This includes field labels, the window title, the dialog title and error messages.

Even if your site is not using internationalization, these prompts can be changed by modifying the English translation files.

Customizing the Look and Feel

The look and feel of this panel can be customized using a Cascading Style sheet. APSAdmin, by default, embeds style definitions directly into its output. However, before doing so, it checks for the existence of a file, on the same directory as the APSAdmin program, called APSAdmin.css. If the css file exists, it will be used instead of the default styles.

The simplest way to build a css file is to bring up the form and select View Source within your browser. Simply copy the styles (those lines in the HTML header starting with a period) from the HTML into a new file called APSAdmin.css and store that file on the same directory as the APSAdmin executable. After that, modify the css file as desired.

By examining the generated HTML source, an HTML-savvy programmer can determine which styles are used for the various elements, then modify them to create the desired look and feel. Be forewarned, however, that Style Sheet support varies between browsers and browser versions, so be sure to test it for your supported platforms.

The same style sheet file is used for both User Selection and User Display panels, though, in most cases, different styles are used on each form. If the css file is used, it will have to be used for both forms.

Suppressing the User Selection Panel Altogether

CA expects that most sites will not want to use the User Selection panel for any of a number of reasons:

- There is no search capability.
- The entry of user DN's for LDAP directories is cumbersome and error-prone.
- It is not "user-friendly".
- The administrator may be able to select users that he does not have access to (APSAdmin will display an "Access Denied" message, but this may not be desirable).

There are a number of ways that APSAdmin can be used that would bypass this screen:

Example 1

An existing user management system could place a button on the User Maintenance form labeled, say, Access. When pressed, it invokes the APSAdmin utility, perhaps in its own window, for the selected user by including the user specification in its query string (see the next section).

Example 2

An existing user management system that has its own directory navigation (complete with search) could display a user icon for displayed users that invokes APSAdmin for the associated user.

Example 3

A site could write its own navigate/search engine that just links directly to the User Display portion of APSAdmin.

User Display

If APSAdmin is invoked with a user, it will attempt to display the information about that user.

The user is specified to APSAdmin using two query string options. Usually, these both must be URL-encoded:

- The User Directory, specified as
  `DIR=<`*`directory`*`>`
- The user identifier or DN, specified as
  `USER=<`*`user identity`*`>`

For example:

http://server/APSAdmin/APSAdmin?DIR=LDAP%3A%2F%2F127.0.0.1
&USER=uid%3Derict%2Co%3Dnds.com

Other options may appear in the query string as well.

If the current administrator does not have access to the requested user, no fields are configured in the APS Configuration File, or a communications error occurs, an error message will be displayed.

Customization Options

APSAdmin supports a number of options in this mode. Some options can be specified in the query string, some options are specified using SiteMinder Responses.Customization Options

Making the User Display Read Only

If READONLY appears in the query string, the display will be in read-only mode; that is, all allowed fields will be displayed, no data entry will be possible.

If multiple query string options are used, READONLY must be separated from other options with an ampersand ("&").

Showing a Help Button

Some sites may wish to display a Help button on this panel. APSAdmin can display a Help button that will display a site-supplied URL.

To make APSAdmin provide a Help button, define a static response (using the SiteMinder Policy Server User Interface) called HELPURL. The value of this attribute should be the URL containing the help text to be displayed.

Different help URLs can be specified for different APSAdmin panels by using different rules based on the query string (this will be complicated because of the many query string options, but it is possible --- you will have to use regular expressions when defining the Rule).

Showing a Cancel Button

Some sites may wish to display a Cancel button on this panel. APSAdmin can display a Cancel button that, when pressed, will return the user to a specified URL.

To make APSAdmin provide a Cancel button, pass the desired URL in the query string using the CancelTo option.

Note that the target is URL-encoded.

Showing a Close Button

Some sites will display the user information form in a read-only mode in a pop-up window. In this case, a Cancel button is inappropriate, since it will leave the window open, but on a new URL.

To tell APSAdmin to display a Close button, use Close on the URL. APSAdmin will then display a Close button, with associated JavaScript to close the current window. Note that if this JavaScript is executed in the main window of the browser, the user will be prompted when the button is pressed.

As is the case with all APSAdmin buttons, the text on the button can be modified in the APSAdmin.lang file.

Suppressing the Reset Button

To suppress the Reset button, use the APSAdmin.lang file (located on the Web Server) to translate the key USERDISPLAY_RESET to a null value. The Reset button will not be displayed.

Changing the Prompts

Some text on the User Display page is translated using the APSAdmin.lang file stored on the Language directory on the Web Server. This includes the window title, the dialog title, contents of drop-down entry values, error messages and quite a few screen constants. It *may not* include field prompts.

APSAdmin does not normally translate the field and section prompts. These prompts are defined in the APS Configuration File and are passed to the APSAdmin program through the API. Under normal operation, they will be displayed as provided (this eliminates the need to update the APSAdmin.lang for every possible prompt).

If a site wants to translate prompts, change the prompt definition in the APS.cfg file so that the prompt starts with the character #. APSAdmin will recognize this as a translation request. The prompt, as supplied, will be translated. The default value for the prompt will be the same as the prompt, with the # character removed.

This applies to both field and section prompts.

Customizing the Look and Feel

The look and feel of this panel can be customized using a Cascading Style sheet. APSAdmin, by default, embeds style definitions directly into its output. However, before doing so, it checks for the existence of a file, on the same directory as the APSAdmin program, called APSAdmin.css. If the css file exists, it will be used instead of the default styles.

The simplest way to build a css file is to bring up the form and select View Source within your browser. Simply copy the styles (those lines in the HTML header starting with a period) from the HTML into a new file called APSAdmin.css and store that file on the same directory as the APSAdmin executable. After that, modify the css file as desired.

By examining the generated HTML source, an HTML-savvy programmer can determine which styles are used for the various elements, and then modify them to create the desired look and feel. Be forewarned, however, that Style Sheet support varies between browsers and browser versions, so be sure to test it for your supported platforms.

The same style sheet file is used for both User Selection and User Display panels, though, in most cases, different styles are used on each form. If the CSS file is used, it will have to be used for both forms.

Changing the Graphics

The small graphics displayed are supplied with APSAdmin as GIF files. They can be replaced with any graphics desired. Keep in mind, however, the size at which they will be displayed.

User Update

APSAdmin confirms the changes upon completion of the user update. This is done using normal error translation for the APSAdmin.lang file on the Web Server. As is the case elsewhere, if the translated message appears to be a URL, APSAdmin will redirect the user to the URL instead.

If the message does not redirect the user, the confirmation message is displayed. By default, when the user clicks the OK button, he will be redirected back to the User Select panel (with no arguments, so all directories will be accessible). This is usually not desirable.

If a SiteMinder response called RETURNTO (HTTP_RETURNTO) is defined, APSAdmin will redirect the user to that URL.

If RETURNTO is not defined, but a CancelTo argument is in the query string, the user will be redirected to the URL identified by CancelTo.

Specifying Form Content and Access Control

The fields, groups and sections displayed by APSAdmin, whether a field is read or read/write, the prompts, whether a password is validated and whether a particular user is even accessible to a given administrator, are all defined in the APSAdmin section of the APS Configuration File.

Access Control

Access to specific users (or directories) by specific administrators can be controlled, in a general way, using the Allowed keyword in the APSAdmin section of the APS Configuration File.

The Allowed keyword identifies whether the current administrator (defined in the Setting Override) can access a particular user (defined by the Value of the setting) at all.

By default, all access to APSAdmin is denied. There must a an applicable instance of the Allowed keyword that specifically grants rights for an administrator to a particular user.

In the APSAdmin section, an additional function is available when specifying the target. This function, Self(), is TRUE if the administrator is the same as the target user. This function can be used, for example, to allow all users specific access to their own records or to allow a Help Desk person access to the entire directory *except* for himself (by using NOT Self() for the setting).

Once the administrator (or on-line user) has "rights" to the target user, APS will continue to check to see which fields can be read and/or written.

User Directory "Portability"

Different types of User Directories impose different naming requirements on the fields in the user record. Some sites impose further naming conventions, particularly for RDBMS-based directories.

For example, the "Full Name" of a user in an LDAP directory, using the IETF standard, is stored in an attribute called cn. For an ODBC database, this might be a column called userFullName.

APS allows sites to *remap* user field names in the Mappings section of the APS Configuration File. The format for such an entry is:

```
<logical name>={<restricted expression>}<underlying name>
```

The <*logical name*> can be any name without spaces. References to this name in the APSAdmin section of the APS Configuration File will actually reference the <*underlying name*> (assuming that the <*restricted expression*> applied).

The <*restricted expression*> is the same as a standard override expression, except that it cannot reference user-specific details, such as attributes and group membership. It can, however, reference functions like IsLDAP(), IsODBC(), and IsInDirectory().

To present a solution to our most recent example:

```
FullName={IsLDAP()} cn
FullName={IsODBC()} userFullName
```

Because of these two lines in the Mappings section, FullName can be referenced in the APSAdmin section regardless of whether the user exists in an LDAP directory or an ODBC directory.

If a line exists for a given <*logical name*>, if the <*underlying name*> is blank, the field, for most intents and purposes, does not exist and APSAdmin will ignore any references to it. This can be useful when an attribute exists in one directory, but not in another.

Entries are not required in the Mappings section. If an entry does not exist at all (as opposed to existing with a null <*underlying name*>), APS will use the <*logical name*> that it is looking for as the actual name.

Mappings are used throughout APS. For certain operational fields required by APS, if a field is mapped to a null value, APS will use the logical name anyway (probably resulting in an error). However, APSAdmin will still honor null mappings regardless.

Groups

This discussion relates not only to column/attribute names, but to group names as well.

Under LDAP, in fact, a site *must* map group names in order to use them in the APSAdmin section because LDAP group DNs contain equal signs (=).

Specifying Attributes/Columns

A user field is "available" to APSAdmin if it appears in the APSAdmin section of the APS Configuration File and *explicitly* allows access to that attribute, by the current administrator, for the target user. By default, administrators do not have access to *any* fields (even if **Allowed**).

Furthermore, read access and write access are separately controlled. An administrator might have read rights, but not write rights, vice versa, both, or neither for a particular field.

The format of a field access specification is:

```
    <right>.<field>={<admin override>} <target override>
```

The <right> can be READ, WRITE or RW (for read and write).

The <*field*> refers to the name of the attribute or column and is subject to the mappings described in the previous section.

The <*admin override*> defines whether this particular setting applies to the current online user (the administrator). If omitted, the setting applies to all users.

The <*target override*> defines whether this right/field is applicable for the user currently requested (the target user). This value can be supplied as TRUE to indicate that it applies to all target users.

During APSAdmin read operations, only the first reference to a particular <*field*> applies. The APSAdminRead function returns XML describing the requested (target) user. In this XML, each attribute is identified as Read, Write or Read/Write. When multiple settings apply for the same <*field*>, only the first <*right*> is returned to APSAdmin.

However, during APSAdmin write operations, APS checks to see if *any* write (write or read/write) access is allowed for this administrator and target. If so, the write is allowed.

Specifying Groups

A user's group membership is "available" to APSAdmin if it appears in the APSAdmin section of the APS Configuration File and *explicitly* allows access to that group membership, by the current administrator, for the target user. By default, administrators do not have access to *any* such information (even if **Allowed**).

Furthermore, read access and write access are separately controlled. An administrator might have read rights, but not write rights, vice versa, both, or neither for a particular group membership.

The format of a field access specification is:

```
<right>.GROUP.[set the product group or family]={<admin override>} <target
override>
```

The *<right>* can be READ, WRITE or RW (for read and write).

The *[set the product group or family]* refers to the name of the group and is subject to the mappings described in the previous section. For LDAP groups, this field *must* be remapped, since the equal sign ("=") in the DN will confuse the parser.

The *<admin override>* defines whether this particular setting applies to the current online user (the administrator). If omitted, the setting applies to all users.

The *<target override>* defines whether this right/field is applicable for the user currently requested (the target user). This value can be supplied as TRUE to indicate that it applies to all target users.

During APSAdmin *read* operations, only the first reference to a particular *[set the product group or family]* applies. The APSAdminRead function returns XML describing the requested (target) user. In this XML, each attribute is identified as Read, Write or Read/Write. When multiple settings apply for the same *[set the product group or family]*, only the first *<right>* is returned to APSAdmin.

However, during APSAdmin write operations, APS checks to see if *any* write (write or read/write) access is allowed for this administrator and target. If so, the write is allowed.

Grouping into Sections

APSAdmin returns field and group information in the order in which they appear in the APSAdmin section of the APS Configuration File (note that only settings that apply to the current administrator are considered and duplicate references to the same field or group are ignored).

Entries can be grouped together into sections using the Section keyword in the APS Configuration File. All settings after a Section definition will appear in that section until a new Section keyword appears.

APSAdmin will display a section header for grouped values.

Sections can only have an administrator override and a name (not a target override).

Once the first section is started, there is no way to "unsection" attributes.

If, after evaluating all of the settings in the section, a section does not contain any field or group references, the section is suppressed and not displayed.

Prompts

By default, APSAdmin uses the name of the field, group, or section as its prompt (the text label alongside the data). This is rarely desirable.

The prompt for each field, group, or section can be changed in the APS Configuration File on a line by line basis. This means that the prompt can be different based on the administrator, target and/or context (discussed later).

To set the prompt for an item, at the end of the setting (it must be after the *<target override>* or section name), place a *prompt specification*:

```
Section=UserInfo [PROMPT User Information]
RW.FullName={@HelpDesk} TRUE [PROMPT Name]
```

The prompt will be passed through the XML along with the actual name. The APSAdmin interface will display the prompt.

Translating Prompts

APSAdmin does not normally translate prompts, since screens are so dynamic. If you wish a prompt to be translated, specify the prompt as the translation key, prefixed with a pound sign (#). For example:

```
Section=UserInfo [PROMPT #User Information]
RW.FullName={@HelpDesk} TRUE [PROMPT #Name]
```

The prompt itself (with the # sign) will be used as the translation key. The default value will be the key *without* the # sign. In other words, the following entry in APSAdmin.lang would be used in the above example:

```
key #Name
val Full Name
```

However, if the #Name key were not found, Name would be used as the translation.

Null Sections

If a section has neither a prompt nor a name (e.g. Section=), the APSAdmin interface will display a section header as a horizontal rule (HTML's <HR> tag).

Varying Content by User

As mentioned previously, each setting that specifies a field or group with its access rights can be overridden by on-line user (Administrator). The same item can be specified more than once, with different expressions for *<admin override>* and *<target override>*. The first entry that applies for a combination of administrator and target will be used.

A site can creatively vary content by making adjustments to these override fields. For example, the field smapsExpirePasswordDays is one of the APS attributes that allows the setting of a password expiration period for a specific user. It should not be used directory-wide, but is frequently used for administrator accounts (that do not get used very often) to prevent their passwords from expiring. The following lines from the APSAdmin section of the APS Configuration File demonstrate one use for varying content by user:

```
READ.smapsExpirePasswordDays=
                {@HelpDesk AND NOT @HelpDeskManager}
                Not IsNull("smapsExpirePasswordDays")
                [PROMPT Expiration]
RW.smapsExpirePasswordDays={@HelpDeskManager}
                true [PROMPT Expiration]
```

In this example, Help Desk personnel (but not Help Desk Managers) will *see* the value of this attribute if the *viewed* user has such an override. If the target user does not have an override, no value is displayed. Even if the value is displayed, the Help Desk person cannot change it.

However, the Help Desk Manager will always be able to see and set the field for all users.

If the field is shown, it will always be shown in the same place on the panel, no matter what kind of Help Desk administrator.

Varying Content by Context

APSAdmin can also, within limits, vary content by *context*. Recall that all override expressions support *context macros*. These are values that only exist at certain times ("contexts") during processing and can be tested within an override.

The APSAdmin interface always sends a context macro of APSAdmin=YES. In the APS Configuration File, you can make sure that a particular field only displays when requested by the APSAdmin interface by using something like:

```
    RW.FullName={%APSAdmin="YES"} TRUE
```

A utility other than APSAdmin using the API would not be expected to pass an APSAdmin macro and this setting would not apply to it.

The APSAdmin interface allows a site to pass arbitrary context macros using either a SiteMinder Response ("Header Variable") or an environment variable (but not both). The former option is far more useful than the latter.

A site could, for example, use multiple virtual directories to access the APSAdmin executable. Under the SiteMinder Policy Server User Interface, each would require its own realm, rule and response. Since each can also have its own policy, each context can be protected differently.

For each context, specify an additional response attribute called APSADMINCONTEXT containing the desired macros. Each macro is in the format <*key*>=<*value*> and multiple macros can be specified, separated by semicolons.

Environment variables must exist within the user process within which the CGI program (APSAdmin) executes. The name of the environment variable is HTTP_APSADMINCONTEXT and its value is the same as the APSADMINCONTEXT response. Setting a environment variable for use under a Web Server, Windows usually requires a machine reboot and Solaris requires that the Web Server be restarted.

An example of the use of contexts:

- An alternate URL exists for the APSAdmin utility (using virtual directories) called:
  `//server/Reset/APSAdmin.exe`
- APSADMINCONTEXT is set to PasswordReset=YES for this particular Realm/Rule.
- The APSAdmin section of the APS Configuration File has the line:
  `RW.userPassword={%PasswordReset="YES"} TRUE`
- When a user (who has rights) access this URL, the user password appears and is available for reset.

Note: Warning: Recall that APSAdmin will use the first setting that applies to the current administrator/target combination. When using contexts, it is possible to fall through the definitions for that context and hit another setting, specific to no particular context. Be sure to fully test all combinations of contexts to make sure that unexpected fields (or rights) do not appear.

Using Self()

The function Self() can be used within <*target overrides*>. This function is available at no other place within the APS Configuration File. The value of the function is TRUE if, and only if, the current administrator is the same as the target user. A really good example of its use is:

```
RW.userPassword={@HelpDesk} NOT Self()
```

This allows all help desk administrators to reset passwords for all users other than themselves.

```
    Allowed=Self()Another example:
    RW.mail=Self()
```

Assuming no other settings in the APSAdmin section, these lines will allow all users to change their own email address, but no other fields.

Validating Password Resets

APSAdmin supports administrative password resets. This operation allows an administrator to change the password of a user to a known value. It does not require that the administrator know the existing password. This is different from the Change Password Interface that 1) only allows users to change their own password and 2) requires the entry of the current password.

The APSAdmin interface recognizes a specific field name called userPassword and will present an appropriate entry interface.

When changes are posted for a field called userPassword, APS may or may not be required to validate the new entry against required password content. Whether or not this validation is to occur is defined in the APS Configuration File, in the APSAdmin section, using the Validate Password setting.

Every instance of the Validate Password setting will be checked by APS. If any setting applies that has FALSE value, the password content will not be validated. If no Validate Password setting applies or none is contained in the file, passwords *will* be validated.

Validation is against password content only. It does not include password history or attribute (profile) checking.

Forcing Password Changes

The Force Immediate Change setting in the APS Configuration File controls whether APSAdmin (the API) should automatically set the immediate change flag (smapsImmediateChange) when a password is reset.

If any applicable setting is TRUE, the immediate change flag will be set.

One exception: if, on the same update, the immediate change flag is explicitly updated, APSAdmin will not update it automatically.

Disabling, Enabling & Redisabling

When building the XML to describe a user account, APS examines a number of things to determine if that element is currently disabling the account, whether the element can be used to disable the account, or if the element will cause the account to be disabled at the next login.

As part of the XML user tag (<*user*>), APS will return the reasons that the account is disabled, if any, in an XML attribute called "disabled". This is a text string identifying all of the reasons that the account is disabled.

The APSAdmin interface will display these reasons at the top of the panel, along with a graphic indicating that the displayed record is disabled. Possibly, the administrator can reset all of these reasons. However, recall that access to each element in the user record must be explicitly granted. It is possible that not all reasons are accessible to the current administrator.

For groups, APS will check to see if the underlying group is a disabling group. If so, an XML attribute is associated with the field identifying the "Disabled Reason". The APSAdmin interface uses this information to determine if the disabled graphic might be required for that group's membership.

The APSAdmin interface "knows" how to handle the disabling fields smapsDisableUntil, smapsDisableAfter, and smapsMustLoginBy. There is no special XML information generated. If any of these fields disable the account, the interface will display the disabled icon.

During XML generation, APS also may produce an additional XML attribute for the smapsBaseDate and smapsFailureCount fields called redisable. If this attribute is passed, it means that the current value of these fields will cause the account to be disabled at their next login, even if all other disabling values are clear.

For example, if the password has expired, the record might be placed into an LDAP disabling group. The group *might* be passed to the interface with the proper reason code. The reason code *will* be passed as part of the XML user tag. If the smapsBaseDate is included in the output, it will have the redisable setting, since, even if the membership in the disable group is reset, the password will just expire again. To get around this, the administrator will need to reset the Base Date. The APSAdmin interface displays a special icon in this case that looks like the standard disabled icon, with a little arrow showing that the user account will become disabled.

Each cause that can disable the user account is separately presented, both in the XML and by the APSAdmin interface. Elements must have explicit rights granted for an administrator to view and/or change. This can be used by a site to create layered administration. For example, there might be three reasons that an account is disabled:

- Failure Count
- Password Expired
- Credit Limit Exceeded

All administrators will see all of the reasons why the account is disabled at the top of the panel (even if they do not have rights to the individual elements). The rationale is that a Help Desk administrator will be able to tell a user, presumably on the phone, why they cannot log in.

A low level Help Desk administrator might have the right to reset users who are locked out because of failure count:

```
RW.GROUP.FailureCount={@HelpDesk} true
```

But not have *any rights* to the other reason codes.

The next level up, *@HelpDeskManager*, might be able to do Password Expired as well:

```
RW.GROUP.PwdExpired={@HelpDeskManager} true
```

But Credit Limit Exceeded can only be reset by the accounting system. The site may wish to grant read rights to that group, so that it is explicitly shown on the form, but no administrator may change it.

Special Field & Group Handling

The APSAdmin interface recognizes all of the APS-specific fields by logical name. It will translate, edit, and display these fields in appropriate way for their meaning. For example, the date fields (which APS always stores in Greenwich Time) will be displayed/entered in the time zone local to the browser. APSAdmin will handle the conversions to and from GMT.

To reiterate: these fields are recognized by *logical name*. If these field names are remapped to something else (like smapsBaseDate is passed as BaseDate), the APSAdmin interface will not recognize the field and will not perform its special processing.

APSAdmin recognizes disabling groups, as discussed in the previous section. This not only includes the "known" APS disabling groups, but site-specific, custom ones as well.

Other Field & Group Handling

Other fields and groups, without specific APS processing, can be used as well, with any desired access right (read, write, read/write). However, the interface will display the value using unmodifable defaults and accept unedited input. No special processing will be performed and there is no way, using the APSAdmin executable, to specify special processing.

The primary purpose of allowing these fields is so that a Help Desk administrator can use things like Name and Phone Number to accurately identify the user on the phone. This type of information often needs to be displayed.

Sometimes, simple fields might also be editable, like Email Address. Just keep in mind that the input field is of limited length and there will be no syntax checking of any kind on the field.

APSAdmin should not be used as a replacement for a full user administration system; it is not intended to be so. If this kind of functionality is required, there are a number of tools available. For LDAP directories, Delegated Management Services (DMS2) is uniquely qualified for this purpose.

Running APSAdmin from the Command Line

APSAdmin can also be run from the command line. This is not useful (or recommended!) for a production system, but can be used effectively during automated or semi-automated testing to 1) reset user records to known states and 2) determining if expected updates to the user record are taking place.

The command line syntax is:

```
APSAdmin -A<adminPath>
             -C<adminCreds>
             -R<userPath>|-W<xmlPath>
             [<macro1>=<value1>]
```

Where:

-A<*adminPath*> The fully qualified path name of the administrator. This is the SiteMinder login user, not the SiteMinder GUI administrator.

This value is a full user path in the form

```
<Namespace>://<server>/<user>
```

such as:

```
LDAP://127.0.0.1/uid=erict,o=Airi
us.com
```

-C<*adminCreds*>	The credentials of an administrator defined to the SiteMinder Policy Server User Interface (GUI). This is not a SiteMinder login. This administrator must have User Management rights. Both the administrator ID and the password must be supplied, separated by a semicolon (";"). The combination can be passed encrypted (using APSEncrypt).

-R<*userPath*>	The fully qualified path to the user whose information is to be retrieved. You can specify either the -R option (to read a user) or -W (to write a user), but not both.
	This value is a full user path in the form
	`<Namespace>://<server>/<user>`
	such as:
	`LDAP://127.0.0.1/uid=erict,o=Airius.com`

-W<*xmlPath*>	Specifies the file containing the updates to write, in XML format. For a description of the format of this file, see the description for the APSAdminWrite API function starting on Unsupported "Page" Cross-Reference.
<*macro1*>=<*value1*>	Context macros can be defined on the command line using this syntax (multiples can be specified). These macros are passed to APS and can be used in Settings Overrides in the APS Configuration File. This can be really useful to limit the output on a user read.

Security Note

There may be a security concern with command line use of APSAdmin. Specifically, when run under the Web, APSAdmin can "guarantee" that the adminPath is the currently authenticated user. However, when run from the command line, this is not possible. There are two things to consider here:

- The administrator credentials must still be supplied. This requires an administrator password.
- The APSAdmin executable can be protected by the operating system to prevent it from being run, except by the Web Server (Agent) user.

Chapter 8: Daily Processing (APSExpire)

This section contains the following topics:

APSExpire (see page 297)
APSExpire Event Log (see page 299)
Event Processing (see page 300)
Command Line Parameters (see page 301)
Performance Adjustments/Job Definitions (see page 302)

APSExpire

APSExpire is a command line utility that can be run as an AT process on Windows or a cron job on Unix. This means that the execution of APSExpire will take place on a *timed, periodic* basis. APSExpire examines users to determine if any of the following events applies:

- The user is not initialized (smapsNextAction is not set).
- A user's password will expire within a number of days.
- A user's password has expired, but a grace period or grace logins remain.
- A user's password has expired, no grace period or grace logins remain and the user record will be disabled.
- A user's account has remained inactive for too long and will be disabled shortly.
- A user's account has remained inactive for too long and will be disabled now.
- A user's account is eligible for purging (deletion).

Each execution of APSExpire processes a single User Directory or part of a User Directory, defined as a *job* in a special APSExpire section of the APS Configuration file. (See Chapter APS Configuration File (APS.CFG) for options on how to define these processing jobs).

When executed, APSExpire will search the specified directory for users who either have a blank smapsNextAction or a value for smapsNextAction that is prior to the current date and time. For each user found, APSExpire determines what actually must be done for that user.

There will be occasions where no action should be taken for the user. When this occurs, a new value for smapsNextAction will be calculated, the user record modified, and APSExpire will continue to the next user entry.

The key to APSExpire is the value of smapsNextAction. APSExpire will only process those user records with a value of before *right now* or blank. If the smapsNextAction attribute for a user is wrong for any reason, APSExpire will "pick up" the user record at the wrong time. If the date is too early, little will happen, since a new date will be calculated and stored back, essentially correcting itself.

If the date is too late, APSExpire will not process the record until the later date. This may cause the user record to not be processed in a timely manner.

Note that all of the "real time" functionality (expiring passwords and user accounts) can also take place in "Just In Time" mode. That is, if one of these events is detected during user login, APS will take action immediately. Thus, if the Next Action date is not correct, it only affects the *asynchronous* action of APS, it does not create a security hole; the event will be processed the next time that the user logs in or when the Action Date comes up.

Every time APS handles a user's record, it recalculates smapsNextAction, based on the current settings in APS.cfg and other dates in the user's record (such as the last login date). APS figures out the next thing that has to be done for the user and saves the date of that event.

Sometimes, during normal processing, conditions change that impact this Next Action date. This could be for three different reasons:

- The user record is modified outside of APS, usually by user maintenance applications.

  The recommended way to handle this is to blank out the smapsNextAction field for the user. The next time that APSExpire runs for that directory, a new value will be calculated.

- A setting in APS.cfg has changed. This means that the settings used for a calculation have changed and some or all users need to have the date recalculated.

  Depending on the change, the amount of work in this case can be different. If the change causes dates to be moved farther into the future (such as Password Expiration changes from 60 to 90), a site does not have to do anything, since, after 60 days, all of the users with invalid dates will be examined, a new date calculated and then stored. The data will clean itself up.

However, if the date moves closer to today (such as Password Expiration changes from 90 to 60), there will be user records with a date too far into the future. A site can take one of two actions:

- Do nothing. Existing users will not be processed until the date shows up and there will be inconsistencies, but the data will eventually clean itself up.
- Run APSExpire with the -A option to process all users. APSExpire will recalculate the next action date for all users defined in the processing job.

- Several settings exist within APS.cfg, each with a different override. Something about a user's record changes such that a different override applies to the record. This is essentially the same as point 1 above. However, the case may not be as obvious because the override may be based on conditions entirely independent of APS variables.

Sites will need to examine these cases and blank out smapsNextAction if such a change is made to a record.

APSExpire Event Log

APSExpire keeps a log of events that it detects. The location of this log is controlled by APSExpire command line parameters (if the location is not specified, APSExpire writes the log to the screen).

Log entries are formatted specifically so that they can be processed by standard desktop tools, such as Microsoft Excel or Seagate Crystal Reports. The logs conform to a file format called Comma Delimited Format (CDF). This format requires that each field be separated from others using a comma, each record terminated by a carriage return and linefeed, and string fields should be quoted.

Each entry in the log appears as:

```
<Event>, <As Of Date>, <User Path>
```

The *event* is the event that APSExpire detected. The *As Of Date* is the date (and time) at which the event occurred. This may not be the same as the current date (if APSExpire has not run for a period of time). The *User Path* identifies the user for whom the event was detected.

By default, APSExpire also outputs headers for each column. This is standard for a CDF file. These headers can be suppressed using command line options.

Event Processing

APSExpire is only concerned with seven conditions (Events) that affect user records. The events and the action taken by APSExpire are described in this section.

In all cases, APSExpire will calculate a new value of smapsNextAction and save it back to the user record.

If no action is required for a user record, smapsNextAction will be set to 99999999999999Z CYCLE COMPLETE so that APSExpire will no longer process the record.

The following conditions (events) concern ASPExpire:

- The user is not initialized (smapsNextAction is not set).

  APSExpire will check to see if any other event should also occur. The user's date will be updated. This event is never logged to the log file.

- A user's password will expire within a number of days. This is useful, for example, when a typical user only logs in, say, every 90 days. Users may never see the on-line password change warnings.

  APSExpire will send email if a template is specified for this event. The mail can contain the date that the password will expire and the number of days remaining before expiration.

  This event will be logged with an event id of PASSWORD WARNING.

- A user's password has expired, but a grace period or grace logins remain.

  APSExpire will send email if a template is specified for this event. The mail can contain the date that the password expired and the number of days remaining before the user is disabled.

  This event will be logged with an event id of PASSWORD SOFT EXPIRE.

- A user's password has expired, no grace period or grace logins remain and the user record will be disabled.

  APSExpire will send email if a template is specified for this event. The mail can contain the date that the password expired.

  APSExpire will disable the user account.

  This event will be logged with an event id of PASSWORD HARD EXPIRE.

- A user's account has remained inactive for too long and will be disabled shortly.

  APSExpire will send email if a template is specified for this event. The mail can contain the date that the account will expire and the number of days remaining before expiration.

  This event will be logged with an event id of ACCOUNT WARNING.

- A user's account has remained inactive for too long and will be disabled now.

  APSExpire will send email if a template is specified for this event. The mail can contain the date that the account expired. Typically, this mail is sent internally.

  APSExpire will disable the user account.

  This event will be logged with an event id of ACCOUNT EXPIRE.

- A user's account is eligible for purging (deletion).

  This event will be logged with an event id of ACCOUNT PURGE. No other action will be taken.

Command Line Parameters

APSExpire supports a number of command line arguments to control its processing:

<*job*>	The name of job to run as defined in APS.cfg.
-v	Verbose mode (tracing output on).
-q	Quiet mode (logging off).
-A	Process all users (for initialization).
-H or -?	Display usage information.
-T	Do not output column titles.
-m	Turn off mail (default with -A).
+m	Turn on mail (default without -A).
-l logfile	Path to write tracing and processing information to.
-o outfile	Path to write activity to (the "log").
-e errfile	Path to write errors to.

Performance Adjustments/Job Definitions

The APSExpire section of the APS configuration file controls the operation of APSExpire.

A site defines jobs by name. Each setting is the name of a job and the values define the criteria for the job. When APSExpire executes, a job name must be specified. The program will look for the definition of this job in this section of the file.

Each job defines a user directory or subset of a user directory. The syntax for ODBC and LDAP directories are different.

LDAP Directories

For LDAP directories, jobs are defined using this syntax:

```
<job name>= <LDAP directory>
                                READ(<ip>)
                                BASE(<base DN>)
                                SCOPE(<scope>)
                                FILTER(<filter>)
```

<job name> is an arbitrary name for the job.

<ip of LDAP directory> is the ip address, the network name, or the SiteMinder User Directory name of an LDAP directory defined to SiteMinder through the Policy Interface (it cannot contain spaces if used here). If it is an ip address, it may contain the port address as well. This must match up with the definition of a User Directory in the SiteMinder Policy Store (APSExpire will attempt to look up the directory using this value).

READ(*<ip>*) is an optional clause that tells APSExpire to read from a different directory than the base directory. In some cases, much higher performance can be achieved by reading from a dedicated replicant directory that either SiteMinder does not use at all or is the last directory in a failover chain. If specified, however, the alternate directory must be a replicant of the "real" directory.

BASE(<*search Base*>) is optional and defines the scope of the search. If it is not specified, APSExpire searchs the entire directory using the search base defined in the SiteMinder User directory entry. This is useful when an entire LDAP directory is *not* to be processed as a single job. Sites do this when the LDAP directory is very large and APSExpire processing is to be spread over multiple servers or jobs.

SCOPE(<*scope*>) is optional and is generally used with the BASE option above. <*scope*> can either be base or sub. It specifies how the LDAP search should be processed.

FILTER(<*extra filter*>) is another optional setting that allows a site to further refine a job. This filter is ANDed with any filters that APSExpire uses for its own operations. Once again, this is intended to segregate an LDAP directory into smaller jobs for performance reasons.

When using BASE, SCOPE and FILTER, it is the responsibility of the site to make sure that every user will be processed. APSExpire does not examine the sum of all defined jobs to ensure that all users get processed.

ODBC Directories

For ODBC directories, jobs are defined using this syntax:

```
<job name>= <ODBC directory>
                    WHERE(<extra WHERE clause>)
```

<*job name*> is an arbitrary name for the job.

<*ODBC directory*> is the DSN name or the SiteMinder User Directory name of an ODBC user directory (neither can have embedded spaces in this context) defined to SiteMinder through the Policy Interface. This must match up with the definition of a User Directory in the SiteMinder Policy Store (APSExpire will attempt to look up the directory using this value).

WHERE(<*extra WHERE clause*>) is another optional setting that allows a site to further refine a job. This clause is ANDed with any WHERE clause that APSExpire uses for its own operations. This is intended to segregate an ODBC directory into smaller jobs for performance purposes.

When using WHERE, it is the responsibility of the site to make sure that every user will be processed. APSExpire does not examine the sum of all defined jobs to ensure that all users get processed.

Performance

Each User Directory should be processed by APSExpire on a periodic basis, preferably daily, even if the special processing for the events is not desired. This ensures that new user records will be initialized properly.

APSExpire must run on the Policy Server machine.

The time that it takes to process a single user is relatively fixed (though based on the platform). However, the time required to find users to process is a function of the size of the directory. It may be obvious, but it must be said: Larger directories will take more time to process than smaller directories.

At APS Version 4.0, APSExpire was completely redesigned to handle larger User Directories. The amount of time required to process each user could not be reduced significantly (in fact, other required enhancement increased the per-user time slightly). However, the change to use smapsNextAction has significantly improved performance for locating users to be processed (which, prior to Version 4.0, was where APSExpire spent most of its time).

The use of smapsNextAction, however, introduced the restrictions presented earlier in this chapter. The performance improvement was so dramatic that the restrictions were deemed acceptable.

In an additional performance-enhancing effort, APSExpire jobs need not define an entire User Directory; they can define a subset of a directory. In this way, a site can load-balance multiple APSExpire executions across multiple Policy Servers and/or spread the processing over a period of time.

When dividing a User Directory into subsets, it is the responsibility of the site to ensure that all users are covered. For example, it would be an error (undetected by APS) to create two jobs, one to handle users whose last names start with the letters "A" through "M" and another job for user names starting with "O" through "Z", since users whose names start with "N" would never be processed.

Chapter 9: Forgotten Password (FPS) Interface (Forgot)

This chapter discusses the various forms used by the Forgotten Password (FPS) Interface. A site must supply all FPS forms, though sample forms are provided in both JSP and ASP. CA Professional Services can be contracted to produce this custom code as a separate contract.

This section contains the following topics:

Programming Notes (see page 305)
Initial Invocation (see page 306)
Identify Pages (see page 307)
Verify Forms (see page 311)
Change Pages (see page 314)
Confirm Pages (see page 315)
Error Pages (see page 318)
Mail Error Pages (see page 318)

Programming Notes

The example code provided in this section is presented in ASP (and JSP is provided with APS as well). However, FPS does not require the use of any specific web language. Some pages can actually be presented entirely in static HTML, others will require some sort of server-side scripting so that fields can be initialized.

Most fields can and should be edited on the client side using JavaScript. However, some browsers allow JavaScript to be disabled and others do not support JavaScript at all, so the process should not depend entirely on it. Additionally, a savvy user can save the source, edit the JavaScript, then post from the modified page. FPS will perform its own edits to ensure that the data is valid.

When building the pages for the FPS process, always consider the BACK button on the browser. If a user uses FPS to recover her password, what happens when that user leaves her terminal and another user goes over and presses the BACK button?

For this reason, all but the static pages should be set to expire and not be cached. In addition, many pages should also do whatever they can to prevent the redisplay of the page. This can be done using the following code. Note that this depends on JavaScript and is subject to the caveat above.

```
<input type="hidden" name="control" value="0">
<script language="JavaScript 1.1">
<!---
If document.forms[0].control.value == "0")
     document.forms[0].control.value = "1";
else
    window.location.replace("SomeplaceElse.html");
-->
</script>
```

The replace method causes the current page to be *replaced* in the browser history with the specified page.

FPS pages must exist on the same server as the FORGOT CGI program, since the requisite cookies will not be transmitted from FORGOT to the pages if they are not.

All FPS forms can and should be referenced using SSL (https). To do so, configure all of the URLs in the APS configuration file with the https: prefix (assuming that your Web Server is capable of handling SSL requests).

Initial Invocation

To start the Forgotten Password process, your site needs to direct the user to the Forgot CGI stub (supplied with FPS). The user will then be redirected into the FPS process.

Typically, this link exists in one of two places. It can exist on the site's public (unprotected) home page ("Forgot your user id or password?") or directly on the login form. In either case, the query string passed to Forgot should include the URL that the user should be returned to upon completion as a Target macro. For example:

```
<a href="/FPS/Forgot?Target=HomePage.html">
Forgot your user id or password?
</a>
```

If placed on the login form, the target should be the URL that the user originally desired. From within an FCC, the following could be used:

```
<a href="/FPS/Forgot?Target=$$target$$">
Forgot your user id or password?
</a>
```

Identify Pages

URL

When FPS is first invoked by a user, this is the page that FPS will cause to be displayed. It is expected to present a form to the user. It must be a page that is *not* protected by SiteMinder. It need not be on the same directory as the Forgot stub, but must be on the same logical Web Server.

Technically, this could be a static page that merely presents a form. Each field on the page should be defined in the APS Configuration File in as either Optional or Required so that FPS will accept input.

The form must post to the Forgot CGI program.

In its simplest form:

```
<HTML>
    <HEAD>
            <TITLE>
                    Please Identify Yourself
            </TITLE>
    </HEAD>

    <BODY>
            <b>Please Identify Yourself</b>
            <FORM method="post" action="/FPS/Forgot">
                    <b>First Name:&nbsp;</b>
                    <INPUT type="text" name="FirstName">
                    <br>
                    <b>Last Name:&nbsp;</b>
                    <INPUT type="text" name="LastName">
                    <br>
                    <INPUT type="submit" value="Submit">
                    <INPUT type="reset">
            </FORM>
    </BODY>
</HTML>
```

Note: On a Windows NT Web Server, the action attribute of the FORM tag must reference Forgot.exe.

That is all that is needed, bare bones. Additional features would include expiring the page (though it is not necessary in this case) and performing edits to ensure that no field is blank or contains invalid data when posting. Additional fields might also be desirable.

Additional code will be necessary if this form is also to be used as the Missing and Multiple URL forms.

Missing URL

If the user fails to fill in all of the required fields or one or more of the fields contains an invalid character, FPS will redirect the user to this URL. The URL will have a list of the missing or invalid fields appended to it, separated with ampersands ("&"). A question mark will be appended before the field names, if required.

There is currently no way to tell if the fields in error were empty but required, or if they contained invalid data.

The original fields values will be in a cookie called NPSFPSFields. This is useful in the event that you wish to repopulate the fields with original values.

Usually, this is the same form as the URL (see URL). Typically, two additional pieces of code are needed. First, near the top of the page, an error message should be displayed to the user:

```
<% If Request.ServerVariables("QUERY_STRING") <> "" Then %>
    <font color="red">
            Fields marked in red are either missing or invalid.
    </font>
<% End If %>
```

Then any prompt associated with a field in error can be turned red.

```
<% ShowRed = (Instr(Problem, "&FirstName&") > 0)
    If ShowRed Then %>
            <font color="red">
<% End If %>
            <b>First Name:&nbsp;</b>
<% If ShowRed Then %>
            </font>
<% End If %>
```

Of course, there are other ways to do this - this is only one way.

Multiple URL

If the user filled out the form presented by the URL, but multiple users were found matching the input criteria, this page is displayed. Depending upon your site, one of two cases may be in effect, in which case, this form must handle the correct case.

First, if the required fields on the URL must *absolutely* define a single user, then you have a data or login error in your flow and this page must handle the error case. In this case, this page should be just a static HTML page reporting the (internal) error and referring the user to your customer support line.

If, on the other hand, your identify form has optional fields on it that can be used to further refine the user search, this page might just request further information. For example, if your identification form requires first name and last name, but mail is optional, there may be more than one "Bill Smith" on file. This page could then request that one or more of the optional fields be supplied to further refine the search.

Typically, a site will use the same URL for this page as for basic identification, with the additional code:

```
<% If UCase(Request.ServerVariables("QUERY_STRING"))
            = "MOREDATA" Then %>
    <font color="red">
            We are unable to uniquely identify you.
            Please supply more information.
    </font>
<% End If %>
```

Of course, this code would have to be ELSE'd with the equivalent code in Missing URL (see Missing URL) if the same page were to be used for both.

Not Found URL

If *no* user matches the input criteria, this page will be presented to the user. This is a terminating case (the FPS process is done, since no user can be identified).

If not supplied, the value of Missing URL is used, with ?NotFound appended.

This should generally be static HTML referring the user to a customer support number.

Disabled URL

If the user is properly identified, but it turns out that the user is disabled (due to an APS disabled group only), the user will be redirected to this page. This is a terminating case, since the FPS process cannot continue.

Note that this case is different, but related to, the Lockout case. Lockout may also cause the user to become disabled (using the Lockout Group DN setting). The first time this occurs (when the user is originally locked out), the user will be sent to the Lockout URL. If the user attempts to use FPS again, they will be sent to this page instead of the Lockout URL, since the user is disabled.

This should generally be static HTML referring the user to a customer support number. Administrator intervention will be required, since the user account will have to be enabled.

Too Recently Used URL

If a user attempts to use FPS too soon after he has already recovered a password (based on the Max Success Frequency setting), the user will be sent to this page.

This case is detected using smfpsLog.

This should generally be static HTML referring the user to a customer support number. The user cannot use FPS until the requisite time has elapsed.

A customer service application *could* delete information contained in smfpsLog to make FPS available again, but this would be counter-productive. The user needs to call the Help Desk to recover the password; the administrator there can reset the password. There is no need to delete logged information; to do so would be deleting auditing information.

Too Recently Attempted URL

If a user attempts to use FPS too soon after any attempt to recover a password (based on the Max Attempts Frequency setting), the user will be sent to this page.

This case is detected using smfpsLog.

This should generally be static HTML referring the user to a customer support number. The user cannot use FPS until the requisite time has elapsed.

A customer service application *could* delete information contained in smfpsLog to make FPS available again, but this would be counter-productive. The user needs to call the Help Desk to recover the password; the administrator there can reset the password. There is no need to delete logged information; to do so would be deleting auditing information.

Lockout URL

If the user is locked out due to the Lockout Count, the user will be redirected to this page. The user will always be redirected to this page the first time that the count is reached. Further attempts to use FPS will also redirect the user to this page *unless* the Lockout Group DN specifies a disabled group, in which case the user will be considered disabled and will be redirected to the Disabled URL on subsequent attempts.

This case is detected using smfpaLockoutCounter.

This should generally be static HTML referring the user to a customer support number. The user cannot use FPS until smfpaLockoutCounter has been cleared.

Verify Forms

At the end of the Identify phase, FPS knows the identity of the user. Some sites may wish to attempt to verify that the user is who he claims he is (this process is highly recommended). If this is desired, the verify forms (defined in the Verify section) will be used. If verification is not desired, the Verify section will be omitted from the APS configuration file.

URL

This URL is the form used to request verification from the user. Its input fields must be defined as Optional or Required and it must have a Lookup setting so that FPS will know what to do with the input data.

Just like the Identify form, this could very simply be an HTML form. If the verification data is fixed, such as "Enter your Phone Number, Middle Name, and Account Number", then this will suffice. However, this is not terribly secure (but may be secure enough for your site).

Most sites want to read a challenge (question) of some sort from the user record and present it to the user. To this end, the Verify form can have initial data passed to it using the Initial setting.

Use the Initial setting to define what the page will need. FPS will create a cookie called NPSFPSData that will contain the initial data. Each field will be named as in the Initial setting, followed by an equals sign ("="), followed by the field's value, URL encoded. Multiple fields will be separated by ampersands. Thus, an Initial setting of:

```
Initial=FirstName=givenname;Question=SmPINQ
```

might create the cookie:

```
NPSFPSData=FirstName=Eric&Question=What%20is%20the%20name%20of%20your%20dog
```

You will have to use a server-side scripting language to read the cookie and parse it. Each scripting language has its own capabilities for doing so.

FPS does not use the cookie after setting it, so you are free to destroy it (FPS will destroy it automatically when the form is posted).

In order to "see" this cookie, your page must reside on the same logical Web Server as the FORGOT stub.

If special instructions (see initial) are used such that multiple values are to be passed for the same field (the Pick keyword), then the multiple values will be passed, each URL encoded, separated by pipes, as in:

```
Initial=Question=SmPINQ[Format=A,Pick=3]
NPSFPSData= Question=100|200|203
```

If a Pick clause is used in special instructions, all of the answer fields must have the same element name and must appear in the same order as presented to the page. FPS tracks the questions and the order in which they should be presented and expects the answers back in the same sequence.

If there is not enough data to satisfy a Pick clause (because of a consume or restrict clause), no data will be returned to the page for that field. If the field is not marked as required (by placing an asterisk before its name in the Initial setting), the Verify form will be displayed, but there will be no value for the field. If the item is marked as required, the No Data URL page will be displayed.

It is especially important that the Verify page expire and prevent reviewing using the BACK button. Without taking these preventative measures, *after* a user has successfully used FPS, another user can press the BACK button to view the answers to the questions.

Another way to solve this problem is to make the input fields *password* fields, since the entered text is masked by asterisks and is not retained when the BACK button is pressed. However, this makes the page harder for users to use.

Missing URL

If the user fails to fill in all of the required fields or one or more of the fields contains an invalid character, FPS will redirect the user to this form. The URL will have a list of this missing or invalid fields appended to it, separated with ampersands ("&"). A question mark will be appended before the field names, if required.

The original fields values will be in a cookie called NPSFPSFields. This is useful in the event that you wish to repopulate the fields with original values. There is currently no way to tell if the fields in error were empty but required, or if they contained invalid data.

Usually, this is the same form as the Verify URL above. Typically, two additional pieces code are needed. First, near the top of the page, an error message should be displayed to the user:

```
<% If Request.ServerVariables("QUERY_STRING") <> "" Then %>
    <font color="red">
            Fields marked in red are either missing or invalid.
    </font>
<% End If %>
```

Then any prompt associated with a field in error can be turned red.

```
<% ShowRed = (Instr(Problem, "&Answer&") > 0)
   If ShowRed Then %>
    <font color="red">
<% End If %>
            <b>Answer:&nbsp;</b>
<% If ShowRed Then %>
    </font>
<% End If %>
```

Of course, there are other ways to do this - this is only one way.

Retry URL

If a user answers the verification question(s) wrong, your site may decide to let the user try again. This is not recommended, but some sites wish to do so.

If your site still wishes to do this, supply a Retry URL setting to specify the form for FPS to use. This form must have the same fields as the Verify URL. Initial values *are* evaluated again, so random question selection (special instructions) may change.

Care should be taken when using this keyword to prevent an unlimited number of attempts (by a hacker). FPS puts no constraints on the number of retries that a user can make. smfpsLockoutCounter is only checked each time that the FPS process is started, not each time that a Verify form is displayed.

If the Retry URL is not specified and the user answers the question incorrectly, he will be transferred to the Invalid URL instead.

Invalid URL

If the user answers the verification wrong, FPS will direct the user to this page. This is a terminal condition; the FPS process is terminated.

This should generally be static HTML referring the user to a customer support number, since they are unable to verify their identity. The user is not locked out of FPS.

No Data URL

The Initial setting (described below) specifies the initial field settings that the URL is to receive before processing. In the Initial setting, some values may be marked *required*. If so, and no value (or not enough values) can be determined, the user will be sent to this page instead. This is a terminal condition (FPS processing terminates - the user cannot use FPS).

In other words, this page is invoked when some data that is required from the user record is unavailable, either because it is blank, restricted, or consumed. This may be a temporary condition (because of some cases of consumed and the use of *restrict*) or it may be permanent and require that the user login and modify their own profile.

Typically, this is a page directing the user to a customer service representative and can be static HTML.

Timeout URL

This setting specifies a page to send the user to if the user fails to answer the questions within the time period specified by the Timeout setting ("You did not answer quickly enough"). This reduces the ability of a hacker to socially engineer the answers to the verification question(s).

This is a terminal condition (FPS processing terminates - the user cannot use FPS at this time).

This could be a static HTML page. It could also be Forgot itself (or an intervening page with a message) so that the user can restart the FPS process.

The user is not locked out of FPS.

Change Pages

At the end of the verify phase, FPS is certain of the identity of the user. Some sites may allow the user to change their password to a known value at this time. This is one of the big potential security holes in your site. If this functionality is desired, the Change forms (defined in the Change section) will be used. If change password is not desired, the Change section should be omitted from the APS configuration file.

URL

This URL is the form used to request the password change from the user. It requires *exactly* two input fields called NewPassword and VerifyPassword. They should be password fields.

As usual, the form posts to the FORGOT stub.

Just like the Identify and Verify forms, this could very simply be an HTML form. However, since FPS will return the user to this URL when there is a problem with password content (the error message, localized, will be passed as the query string), a server-side scripting language is desirable in order to process (display) a query string, if one exists.

It is especially important that this page expire and prevent reviewing/reposting using the BACK button. Without taking these preventative measures, *after* a user has successfully used FPS, another user may be able to press the BACK button to change the password again.

Timeout URL

This setting specifies a page to send the user to when the form is not submitted within the period specified by the Timeout setting ("You did not answer quickly enough").

This is a terminal condition (FPS processing terminates - the user cannot use FPS at this time).

This could be a static HTML page. It could also be Forgot itself (or an intervening page with a message) so that the user can restart the FPS process.

The user is not locked out of FPS.

Confirm Pages

There are several different strategies to finishing the FPS process, all configured using the Confirm section of the APS configuration file.

- Give static information to tell the user how to proceed (sending the information via email).
- Give the user the information on a custom page, email, or a combination of the two.

- Let FPS display a message box displaying the information or part of the information (the rest being transmitted via email)
- Log the user in automatically.

One way to do confirmation at the end of the FPS process is to send the information to the user via email. If this is to be done, the confirm page should say something like "The information that you requested has been sent to the email addressuser@somesite.com".

To pass information to a custom Confirm page, specify the Initial setting in the Confirm section. Unlike the similar keyword in the Verify section, this information will not be passed in a cookie (there would be no way to destroy the cookie when weíre done with it). Instead, the information is passed to the Confirm URL in its query string (in exactly the same format as the cookie).

To pass the new password, use password in the Initial setting instead of userPassword (the actual LDAP attribute name), since userPassword will be encrypted (actually, hashed).

CA *strongly* recommends that the user id and password not be presented to the user in the same medium. Send one via mail and display the other.

Another option is to display HalfPassword1 on the custom page and send HalfPassword2 via email. These macros return, in clear text, either the first half or the second half of the user's password. Of course, either the mail or the custom page will have to instruct the user that the two halves must be put together before using.

If you do not wish this information appear in the query string, put the ultimate target URL in the confirm URL setting and put an asterisk in front of it. FPS will dynamically build a page to display the confirmation message (and the initial data will not be placed in the query string). The message will come from the translation for CONFIRM_MESSAGE (the title of the page coming from CONFIRM_TITLE). Macros will be replaced in the confirmation message, including Password, uid, HalfPassword1, HalfPassword2, and OneShotPassword.

It *is* possible to automatically log the user in at the completion of the process. There are several ways to do this. The easiest (but least secure) way to do it, that works with all releases of SiteMinder, is to take the Initial information, pre-populate input fields on a login form, and POST to an FCC. This is not very secure, in that the password appears not only in the query string, but in the initial values for the input fields as well. By pressing the BACK button and *View Source*, this information is very visible, even in an SSL environment. The exception would be to use OneShotPasswords.

Instead of using the "real" password, use the OneShotPassword instead. FPS creats a single-use password when the macro "OneShotPassword" is referenced in a mail file or in an initial setting for the confirmation page. The OneShotPassword requires special set up to use (See Authentication Scheme.) but it is more secure than using a multiple use password, even if you allow the user to select their own password (since you will never show the selected password).

SiteMinder FCCís support a special keyword called "@loginonget" which allows an fcc to be used to authenticate a user without displaying a form (simply a redirect). FPS fully supports this option, using the following steps:

1. On your Web Server, create a file called oneshotfps.fcc. This file should be in your fcc directory, usually <*SiteMinder Web Agent home directory*>/samples/forms.

   This file should contain *exactly* the following text:

   ```
   @target=/FPSOneShot/DoesNotExist
   @username=%uid%
   @password=%OneShotPassword%
   @loginonget
   ```

2. In APS.cfg, in the [FPS-Confirm] section, place the lines:

   ```
   URL=/siteminderagent/forms/oneshotfps.fcc
   Initial=uid;OneShotPassword
   ```

   The actual URL to the FCC should be used.

3. In the SiteMinder Policy Server User Interface, create an authentication scheme called "FPS One Shot" as directed on Unsupported "Page" Cross-Reference (no params, library="smaps").

4. Create a realm called "FPS One Shot". Its filter is /FPSOneShot/ on the correct agent(s). The Authentication Scheme is the one created above ("FPS One Shot".)

5. Create a rule under the "FPS One Shot" Realm called "FPS One Shot Trigger". The filter is DoesNotExist and the action is GET.

6. Create a rule under the "FPS One Shot" Realm called "FPS One Shot Redirect". This is an OnAuthAccept rule.

7. Create a response called "FPS One Shot Redirect". This response should have a single OnAcceptRedirect value of the *real* URL where the user should end up (typically your site's home page).

8. Create a policy called "FPS One Shot". The Users tab should be all users allowed to use One Shot (ODBC and LDAP only). Attach BOTH rules created above. Attach the "FPS One Shot Redirect" response to the "FPS One Shot Redirect" rule.

If different users are to go to different ultimate targets, then different policies can be used for the "FPS One Shot Redirect" rule, for different users, with different responses controlling the actual URL.

Error Pages

If FPS encounters any error during processing for which there is no specific setting to handle, the user will be redirected to the page identified as the Error URL. Most errors are of two types:

- Bad configuration, which should be detected during testing.
- Errors returned from the LDAP server.

Actual error text will passed to the page in the query string (URL encoded).

Generally, the message should be displayed to the user, and the user should be referred to the Help Desk.

Support personnel can obtain additional information about the error from the Console Logs.

Mail Error Pages

The Mail URL page exists only to prevent a problem where we are unable to send mail to the user that is required for the FPS process. The cause of the failure will be one of the following:

- All of the mail files specified could not be found
- All of the files specified were rejected by the SMTP server. The SMTP server will only reject for the following reasons:
  - Bad mail configuration
  - Invalid or missing send to address
  - Internal error with the mail file

Note that a non-existent target address will not cause a failure in any case, since it wonít be detected at send time.

This URL should typically just display an internal error-style message, referring the user to the Help Desk.

Chapter 10: Application Programming Interface

Starting with Version 3.0, APS provides a true Application Programming Interface (API). This interface does not provide a generic interface to updating directories; rather, it provides programmatic access to APS' password evaluation capabilities and run-time authentication checking.

The API is "housed" in a library called APSAPI (under Windows, APSAPI.DLL, and under Solaris, libAPSAPI.so). This library and the required C/C++ language header file, are installed with APS on the Policy Server under the SDK directory.

The API is supplied for C/C++ and COM only. If it must be called from Java, it is up to the site to build a JNI interface to the API, though a JNI interface may be provided with a future release. It is not possible at this time to produce a native Java API, because the APS API calls into the SiteMinder Agent API, which is C/C++ only. Thus, a JNI call must be made at that point. The SiteMinder Agent API remains C/C++ only for security reasons.

CA Professional Services can be contracted to write programs that use this API and can be contracted to help architect applications that use it. However, we do not provide direct support for customer-written programs that use the API. Since APS includes utilities (such as SmCPW and APSAdmin) that call the API, the API itself can be tested cleanly without using customer code. Failures in customer code should be tested using a CA-supplied interface before calling for support.

This section contains the following topics:

Compiling & Linking (see page 319)
Run-Time Requirements (see page 320)
Function Descriptions (see page 320)

Compiling & Linking

The APSAPI package includes a file called APSAPI.h, which contains all of the function definitions for the API. This file must be included in your source code in order to declare the functions.

When linking under Windows, you will need APSAPI.lib (also provided with the APSAPI kit). This file contains information that the linker needs to connect your program to APSAPI.DLL.

Under Solaris, you can specify libAPSAPI.so in your makefile for linking purposes.

Run-Time Requirements

Each of the functions has different requirements as to how it communicates with APS. See the description of each function below to determine the specific requirements.

If a function is to be called remotely (in other words, the program using the API will not be resident on a Policy Server with APS installed), then that machine will require that a configured copy of SmPortal be installed. SmPortal is used to communicate with a SiteMinder Policy Server.

Function Descriptions

IsPasswordValid

Synopsis:

This function can be used to determine if an arbitrary password is valid. It is intended to be used by Administrator Password Reset type applications. It does not check historical information.

Prototype:

```
bool IsPasswordValid (      const char*      lpszUserPath,
                            const char*      lpszPassword,
                            char*            lpszErrorBuffer,
                            int              nErrorBufferSize,
                            const char*      lpszLanguage = NULL,
                            const char*      lpszCountry = NULL,
                            const char*      lpszQualifier = NULL);
```

Callable:

Locally (direct), if available, otherwise Remote (SmPortal)

Returns:

True if the password is valid, otherwise false.

Arguments:

const char* lpszUserPath	The full path to a user. This is in the format LDAP://<server>/<DN> or WinNT://<server>/<*user*>. It is used to determine which settings are in effect, the actual user record is not checked (it need not even exist). This value can be supplied as NULL.
const char* lpszPassword	The actual password to check. Cannot be NULL.
char* lpszErrorBuffer	A caller-supplied buffer to contain any returned error message. This may be the error message describing *why* the password is invalid or it may describe communications or parameter errors.
int nErrorBufferSize	The size of error message buffer above.
const char* lpszLanguage	The ISO-standard code of the desired language that the error message should be provided in. This argument can be supplied as NULL. If so, EN will be used.
const char* lpszCountry	The ISO-standard code of the desired country (locale) that the error message should be provided in. This argument can be supplied as NULL.
const char* lpszQualifier	This argument is usually supplied as NULL, meaning no qualifier. If supplied, it will be used to qualify the service name provided to SmPortal (has no effect when the API is called locally), so that multiple Policy Servers can be used by the API from the same source computer.

Description:

This function generically checks to see if a password is valid. Typically, it is called by administration programs, since the current password is not needed.

The lpszUserPath argument is used to determine which settings apply; the user does not need to actually exist.

Note: The case (upper/lower) of the user's password must change, this function will return UPPER or LOWER in the lpszErrorBuffer argument and return a successful result. If the calling application is going to write this new password to the User's Directory entry, the password's case must be changed before modification.

Limitations:

Even if the following settings are specified in APS.Cfg, this function does not evaluate:

- Percentage change (since the old password is not supplied)
- Specials (embedded user attributes)
- Password History

IsMyPasswordValid

Synopsis:

This function can be used to determine if a password is valid for a particular user. It is intended to be used by user-initiated password changes (though it is not needed, since ChangeMyPassword() also performs this function)

Prototype:

```
Bool IsMyPasswordValid(     const char*     lpszUserPath,
                            const char*     lpszOldPassword,
                            const char*     lpszNewPassword,
                            char*           lpszErrorBuffer,
                            int             nErrorBufferSize,
                            const char*     lpszLanguage = NULL,
                            const char*     lpszCountry = NULL,
                            const char*     lpszQualifier = NULL);
```

Callable:

Remote Only (SmPortal)

Returns:

True if the password is valid, otherwise false.

Arguments:

const char* lpszUserPath	The full path to a user. This is in the format LDAP://<server>/<DN> or WinNT://<server>/<*user*>. This value *cannot* be supplied as NULL.

const char* lpszOldPassword	The existing password on the user account. Cannot be NULL.
const char* lpszNewPassword	The new password to check. Cannot be NULL.
char* lpszErrorBuffer	A caller-supplied buffer to contain any returned error message. This may be the error message describing *why* the password is invalid or it may describe communications or parameter errors.
int nErrorBufferSize	The size of error message buffer above.
const char* lpszLanguage	The ISO-standard code of the desired language that the error message should be provided in. This argument can be supplied as NULL. If so, EN will be used.
const char* lpszCountry	The ISO-standard code of the desired country (locale) that the error message should be provided in. This argument can be supplied as NULL.
const char* lpszQualifier	This argument is usually supplied as NULL, meaning no qualifier. If supplied, it will be used to qualify the service name provided to SmPortal (has no effect when the API is called locally), so that multiple Policy Servers can be used by the API from the same source computer.

Description:

This function checks to see if a password is valid for the specified user. This function actually is never needed, since ChangeMyPassword (below) will check a user's password before changing it.

This function will fail if lpszOldPassword is not valid for the account. If it is not valid, the in-memory Failure Count (for APS authentication) will be incremented (though not stored back to the directory). This may prevent users from being logged in or may cause users to be disabled. If the in-memory Failure Count exceeds Max Failures, this function will not work for that account.

Note: If the case (upper/lower) of the user's password must change, this function will return UPPER or LOWER in the lpszErrorBuffer argument and return a successful result. If the calling application is going to write this new password to the User's Directory entry, the password's case must be changed before modification.

Limitations:

This function requires SmPortal, even if called on the Policy Server, to ensure that the activity is logged to a SiteMinder console (since the user may be locked out).

IsMyPasswordValid2

Synopsis:

This function can be used to determine if a password is valid for a particular user. It is intended to be used by user-initiated password changes (though it is not needed, since ChangeMyPassword() also performs this function). This function is only callable from within Authentication Schemes.

Prototype:

```
bool IsMyPasswordValid2 (    const Sm_Api_Context_t*    lpApiContext,
                             const char*                lpszUserPath,
                             const char*                lpszOldPassword,
                             const char*                lpszNewPassword,
                             char*                      lpszErrorBuffer,
                             int                        nErrorBufferSize,
                             const char*                lpszLanguage=NULL,
                             const char*                lpszCountry=NULL);
```

Callable:

Local Only (from Authentication Scheme)

Returns:

True if the password is valid, otherwise false.

Arguments:

const Sm_Api_Context_t* lpApiContext	API Context as passed from SiteMinder to the calling Authentication scheme.
const char* lpszUserPath	The full path to a user. This is in the format LDAP://<server>/<DN> or WinNT://<server>/<*user*>. This value cannot be supplied as NULL.
const char* lpszOldPassword	The existing password on the user account. Cannot be NULL.
const char* lpszNewPassword	The new password to check. Cannot be NULL.

char* lpszErrorBuffer	A caller-supplied buffer to contain any returned error message. This may be the error message describing why the password is invalid or it may describe communications or parameter errors.
int nErrorBufferSize	The size of error message buffer above.
const char* lpszLanguage	The ISO-standard code of the desired language that the error message should be provided in. This argument can be supplied as NULL. If so, EN will be used.
const char* lpszCountry	The ISO-standard code of the desired country (locale) that the error message should be provided in. This argument can be supplied as NULL.

Description:

This function checks to see if a password is valid for the specified user. This function actually is never needed, since ChangeMyPassword (below) will check a user's password before changing it.

This function will fail if lpszOldPassword is not valid for the account. If it is not valid, the in-memory Failure Count (for APS authentication) will be incremented (though not stored back to the directory). This may prevent users from being logged in or may cause users to be disabled. If the in-memory Failure Count exceeds Max Failures, this function will not work for that account.

Note: If the case (upper/lower) of the user's password must change, this function will return UPPER or LOWER in the lpszErrorBuffer argument and return a successful result. If the calling application is going to write this new password to the User's Directory entry, the password's case must be changed before modification.

Limitations:

This function must be called from within an Authentication Scheme, since it requires the ApiContext. Do not attempt to create your own ApiContext structure, this function will not work.

ChangeMyPassword

Synopsis:

This function can be used to change a user's password. The password is checked for validity before the change is actually made. It is intended to be used by user-initiated password changes.

Prototype:

```
bool ChangeMyPassword(    const char*    lpszUserPath,
                          const char*    lpszOldPassword,
                          const char*    lpszNewPassword,
                          char*          lpszErrorBuffer,
                          int            nErrorBufferSize,
                          const char*    lpszLanguage = NULL,
                          const char*    lpszCountry = NULL,
                          const char*    lpszQualifier = NULL);
```

Callable:

Remote Only (SmPortal)

Returns:

True if the password is changed, otherwise false.

Arguments:

const char* lpszUserPath	The full path to a user. This is in the format `LDAP://<server>/<DN>` or `WinNT://<server>/<user>.` This value cannot be supplied as NULL.
const char* lpszOldPassword	The existing password on the user account. Cannot be NULL.
const char* lpszNewPassword	The new password to store. Cannot be NULL.
char* lpszErrorBuffer	A caller-supplied buffer to contain any returned error message. This may be the error message describing why the password is invalid or it may describe communications or parameter errors.

int nErrorBufferSize	The size of error message buffer above.
const char* lpszLanguage	The ISO-standard code of the desired language that the error message should be provided in. This argument can be supplied as NULL. If so, EN will be used.
const char* lpszCountry	The ISO-standard code of the desired country (locale) that the error message should be provided in. This argument can be supplied as NULL.
const char* lpszQualifier	This argument is usually supplied as NULL, meaning no qualifier. If supplied, it will be used to qualify the service name provided to SmPortal (has no effect when the API is called locally), so that multiple Policy Servers can be used by the API from the same source computer.

Description:

This function checks to see if a password is valid for the specified user. If so, the user's password is changed to the new value.

This function will fail if lpszOldPassword is not valid for the account. If it is not valid, the in-memory Failure Count (for APS authentication) will be incremented (though not stored back to the directory). This may prevent users from being logged in or may cause users to be disabled. If the in-memory Failure Count exceeds Max Failures, this function will not work for that account.

Note: If the case (upper/lower) of the user's password must change, this function will return UPPER or LOWER in the lpszErrorBuffer argument and return a successful result.

Limitations:

This function requires SmPortal, even if called on the Policy Server, to ensure that the activity is logged to a SiteMinder console (since the user may be locked out).

ChangeMyPassword2

Synopsis:

This function can be used to change a user's password. The password is checked for validity before the change is actually made. It is intended to be used by user-initiated password changes. This version can only be called from within an authentication scheme.

Prototype:

```
bool ChangeMyPassword2 (    const Sm_Api_Context_t*    lpApiContext,
                            const char*                lpszUserPath,
                            const char*                lpszOldPassword,
                            const char*                lpszNewPassword,
                            char*                      lpszErrorBuffer,
                            int                        nErrorBufferSize,
                            const char*                lpszLanguage = NULL,
                            const char*                lpszCountry = NULL);
```

Callable:

Local Only (from within an Authentication Scheme)

Returns:

True if the password is changed, otherwise false.

Arguments:

const Sm_Api_Context_t* lpApiContext	API Context as passed from SiteMinder to the calling Authentication scheme.
const char* lpszUserPath	The full path to a user. This is in the format `LDAP://<server>/<DN>` or `WinNT://<server>/<user>.` This value cannot be supplied as NULL.
const char* lpszOldPassword	The existing password on the user account. Cannot be NULL.

const char* lpszNewPassword	The new password to store. Cannot be NULL.
char* lpszErrorBuffer	A caller-supplied buffer to contain any returned error message. This may be the error message describing why the password is invalid or it may describe communications or parameter errors.
int nErrorBufferSize	The size of error message buffer above.
const char* lpszLanguage	The ISO-standard code of the desired language that the error message should be provided in. This argument can be supplied as NULL. If so, EN will be used.
const char* lpszCountry	The ISO-standard code of the desired country (locale) that the error message should be provided in. This argument can be supplied as NULL.

Description:

This function checks to see if a password is valid for the specified user. If so, the user's password is changed to the new value.

This function will fail if lpszOldPassword is not valid for the account. If it is not valid, the in-memory Failure Count (for APS authentication) will be incremented (though not stored back to the directory). This may prevent users from being logged in or may cause users to be disabled. If the in-memory Failure Count exceeds Max Failures, this function will not work for that account.

Note: If the case (upper/lower) of the user's password must change, this function will return UPPER or LOWER in the lpszErrorBuffer argument and return a successful result.

Limitations:

This function must be called from within an Authentication Scheme, since it requires the ApiContext. Do not attempt to create your own ApiContext structure, this function will not work.

Authenticate

Synopsis:

This function can be used from within an Authentication Scheme to allow APS to evaluate the Authentication. It will return any redirection that is a result of the authentication. It can *only* be called from an Authentication Scheme.

Prototype:

```
int Authenticate (        const Sm_Api_Context_t*            lpApiContext,
                          const Sm_Api_UserContext_t*        lpUserContext,
                          Sm_AuthApi_Status_t*               Status,
                          const char*                        lpszParams,
                          char*                              lpszRedirect,
                          int                                nRedirectLen);
```

Callable:

Local, from within an Authentication Scheme Only

Returns:

Always returns zero.

Arguments:

const Sm_Api_Context_t* lpApiContext	API Context as passed from SiteMinder to the calling Authentication scheme.
const Sm_Api_UserContext_t* lpUserContext	User Context as passed from SiteMinder to the calling Authentication scheme.
Sm_AuthApi_Status_t* Status	The status of the authentication. APS can change this status.
const char* lpszParams	Parameters that would be passed to SmApsRedirect, such as macros or a default redirection.
char* lpszRedirect	A caller-supplied buffer in which to return any redirection.
int nRedirectLen	Length of passed redirection buffer

Description:

This function can be called from an authentication scheme to perform the standard authentication time APS checks. Under normal circumstances, this is not required, since SiteMinder will automatically make this call; however, there are some unusual situations where this call will be required.

You must make sure that SiteMinder *does not* call this function when it is called from an Authentication Scheme. You can do this by disabling Password Services for this Authentication Scheme (a checkbox on the Authentication Scheme definition panel in the SiteMinder Policy Interface).

APS assumes that all authentication checking has already been done *before* this call is made (this call should *not* be made during the disambiguation phase). The value of the Status argument reflects the current state of authentication, pass or fail.

Limitations:

This function can only be called by an Authentication Scheme. It requires in-memory structures provided by SiteMinder (and a few that are not passed as arguments), so it can never be called from anywhere else.

APSAdminGetDirectory

Synopsis:

APSAdminGetDirectory retrieves a list of the User Directories (supported by APSAdmin) that are defined to the SiteMinder Policy Server. A valid Administrator name/password (for the password GUI, not for the User Directory) must be passed. This administrator id must have rights to Users.

The information is returned in XML format.

Prototype:

```
int APSAdminGetDirectory (      const char*     Administrator,

                                char*           lpszOutput,
                                int             nOutputSize,
                                char*           lpszErrorBuffer,
                                int             nErrorBufferSize,
                                const char*     lpszLanguage = NULL,
                                const char*     lpszCountry = NULL,
                                const char*     lpszQualifier = NULL);
```

Callable:

Remote Only (SmPortal)

Returns:

Error Code. One of the following:

- APSADMIN_ACCESS_DENIED
- APSADMIN_BAD_ADMIN
- APSADMIN_BUFFER_TOO_SMALL
- APSADMIN_COMM_TROUBLE

Arguments:

const char* Administrator	The credentials of an administrator defined to the SiteMinder Policy Server User Interface (GUI). This is *not* a SiteMinder login. This administrator must have User Management rights. Both the administrator ID and the password must be supplied, separated by a semicolon (";"). The combination can be passed encrypted (using APSEncrypt).
char* lpszOutput	A buffer to hold the returned XML.
int nOutputSize	The maximum size of lpszOutput.
char* lpszErrorBuffer	Returned error message
Int nErrorBufferSize	The maximum size of lpszErrorBuffer
const char* lpszLanguage	The desired language of any error messages. Defaults to EN.
const char* lpszCountry	The desired country (locale) of and error messages. Defaults to US.
const char* lpszQualifier	Not used at this time.

Description:

The result is returned in lpszOutput in XML format. The XML is self-contained; that is, it contains its own DTD (it does not refer to an external DTD).

At the time of this writing, the DTD is as follows:

```
<!DOCTYPE APSGetDirectory [
<!ELEMENT UserDirectories (UserDirectory*)>
<!ELEMENT UserDirectory (#PCDATA)>
<!ATTLIST UserDirectory name CDATA #REQUIRED>
]>
```

All LDAP and ODBC directories are returned. The name returned is the Description field in the User Directory definition in SiteMinder. The data is the server definition (usually the IP address). A directory will *not* to be returned if the description of the directory (as defined to SiteMinder) begins with a pound sign ("#"). The data is the server definition (usually the IP address).

This function is used by APSAdmin to get a list of the available directories for user lookup.

Limitations:

APSAdminGetDirectory only returns LDAP and ODBC directories.

APSAdminRead

Synopsis:

APSAdminRead retrieves information about a specified user and returns it in XML format. The information returned is defined in the APS Configuration file (in the APSAdmin section).

A valid Administrator name/password (for the password GUI, not for the User Directory) must be passed. This administrator id must have rights to Users.

Prototype:

int APSAdminRead	(const char*	Administrator,
	const char*	AdminUserPath,
	const char*	TargetUserPath,
	char*	lpszOutput,
	int	nOutputSize,
	char*	lpszErrorBuffer,
	int	nErrorBufferSize,

```
const char*     lpszMacros = NULL,
const char*     lpszLanguage = NULL,
const char*     lpszCountry = NULL,
const char*     lpszQualifier = NULL);
```

Callable:

Remote Only (SmPortal)

Returns:

If > 0, it is the number of attributes returned.

One of the following error codes if < 0:

- APSADMIN_ACCESS_DENIED
- APSADMIN_BAD_ADMIN
- APSADMIN_BAD_TARGET
- APSADMIN_BAD_FORMAT
- APSADMIN_BUFFER_TOO_SMALL
- APSADMIN_COMM_TROUBLE

Arguments:

const char* Administrator	The credentials of an administrator defined to the SiteMinder Policy Server User Interface (GUI). This is *not* a SiteMinder login. This administrator must have User Management rights. Both the administrator ID and the password must be supplied, separated by a semicolon (";"). The combination can be passed encrypted (using APSEncrypt).
const char* AdminUserPath	The fully qualified path name of the administrator. This is the SiteMinder *login* user, not the SiteMinder GUI administrator. This value is a full user path in the form <Namespace>://<server>/<*user*>, such as: `LDAP://127.0.0.1/uid=erict,o=Airius.com`

const char* TargetUserPath	The fully qualified path to the user whose information is to be retrieved. This value is a full user path in the form <Namespace>://<server>/*<user>*, such as: `LDAP://127.0.0.1/uid =erict,o=Airius.com`
char* lpszOutput	A buffer to hold the returned XML.
int nOutputSize	The maximum size of lpszOutput.
char* lpszErrorBuffer	Returned error message
int nErrorBufferSize	The maximum size of lpszErrorBuffer
const char* lpszMacros	Any macros desired in the form <name>=<value>, multiples separated by semicolons. These can be used as context macros in override expressions in the APS.cfg file.
const char* lpszLanguage	The desired language of any error messages. Defaults to EN.
const char* lpszCountry	The desired country (locale) of and error messages. Defaults to US.
const char* lpszQualifier	Not used at this time.

Description:

The result is returned in lpszOutput in XML format. The XML is self-contained; that is, it contains its own DTD (it does not refer to an external DTD).

```
<!DOCTYPE APSReadUser [
<!ELEMENT User (Attribute|Section|Group)*>
<!ATTLIST User name CDATA #REQUIRED>
<!ATTLIST User disabled CDATA "">
<!ELEMENT Section (Attribute|Group)+>
<!ATTLIST Section name CDATA #IMPLIED>
<!ATTLIST Section prompt CDATA #IMPLIED>
<!ELEMENT Attribute (Value*)><!ATTLIST Attribute name CDATA #REQUIRED>
<!ATTLIST Attribute read (true|false) "false">
<!ATTLIST Attribute write (true|false) "false">
<!ATTLIST Attribute prompt CDATA #IMPLIED>
<!ATTLIST Attribute redisable (true|false) "false">
<!ELEMENT Group EMPTY>
<!ATTLIST Group name CDATA #REQUIRED>
<!ATTLIST Group read (true|false) "false">
<!ATTLIST Group write (true|false) "false">
<!ATTLIST Group prompt CDATA #IMPLIED>
<!ATTLIST Group member (true|false|restricted) "restricted">
<!ATTLIST Group reasoncode CDATA #IMPLIED>
<!ELEMENT Value (#PCDATA)>
]>
```

The specific fields/groups returned for a given user are defined in the APS Configuration file (in the APSAdmin section). The field and group names should be abstracted (they will not usually reflect the "real" name of the underlying attribute).

APSAdminRead has restrictions on certain APS fields. For example, smapsNextAction *cannot* be read/write. If specified that way, the API will only return it as readable, not writeable.

Limitations:

All attributes are available within limits. Certain attributes may be suppressed for reading (such as userPassword and smapsHistory) and others are suppressed for writing (such as smapsHistory and smapsNextAction).

APSAdminWrite

Synopsis:

APSAdminWrite updates information about a specified user supplied in XML format. It cannot create a new user. The fields that can be updated are defined in the APS Configuration file.

A valid Administrator name/password (for the password GUI, not for the User Directory) must be passed. This administrator id must have rights to Users.

Prototype:

```
int APSAdminWrite        (const char*        Administrator,
                         const char*         AdminUserPath,
                         const char*         Instructions,
                         char*               lpszErrorBuffer,
                         int                 nErrorBufferSize,
                         const char*         lpszMacros = NULL,
                         const char*         lpszLanguage = NULL,
                         const char*         lpszCountry = NULL,
                         const char*         lpszQualifier = NULL);
```

Callable:

Remote Only (SmPortal)

Returns:

If > 0, it is the number of attributes actually updated.

One of the following error codes if < 0:

- APSADMIN_ACCESS_DENIED
- APSADMIN_BAD_ADMIN
- APSADMIN_BAD_TARGET
- APSADMIN_BAD_FORMAT
- APSADMIN_UPDATE_FAILED
- APSADMIN_BUFFER_TOO_SMALL
- APSADMIN_COMM_TROUBLE

Arguments:

const char* Administrator	The credentials of an administrator defined to the SiteMinder Policy Server User Interface (GUI). This is *not* a SiteMinder login. This administrator must have User Management rights. Both the administrator ID and the password must be supplied, separated by a semicolon (";"). The combination can be passed encrypted (using APSEncrypt).
const char* AdminUserPath	The fully qualified path name of the administrator. This is the SiteMinder *login* user, not the SiteMinder GUI administrator. This value is a full user path in the form *<Namespace>://<server>/<user>*, such as: LDAP://127.0.0.1/uid=erict,o=Airius.com
const char* Instructions	The XML instructions for the update. The user to be updated is defined in the XML.
char* lpszErrorBuffer	Returned error message
int nErrorBufferSize	The maximum size of lpszErrorBuffer
const char* lpszMacros	Any macros desired in the form *<name>=<value>*, multiples separated by semicolons. These can be used as context macros in override expressions in the APS.cfg file.
const char* lpszLanguage	The desired language of any error messages. Defaults to EN.
const char* lpszCountry	The desired country (locale) of and error messages. Defaults to US.
const char* lpszQualifier	Not used at this time.

Description:

The updates to be performed are supplied in the Instructions argument.

The XML should be formatted according to the following DTD (any contained DTD in the XML is ignored).

```
<!DOCTYPE APSWriteUser [
<!ELEMENT User (Attribute|Group)*>
<!ATTLIST User name CDATA #REQUIRED>
<!ELEMENT Attribute (Value*)>
<!ATTLIST Attribute name CDATA #REQUIRED>
<!ELEMENT Group EMPTY>
<!ATTLIST Group name CDATA #REQUIRED>
<!ATTLIST Group member ("1"|"0") #REQUIRED>
<!ELEMENT Value (#PCDATA)>
]>
```

Limitations:

APS will not write some fields regardless of its configuration. There is also special processing for certain fields *by name*.

With the exception of smapsNextAction, if an attribute does not appear, it will not be updated.

- smapsGenerationalRedirects

  Each value element should be in the format *<key>=<value> <comment>*. Missing keys will be removed from the list.

- smapsLoginHistory

  No values can be specified for this attribute. If this attribute appears (with no values), the attribute is *cleared*. Note that this means that the attribute cannot be changed.

- smfpsLog

  No values can be specified for this attribute. If this attribute appears (with no values), the attribute is *cleared*. Note that this means that the attribute cannot be changed.

- userPassword

  The user's password is changed. This may involve encryption.

  If password validation is turned on, the password's content will be validated. If validation fails, no update will occur and the proper error message will be returned.

- smapsNextAction

  This attribute can never be changed directly using this function. APS will recalculate this value before changing the record.

Chapter 11: Authentication Scheme

APS includes an Authentication Scheme. It actually serves *two* different functions:

- Enforcing/supporting case-insensitive passwords. In this mode, it can be used with virtually *any* other authentication scheme without interfering with the operation of that scheme.
- Allow the use of the OneShotPassword capability of the FPS component of APS.

For the first option, the SMAPS library needs to be configured as an *Authentication Scheme Wedge*. For the second, it is a normal authentication scheme.

This section contains the following topics:

Configuring as a Wedge (see page 341)
Configuring for FPS (see page 342)

Configuring as a Wedge

An Authentication Scheme Wedge is a piece of code that sits between SiteMinder and a "real" authentication scheme.

To configure the APS Authentication Scheme as a Wedge, configure (and test) your regular authentication scheme first. Once it has been established that your existing authentication scheme is working, edit the properties of the authentication scheme using the SiteMinder Policy Interface.

Copy the library name for the existing authentication scheme into the Parameters field and follow it with a semicolon (separating the library name from the original Authentication Schemes parameters).

Put "smaps" (without quotes, lower case) into the library name field.

The wedge is now installed.

The wedge will intercept all calls to your original authentication scheme and pass them on, returning all codes returned from it back to SiteMinder.

If the Force Case setting is used in the Configuration File, it will cause APS to change the case of the old and new passwords during the *password change* process to the desired case before performing *any* processing. This will guarantee that all passwords changed using APS will be stored in the desired case.

However, this does not guarantee that the password entered during the authentication process will be correctly entered. This task was frequently handled in the past by JavaScript code on the login form. JavaScript is not dependable, not universal, not standard, and can be overridden by the user.

The APS Authentication Scheme Wedge can fulfill this function.

For most (but not all) authentication schemes, the APS authentication scheme can determine the proper case setting for the current user, so that overrides for the Force Case setting can be used at a site and work properly. If it does not work with your authentication scheme, contact CA Professional Services.

Configuring for FPS

When used to support OneShotPasswords, the APS authentication scheme should be set up as a normal, custom authentication scheme. The library name is smaps (lower case). It uses no shared secret or parameters.

When used in this way, the Authentication scheme expects the supplied password to be the OneShotPassword set up by the FPS process. It must be used within 5 minutes of when it was assigned (this is not configurable). If it is wrong, if 5 minutes have elapsed, or if it has already been used, the authentication will be rejected.

You will have to set up a special realm so that this authentication scheme will be used. Since there is no form, normal authentication using this method is impossible.

Chapter 12: Custom Logging Extension (SmAPSLog)

APS calls a library called SmAPSLog at certain points during processing in order to record certain events. As supplied, SmAPSLog will record these events into a specified flat file. SmAPSLog is also supplied in source code so that sites can write their own logging functions.

The source code is supplied "as is", without additional support. If you choose to use the library as supplied (without recompilation), CA Professional Services will be glad to help you with any problems. If, however, you choose to build your own version, CA is not equipped to support your code.

The Policy Server installation kit installs SmAPSLog.cpp and SmAPSLog.h to your SiteMinder/Samples directory. These are the actual source files used to build the version of the SmAPSLog library that is installed by the kit.

Basically, if the SmAPSLog library exists, APS will load it and call its initialization function (called SMAPSLogInit). APS will pass a structure containing a large number of function pointers (defined in SmAPSLog.h), all initialized to NULL. If the SmAPSLog library is interested in any particular event, it must provide a valid function pointer for that event.

The SmAPSLog library must be fully re-entrant and thread-safe. It must *not* spend too much time processing a request, since this will have a direct impact on APS performance.

SmAPSLog cannot be installed while the SiteMinder services are running.

SmAPSLog must be written in C++. There are no plans to support any other languages and it is doubtful that it would even be possible at this time; it certainly would not be desirable, due to the high-performance and multi-tasking constraints of the functionality. If the logger must exist in another language, it would be best to write it as another server process and have a C++ language SmAPSLog library queue requests to it.

The SmAPSLog has some limited access to APS.cfg. All settings in the [Logging] section will be read and parsed by APS and passed to the SmAPSLog library as pointer to a class of APSDictionary that will allow the library to enumerate through the list of parameters and to retrieve specific values.

APS will automatically include the following settings in the dictionary, regardless of what is set in APS.cfg:

APS.SettingsIteration	APS rereads the APS.cfg file every time that it has changed. Each time that it is reread, the iteration number is incremented. This allows SmAPSLog to determine that the settings have changed since a prior call.
APS.Trace	The setting of the TRACE flag within APS.

Chapter 13: Custom Extensions (SmAPSEx)

APS calls a library called SmAPSEx at certain points during processing. This library exists so that CA Professional Services can extend the functionality of APS without changing the base product (which would result in a completely custom APS with relevant maintenance issues).

SmAPSEx has a number of extensibility options built in. However, it poses a possible security threat to your site, in that it has access to APS internal memory and information about the user, including clear text passwords (which may be required by SmAPSEx's custom processing).

To secure this threat, there is special internal processing and handshaking that takes place between a SmAPSEx library and APS to ensure that the library is a valid one and not a "Trojan Horse". This is why the extensibility available through the SmAPSEx library is not provided directly to customers, as is SmAPSLog.

APS does not ship or install a SmAPSEx library.

Customization of APS, beyond the various forms and configuration options, can be supplied in two ways:

- A Professional Services consultant can create a custom SmAPSEx library to perform your function. This is usually the desirable option for things like custom encryption.
- An enhancement request can be submitted for APS itself.

Please contact the Professional Services Custom Development Manager for details and pricing for each option.

Chapter 14: Utilities

APS includes a number of command line utilities and other tools to help in the configuration, maintenance and testing of APS. This section details the useful utilities. Other useful testing utilities are provided with the APS prerequisites (for example, SmPortalVfy). See the documentation for each subsystem for details of their use.

Every APS utility or tool that takes the LDAP administrator password (the -w option) now allows this password to be supplied in encrypted format. Use APSEncrypt to encrypt these passwords.

Each utility accepts, minimally, a command line switch of -? that can be used to display the utility's command line syntax. Those utilities that *require* command line options will display their syntax if run without them.

This section contains the following topics:

APSComplexity - Calculate Password Complexity (see page 347)
APSTestSettings - Check APS Configuration (see page 348)
APSEncrypt - Encrypt an Administrator Password (see page 349)
APSMailTest - Test Email Settings (see page 349)
APSPing - Test connectivity (see page 350)
APSVersion - Display Module Version Numbers (see page 350)
APSXlateTest - Test Translation Settings (see page 350)
SmCPW - Load Test Password Changes (see page 351)
APSForcePWChg - Set Force Password Change Flag (see page 352)
SmPortalVfy - Verify SmPortal Configuration (see page 353)

APSComplexity - Calculate Password Complexity

The password complexity algorithm used by APS may seem a little arcane. APS allows the user to fine-tune the scoring system that it uses.

Administrators can set the threshold, using the Complexity setting, that is required of all new passwords. Determining a value for this setting can be difficult, if not arbitrary.

APSComplexity is provided so that administrators can "get a feel" for the scoring mechanism, fine tune it, and to help them determine a proper setting for the required complexity.

APSComplexity is run with a single command line argument, the password to be analyzed. The utility will display the total complexity score and will also display how the score was determined.

APSComplexity is installed onto the SiteMinder BIN directory on a Policy Server and must be run from there (it requires access to the APS.cfg file).

APSTestSettings - Check APS Configuration

Modifying the APS Configuration File can be a daunting configuration task. When setting overrides are involved, the complexity is further increased. Add to that the fact that APS will use the *most restrictive* setting applicable, even if it is different from the most specific setting for the user. It can be very difficult to determine the exact setting that APS will use for a particular user.

APS is very consistent in how it handles configuration. It is very predictable. But it is easy for APS to wade through potentially hundreds of settings, determining limits, applicability and restrictiveness based on internal rules. It is not so easy for a human administrator.

APSTestSettings is provided to report the current configuration settings. Without a command line argument, APSTestSettings will report the default settings in effect. Output is grouped functionally. Each setting is flagged as to whether that setting supports overrides.

Command line arguments can be used to specify a particular user, in which case APSTestSettings will show the settings that are currently in effect for that user.

APSTestSettings calls into the APS libraries to obtain the settings. It does not use code separate from APS. Thus, barring output differences, APSTestSettings will always report what APS will actually see for a given user.

Command Line Options

-?	Show Help
-m[sel]	Mode selection
	s: Settings Mode (default)
	e: Evaluator Mode
	l: Lexer Mode
-q	Quiet (all modes)
-d	Show license detail (Settings Mode Only)

-#	Show line numbers (Settings Mode Only)
-r	Show attribute mappings (Settings Mode Only)
-c	Show complexity settings (Settings Mode Only)
-t	Show timings (All Modes)
-S<sections>	Explicitly control which sections are displayed (Settings Mode Only)
-u userpath	User path, such as LDAP://127.0.0.1/uid=erict,o=nds.com (Settings & Eval Modes Only)
macro=val	Context macro definitions (Settings & Eval Modes Only)

APSEncrypt - Encrypt an Administrator Password

To use certain features of APS, specifically the LDAP Rebind and Write back features, the APS Configuration File must contain an LDAP administrator's password. Putting this in the file in clear text may be considered a security problem at some sites.

APS will read the password setting from the file. If the value of the setting is encrypted, APS will decrypt it before use. If the setting is not encrypted, it will be used as the administrator password verbatim.

The APSEncrypt utility will take a password on the command line, encrypt it and output it to the screen. It can then be cut and pasted into the APS Configuration File. Thus, LDAP administrator passwords will not appear in the configuration file in clear text.

Note that such encryption is only supported within the APS Configuration File. It is not supported by command line utilities.

There is no command line utility to decrypt resulting values.

APSMailTest - Test Email Settings

The APSMail library supports a utility called APSMailTest that can be used to test the CA Mail Service's capabilities when talking to your SMTP Mail server.

This utility can also be used to test the formats of your mail files.

APSPing - Test connectivity

APSPing, new with SmPortal 5.0, tests the connectivity between a client and a SiteMinder Policy Server. It is similar to the standard ping utility.

APSPing communicates using standard SiteMinder libraries to a library called APSTransponder, located on a SiteMinder Policy Server.

APSVersion - Display Module Version Numbers

Starting with APS Version 2.2 (and all of its supporting systems), every library and executable has a product and component version number embedded within it.

The APSVersion utility will display the version information for each module name provided on its command line. Wild cards are supported.

APSXlateTest - Test Translation Settings

This utility interactively allows you to test language translation. It is part of the APSXLate package and is documented there.

SmCPW - Load Test Password Changes

Prior to Version 3.0, a utility called SmCPWTest was provided for testing password changes from the command line. Starting in Version 3.0, this functionality has been merged with SmCPW itself.

SmCPW is the CGI interface for user password changes. If run from the command line, it can also do password changes and is useful for testing APS.

SmCPW can detect whether it is running under a Web Server.

If run with no arguments or just language arguments (-L and/or -C), SmCPW will act as it does under a Web Server satisfying a GET action: it will produce the HTML required for the default form (using the language identifiers, if specified).

If invoked with a User Path, old password and new password, it will post the password change through the APS API and display the result (without any HTML wrapped around the resulting message).

The format of the command (when run from the command line) is:

```
SmCPW <options> <userpath> <oldpassword> <newpassword>
```

<options> include:

-L *<language>*	Indicates the desired language code (as the ISO abbreviation). "EN" is the default.
-C *<country>*	Indicates the desired country code (as the ISO abbreviation).
-Q *<qualifier>*	Indicates the qualifier to use for the API calls (useful for testing to multiple Policy Servers from a single client without changing SmPortal.cfg).

The *<userpath>* must be in the format LDAP://*<server>*/User DN for LDAP users, *WinNT://<server>*/*<userid>* for Windows NT users and ODBC://*<server>*/*<userid>* for ODBC users.

SmCPW uses SmPortal to communicate with the Policy Server. It can be run on any machine where SmPortal is installed and properly configured. Neither it nor SmPortal require a Web Server or SiteMinder Web Agent.

APSForcePWChg - Set Force Password Change Flag

The APSForcePWChg utility allows the setting of the smapsImmediateChange attribute in an LDAP directory for specified users.

This program accepts one or more LDAP Distinguished Names (DN's) and sets the flag for each supplied DN. The program was designed to accept either one DN per line or a standard LDIF file.

Technically, this utility is no longer needed, since the APS Blob no longer exists. However, it is useful for bulk loaders and many sites have tied in user maintenance utilities to set the flag. It is, then, provided for backwards compatibility.

The command line arguments for this utility are as follows. Command line switches are case-sensitive. The space between the command line switch and its argument is optional.

-v	Verbose mode. Additional messages will be output to the console (and can be captured using output redirection).
-n	Produce the report, but don't actually perform any updates.
-a *<attr>*	Blob attribute for APS ("audio" assumed). No longer used.
-h *<host>*	The LDAP server name or IP address. The default is 127.0.0.1, which indicates the current machine.
-p *CA Portal*	The LDAP server TCP port number. The default is 389, the default LDAP port.
-D *<binddn>*	The Administrator DN to use to log into the LDAP directory. It defaults to cn=Directory Manager.
-w *<password>*	The password associated with the binddn. There is no default. This value can be supplied in encrypted form, as supplied by APSEncrypt.
-H	Display usage information (help text)
-c	Continuous mode (do not stop on errors)
-f *<file>*	Read modifications from the specified file instead of standard input
-e *<rejectfile>*	Save rejected entries in rejectfile

Note the following:

- Continuation lines in the input file (by starting a line with spaces) are not supported, even though they are allowed in LDIF files.
- Leading and trailing spaces are not significant.
- Only one DN per input line is supported.
- If a colon (:) precedes the first equal sign (=) on any given line, an LDIF-style input line is assumed. If this is the case, the line is ignored unless the attribute name (before the colon) is dn (case-insensitive). This allows LDIF files to be used for input or just lists of DNs.
- If an LDIF-style input file is used, only those DN's that start with uid= will be processed. This is so that a full LDIF can be used, even those containing definitions for objects other than users (only users will be processed). If your directory uses an attribute other than uid for users, then LDIF-style input cannot be used.
- This program only supports LDAP directories.

  This utility was originally designed to be used to set the flag for users after a Directory load. That is, when a site bulk loads users into its LDAP directory, the same LDIF file used for the bulk loading can be fed to this utility so that the Force Change password flag can be set.

  At version 2.1 of APS, this utility has been modified so that the bindDN and password passed on the command line are set up for *rebinding*. This is a special feature of LDAP that allow this utility to support two features:

  - LDAP Referrals
  - LDAP communications breakdowns since the utility was started

  At Version 3.0 of APS, this utility has been modified to allow user DNs to be passed on the command line itself (this is obviously a low-volume solution). If so, APSForcePWChg will not go into an input loop. This is useful when the utility is to be invoked as a subprocess to another program.

SmPortalVfy - Verify SmPortal Configuration

SmPortalVfy is a new utility supplied with SmPortal/SmTransact Version 5.0. It tests the validity of the entire SmPortal.cfg file.

It requires no command line arguments.

It can be run either form the command line or under a web server (it does not require a SiteMinder Web Agent, nor does it need to be protected, though we recommend that it be protected if it will be exposed).

Chapter 15: Integrating APS with User Management Tools

Advanced Password Services does not perform User Management (Identity Management) functions. It is only involved with the application and enforcement of password content, password lifetimes and account lifetimes.

This section contains the following topics:

Integration (see page 355)
Delegated Management Services (DMS2) (see page 358)
CA Identity Manager (see page 362)

Integration

There are a number of places where APS and a User Management tool must integrate.

Out of the box, APS *does* provide the Help Desk Interface (See Help Desk Interface (APSAdmin).). However, this is *not* a full-function tool. It is really designed to simplify existing user management applications or to provide a tool to aid in APS testing.

Enabling and Disabling Accounts

APS provides a wealth of options for enabling and disabling user accounts. There are specific ways in which APS can disable accounts during processing and APS can recognize a large number of site-defined mechanisms. Any User Management application should recognize *all* of the reasons used by a site (including those configured for APS to use). These tools should be able to enable and disable accounts for those mechanisms in use at the site.

Strategies

- A site can incorporate the Help Desk Interface (APSAdmin) (see page 277) as a pop-up or linked page to perform almost all of this integration. However, if the functionality Disable is in use, the application may need to take this into account.
- A site can use the APSAdmin XML interface provided by the APSAPI to perform the same functions. This involves modifying the application to make the APSAPI calls, interpreting the XML and generating new XML for submission to the API.
- A site can read/write all of the user attributes that APS uses to enable/disable account. These attributes are described in User Directories: Schema, Storage and Capabilities (see page 189). This method is not trivial and is error-prone. It is also subject to changes to APS in the future.

Ensuring Password Validity

User passwords should always conform to the password content rules configured to APS in its Configuration File. This applies not only to ongoing user password changes, but to administrator password resets and initial passwords (during account registration) as well.

Normally, ongoing password changes are handled by the APS Change Password Interface (SmCPW) and CA highly recommends that it be used.

Out of the box, APS supports password change forms written in *any* language. It merely requires that the form POST to the Change Password Interface (SmCPW) (see page 271). Non-web based change password applications can call the APSAPI to perform this function.

Some sites want to put the change password fields onto another form, often the Change Profile form. This is usually very confusing to users. It is often better to put a "Change Password" link on a page rather than trying to incorporate both profile and password updates on the same form. Part of the problem is handling errors that might occur in one update or the other.

It is *not* possible for an application to perform all of its own password updates. The password history field is not accessible to applications; it is stored in encrypted, compressed format.

During user *registration* (account creation), the account's initial password should conform to the password content rules imposed at the site.

Initial registration creates some interesting problems for password validation:

- The password content cannot be checked against user attributes, since the user record does not yet exist.
- If the password settings are overridden in terms of user attributes or group membership, APS cannot determine the proper requirements for the new passwords' content.

  Some sites modify their User Management tool to call the APSAPI's IsPasswordValid function to determine the initial password's validity, then just saves the password (possibly updating other APS attributes as well). Sites that do this need to be aware of the above limitations.

  Another method, which has been used quite successfully, is to save the initial user record with a dummy (**very** random) password and a "state" flag, *then* call ChangePassword in the APSAPI to actually set the password. Once ChangePassword has been called successfully, the state is changed to enable the record. This bypasses the above limitations, but may leave your directory in an inconsistent state if the user abandons the process.

Another method is to set the initial password to some known (or random) value, then either give it to the user or use SiteMinder Auto-Login process (similar to how FPS can do it in the CONFIRM phase), then force the user to change their password immediately. If a site does this, it *should* use the smapsMustLoginBy attribute to ensure that this known password is used within a short timespan and the **smapsImmediateChange** attribute to cause APS to force the user to change this password (these attributes are described in Schema & Storage).

Maintaining Forgotten Password Services (FPS) Questions/Answers

The Forgotten Password Services (FPS) functionality of APS can *use* information stored in the user record to interactively help a user reset their own password. However, APS provides no interface that can be used by the Users to maintain this information.

APS must be configured to tell it the *name* of the attribute(s) to use for this customer interaction and, for certain configurations, how that information is stored.

It is important that the user management application allow users to maintain this data. It is *critical* that the APS configuration correspond to the actual method used.

Some information on strategies is contained in How Do I: Get the FPS challenge questions into the User Directory? (see page 441).

Triggering APS Events Properly

APSExpire processes events based on the value of smapsNextAction. Every time APS touches a user record, it ensures that the value of this field is accurate at that time. Some of the settings in the APS configuration file used to calculate this date may be dependent on attributes in the user record. For example:

Password Expiration={userType="Admin"} 30

Password Expiration=60

If the user's type is changed to or from Admin, the effective password expiration date changes and APS may need to update the value of smapsNextAction.

However, any User Management tool should not need to be aware of the conditions that APS uses to make these determinations.

There is a very simple way to handle this issue. *Every time* a user management tool modifies a user entry, it should delete the contents of smapsNextAction. APSExpire, scheduled to run daily, will recalculate the date on its next execution. Since user management is an extremely low volume operation, the impact on performance should be nominal.

Delegated Management Services (DMS2)

Delegated Management Services (DMS2) is a CA product, which some sites use to maintain user entries in a Directory. It supports self-registration, self-service profile management and help desk functionality.

DMS2 has been superceded by a new product from CA called Identity Manager.

DMS Workflow Library

APS includes sample source code for a DMS workflow library called APSDMSWorkflow. This file can be built and configured into the SiteMinder Policy Server to intercept updates to the user's password. The library will ensure that no passwords are saved that do not conform to the formatting requirements imposed by APS.

The library is provided in source as a *sample* only. CA Support cannot support any custom code derived from this source.

DMS2 supports a single workflow library, yet many sites need to perform their own workflow. Thus, the APS workflow is provided as sample source to show how to make the APSAPI call. Sites can then incorporate this sample code into their own workflow libraries, if desired.

Note: Prior to APS Version 4.2, this library was only supplied in binary form. The APS installer does not delete the prior version to prevent destroying a working system. However, be aware that any pre-existing copy of APSDMSWorkflow.dll or libAPSDMSWorkflow.so are not part of the APS product.

Enabling and Disabling User Accounts

Please refer to DMS Manual Chapter 7 of "Using DMS" for managing any user account as a Super Administrator or as an Organizational Administrator.

In the Managing Users Screen, after selecting a user account, you can press the button "APS Information" for the APS Information (and also enabling & disabled users).

The field names and the handling (R= Read, W= Write, C= Clear) properties of each of the APS attributes of LDAP Schema and Storage are mentioned in the following table.

APS Attribute	Prompt	Handling Props.	Comment
smapsBaseDate	Base Date	RW	Can only set to the current date
smapsLastLogin	Last Login	R	
smapsPreviousLogin	Previous Login	R	

smapsImmediateChange	Immediate Change	RWC	Can clear or set to a string that will be a comment that includes the date set
smapsDisableUntil	Disable Until	RWC	Date, time and reason
smapsDisableAfter	Disable After	RWC	Date, time and reason
smapsLastPasswordChange	Last Password Change	R	
smapsFailureCount	Failure Count	RW	Just clear, but clear is not to an empty value.
smapsLoginHistory	Login History	R	Will show number of entries and "Select to View" button
smapsExpirePasswordDays	Expire Password Days After	RWC	Number of days and comment
smapsAccountInactivityDays	Account Inactivity Days	RWC	Number of days and comment
smapsGraceLoginsUsed	Grace Logins Used	R	
smapsMustLoginBy	Must Login By	RWC	Date, time and comment
smapsGenerationalRedirects	Generational Redirects	R	Shows the number of entries and a "Select to View" button.
smapsFailuresSinceLastLogin	Failures since Last Login	R	
smapsFailuresSincePreviousLogin	Failuressince Previous Login	R	
smapsMaxFailures	Maximum Failures	R	
smapsTotalLogins	Total Logins	R	
smapsTotalFailures	Total Failures	R	
smfpsLog	FPS Log	R	Shows number of entries and "Select to View" button

smfpsLockoutCounter	FPS Lockout Counter	RC	Cleared to re-enable FPS for accounts
smapsNextAction	APSExpire trigger	C	ALWAYS clear

Note: The prompts that appear in the default set up are for English-US Locale (as defined in the file dms_en_US.properties).

Enabling a Disabled User Account

If the user account is disabled, the screen will display a message "User is Disabled".

It will also show why the user account is disabled and that may happen in the following two ways.

- The account may be disabled because it belongs to any *Disabled Group*. For example if the user is disabled due to Failure Count then it will be a member of the group called Disabled-FailureCount.
- The account may be disabled due to the unsatisfied conditions (as mentioned in APS.cfg) of the following fields
  - Must Login By
  - Disable Until
  - Disable After

If you want to enable the account, you can do it in *either* in the following ways:

- Check the Enable User CheckBox and click on Submit button.
- Check the Delete the user from the Disable Groups checkBox and modify the relevant fields as required.
- On Successful modification of the LDAP attributes you will see a SUCCESS message or else you will get an ERROR message.

Note: The date and time displayed in the DMS UI pages are all in local time, though the date and time are stored as GMT in the LDAP.

Internationalization

The Field names that are appeared in the default set up are for English US Locale (as defined in the file dms_en_US.properties). If the users want to change it, they need to define their own locale and the locale file should be present in

```
working-dir/DMS/properties/default/locale
```

Integrating Forgotten Question and Answers

Please refer to DMS Manual, Chapter 8 of "User Registration with Forgotten Password Support," for setting up for self-registering a user.

If the user is redirected to the correct fcc file, upon clicking on the link "Click here to register as a new user", the registration form will open with the questions that are defined in the property.

Format for the Property File

The value for the key "pick" signifies the number of questions to be picked.

The value for the key "questions" signifies the number of questions needed to be displayed.

The rest has the structure like QID = The Actual Question where QID is the question identifier and could be any value.

If the admin need to restrict a question to be displayed for a user while registering a new user, the admin may put a "*" in the beginning of any QID.

Do not put a "*" in the beginning of the key pick and/or questions.

Please follow the sample property file as a guideline.

Customization Required for Supporting Internationalization

In the forgottenpassword.jsp file for Language=English and Country=USA, the default lines are as follows:

```
<jsp:setProperty name="table" property="language" value="en"/>
<jsp:setProperty name="table" property="country" value="US"/>
```

For example, if the user wants to customize for Language=French and Country=Canada, these lines would be like this:

```
<jsp:setProperty name="table" property="language" value="fr"/>
<jsp:setProperty name="table" property="country" value="CA"/>
```

Please note then the name of the property file would be questions_fr_CA.properties.

APS.cfg configuration

In the APS.cfg file under [Verify], for the keyword Initial specify the special instruction as format=B**.** For example,

```
Initial=*SecretQuestion=carlicense[format=B,Pick=2,sorted]
```

CA Identity Manager

At the time of this release, integration between Identity Manager and APS has not been developed by CA. It is, however, in the planning stage and may be available in the future.

Many CA customers have integrated Identity Manager themselves in a variety of ways. Many have used the DMS2 integration guidelines above quite successfully.

Chapter 16: Using Email

This section contains the following topics:

Introduction (see page 363)
Email Templates (see page 367)
Strategies & Best Practices (see page 378)
Troubleshooting (see page 387)

Introduction

APSMail is the library used by APS to send email, via an SMTP (Simple Mail Transfer Protocol) server, based on dynamic instructions in a *mail template*.

The APSMail library is designed to be called from APS; its programmatic interface is not public. However, a command line program is provided so that email templates can be sent from the command line, ostensibly for testing.

Simple Mail Transfer Protocol (SMTP)

The *Simple Mail Transfer Protocol* (SMTP) provides a method to exchange electronic mail messages. The SMTP protocol exchange is very simple, as suggested by the protocol name. It is one of the very earliest of the internet protocols and has not changed significantly, due to its extremely wide use and installed base.

An SMTP client wishing to exchange mail with an SMTP server contacts the server on the well-known service port. The SMTP protocol is a simple ASCII, line-oriented command/response protocol. Each command is a simple command name followed by some parameters, depending on the command. Responses consist of a three-digit numeric code followed by a string explaining the response. The numeric code is easily processed by the client software and the error string can be passed on for human interpretation. Note that responses can appear on more than one line (provided the server sends the appropriate indicator).

After contacting the server, the client waits for a simple greeting message from the server. When the client receives the message, it sends a HELO (or EHLO, for *Extended* SMTP) command identifying itself. Further commands identify the sender and recipients and transfer the message data itself.

ESMTP, or *Extended Simple Mail Transfer Protocol*, is a relatively recent update to the SMTP specification that is fully backwards compatible with older SMTP clients.

APSMail communicates using RFC compliant SMTP and ESMTP, with a few optional extensions.

The original SMTP protocol is defined in RFC 821. Related RFCs include 1869 (ESMTP), 822 (Format of Electronic Mail Messages), 1521 (MIME), 1725 (POP3), and 1730 (IMAP4).

Performance

As stated above, SMTP (and ESMTP) is a line-oriented, ASCII protocol. By its very nature, it is quite slow, compared to most network protocols. While APSMail has been architected to maximize performance, this performance limitation should be taken into consideration when implementing a site.

Network Latency

Network latency can be significant. This is the time taken to locate and set up/tear down connections to the SMTP server.

Mail Services are designed to be distributed, a network of servers interconnected and routing information between them. APSMail is not a server/router, but rather just a client that submits messages into this network for delivery.

As such, it need not have access to *specific* SMTP servers, just a single server that provides an access point to the internet mail "network". In network parlance, this would be an "email gateway".

An SMTP server is a relatively lightweight service; it is for the submission of messages only, not for reading messages. As such, they do not consume large amounts of resources on the machines that they are running on. They are also relatively inexpensive, cost-wise. Microsoft Windows™ NT server, Windows 2000™ Server and Windows XP™ Server all include an SMTP server. Almost all flavors of UNIX include one as well.

To minimize network latency (and to improve fault tolerance, as discussed below), CA recommends that an SMTP server be installed and operational on the *same machine* where APSMail is running. In this configuration, all communications between APSMail and the SMTP server are local to the machine; no external network traffic is required. The SMTP server, after accepting the message, will communicate with the rest of the mail "network" transparently to APS. If this SMTP server will be used exclusively by APSMail then a site should configure the server to not allow access from other hosts.

Fault Tolerance

SMTP servers, by their very nature, are *store and forward* servers. This means that it will accept messages and hold them until they can be delivered (SMTP servers are configured to retry, etc, as necessary to keep attempting delivery).

APSMail is *not* an SMTP service. It does not have the capability to retry sending a message if the initial send fails, that is the function of the SMTP server. A well-architected site will be designed to get the message from APSMail to the SMTP service as quickly and robustly as possible and utilize the SMTP server's own fault tolerance.

To this end, CA recommends that the SMTP service and APSMail be running on the same machine (also as mentioned above for performance). Thus, network outages, machine outages, etc, will not impact APSMail delivery.

Resource Utilization

APSMail is designed to run in either a server (multi-threaded) mode or from a command-line type mode. Each has its own resource conservation requirements. APSMail is designed to be somewhat resource conservative; that is, it makes every attempt to conserve limited machine resources, memory in particular.

This may cause slight performance degradation and creates the potential for a small fault tolerance problem.

APSMail also puts no restrictions on the *number* of connections to a given SMTP server. However, some SMTP servers do restrict the number of connections from a given host. This may mean that a site has to configure their SMTP service to remove this restriction. For example, if an SMTP server restricts a given client to 10 connections, but APSMail, running under a SiteMinder Policy Server, needs to send eleven pieces of mail at once, the eleventh send may fail because the SMTP server rejects the eleventh connection (there are all kinds of timing considerations that come into play). This is yet another reason to have a "private" SMTP server dedicated to each instance of APSMail.

Configuration

APSMail is designed to be integrated with APS and can be configured in numerous ways. APSMail has a specific, predefined precedence system for resolving configuration conflicts.

At the highest level, APSMail can be configured using *environment variables*. These settings will be used unless overridden elsewhere. Most of these settings can also be made within APS. Some may also be specified or overridden within the email template files (as described later in this chapter).

APSMAIL_LOCALHOST

This environment variable can be used to override the name of the local host as transmitted to the SMTP server in the HELO (EHLO) message. If not specified, it can be overridden in an email template file. If still not set, APSMail will use the network name of the local machine.

APSMAIL_SERVER

This environment variable specifies the IP or network name of the SMTP server. It can be overridden within the APS.cfg file and again in email template files. If not specified anywhere, mail will not be sent.

APSMAIL_LOGIN

This variable specifies the email address to send to the SMTP server in the MAIL FROM command. Under the older SMTP protocol, this amounts to the login id for the SMTP server and need not match the FROM address for the message itself.

Some SMTP configurations require that this address be known to the SMTP server.

This can be overridden in an email template. If not specified at all, the FROM address of the email template will be used.

APSMAIL_LOGPATH

Specifies the path to an APS "send" log file, where information about sent (and rejected) mail is written. This value can be set within the APS.cfg file and from within a template. If not specified, no send log will be maintained.

APSMAIL_PATH

This specifies a list of directories where APSMail should look for mail template files. Multiple directories can be specified. Under Windows, directories are separated by semicolons; on UNIX, directories should be separated by colons.

This value can be set in the APS.cfg file. If not specified, APSMail will not search for files; it will try to open them as configured to APS.

APSMAIL_COMMLOG

This environment variable specifies the name of a file where all communications between APSMail and the SMTP server(s) should be logged. Note that this file can get quite large, so it should not be used except for troubleshooting. If not specified, no communications log will be kept. This is the only way to turn on the communications log.

Email Templates

APSMail sends mail as defined by an *email template file*. Everything that APSMail needs to know to construct and send the message can be defined within the template.

Template files are simple ASCII text files; they require no specific editor or tools to operate on. Text lines may be terminated with either carriage-return/linefeed pairs or merely linefeeds (so the differences between Windows and UNIX are not significant).

A template file contains the body of the message to send, along with *directives* that control how the message is sent and related operations. In addition, embedded within the text can be references to *macros* and *lookups*.

How APSMail Uses Templates

Each email template file should contain all of the information required to send the file. Some settings may or may not be required by your mail server.

As each line is read from the file, the line is examined to see if it certain text must be replaced with other text ("macros" or "lookups"). After replacement, if a line in the file begins with one of the directives described in the following sections, it will not be used as part of the message body and the text following the keyword will be used as described. Note that replacements references (macros and attributes) *are* replaced in these lines.

Everything else is considered part of the message body.

Replacement macros and attributes are also supported in file names, though a UNIX site must be *very* careful with case sensitivity in file names. This can be useful for supporting multiple languages or special mails by user type.

Macros

Macros are named values. That is, a name represents a value that is determined at a later time or in a different place.

APSMail allows the use of three different kinds of macros: Application, Define and Environment variables.

A macro reference may appear *anywhere* in the email template file *or in a template file name or path*. Before processing a file name or path, any macro references contained within the path are replaced with their value. Similarly, as each line is read in from a template file, any macro references are replaced with their current value.

Application Macros

Application macros are defined by the calling application and are usually context specific. For example, when sending Password Expiration Warning email, APS exposes a DaysLeft macro so that the email can contain the number of days before the password actually expires. See the application's documentation to determine what application macros are available in each context.

Application macros can be referenced in three ways: as %<*macro name*>%, {macro:<*macro name*>} or as {%<*macro name*>}. All are equivalent, but the first form is deprecated and is being phased out.

Using the prior example, in an APS Password Expiration Warning email, the text might read:

```
Your password will expire in {%DaysLeft} day(s). Please change it before that time.
```

Define Macros

Macros can also be defined within the template files themselves using the [DEFINE] directive (discussed later in this chapter). This is typically used with the [INCLUDE] statement to specify settings.

Define macros are referenced in exactly the same way as application macros.

```
[DEFINE SMTPServer] 127.0.0.1
[MAIL SERVER] {%SMTPServer}
```

Environment Variables

Environment variables can also be referenced within an email template file. The syntax is very slightly different. Instead of a percent sign ("%"), use a dollar sign ("$").

```
[MAIL SERVER] {$OUR_MAIL_SERVER}
```

Several alternate formats can be used for referencing environment variables. All of the following are also supported, but not recommended (the second form is deprecated).

```
[MAIL SERVER] {env: OUR_MAIL_SERVER}
[MAIL SERVER] %ENV_OUR_MAIL_SERVER%
```

Lookups

APSMail also supports *Lookups*. When APSMail encounters a lookup, it calls back into APS to get a value for the lookup. This is typically done to retrieve values from the user record.

Note that lookups are supported by the *APS*, not APSMail directly.

Lookups are referenced within curly braces. There are two forms:

```
{<attribute>}
```

This form generally represents a raw data field from the default user record. For example, {cn} would typically be replaced with the value of the cn attribute of the user.

Another form of a lookup reference is

```
{<qualifier>:<attribute>}
```

This form has specific, application-defined uses. For example, if there is more than one user involved, an application might have a qualifier of "Administrator", in which case "Administrator:cn" refers to the common name of the administrator instead of the common name of the user.

APS docs not currently support any qualifiers.

Well Known SiteMinder Attributes

In most cases, if the first form of lookup is supported, you can use the *well known SiteMinder* attributes as well as attributes that reside in the user directory itself.

The Well Known attributes are documented in the SiteMinder manuals and include such things as the user's IP address (only useful if sending mail while a user is currently online).

Directives

The email template file contains *directives* that tell APSMail how to send the message. Each directive appears on a single line and must start at the beginning of the line. The entire line is considered part of the directive and will not be included in the message body. Anything that is *not* a directive will be included in the message body.

All directives are surrounded by square brackets ("[" and "]") and start at the beginning of the line. Directives may appear anywhere in the file, but the file is only processed once, so directives that have an effect on file processing will only affect from the point where they appear in the file.

Some directives may appear more than twice and will be processed for each appearance (such as the ATTACH directives). Other directives should only appear once; if such a directive appears more than once, the last instance will be used.

A directive *can* reference macros and lookups.

Template Processing Control

Including Another Template

Example:

```
[INCLUDE] <path>
```

The Include directive tells APSMail to include the file specified and process its contents as though it appeared at that same point in the original template. Included files *may* include other files, but APSMail will detect if the same file is included recursively and will issue an error.

The *<path>* is not processed in any way (the search paths are not used), it is merely opened directly. Most systems should use an environment variable (as a macro) to identify the location of this file.

```
[INCLUDE] {$MASTER_MAILFILE}master.etf
```

Defining Macro Substitutions

Example:

```
[DEFINE <name>] <value>
```

This directive allows a site to create definitions of symbols (*<name>*) representing a value (*<value>*). If *<name>* is represented elsewhere in the file (below the point where the [DEFINE] directive appears), the reference is replaced with the value. For more information, see Define Macros.

Lookups can be used in Macro definitions.

Macros are only "visible" from the point at which they are defined.

Comments

Example:

```
[COMMENT] <text>
[REMARK] <text>
[REM] <text>
```

Any line containing this directive is completely ignored. It is useful for placing comments into the template file to ease maintenance or to temporarily disable other directives in the file.

```
[REM] Last updated by John Smith on 4/1/2003
[REM] Test sending synchronously - [ASYNCHRONOUS]
```

Logging Path

Example:

```
[LOGPATH] <path>
[LOG PATH] <path>
```

This directive specifies a path where the result of this send should be logged. This overrides both the application and environment variables. It is not often used. It might be used, however, if different templates are to be logged to different files.

Server Communications Control

Specifying the Mail Server

Example:

```
[MAILSERVER] <server>
[MAIL SERVER] <server>
[SMTPSERVER] <server>
[SMTP SERVER] <server>
[SERVER] <server>
```

This directive overrides both the environment variable and any application-specified mail server to tell APSMail where to find the SMTP server. If no SMTP server is specified, then no message will be sent.

This directive is frequently used in a centrally included template file.The *<server>* may be specified as an internet machine name or an IP address. It *may* also be followed by the port to use, separated from the server address by a colon, as in smtp.acme.com:50 or 127.0.0.1:50. It is very unusual to use a port other than the default SMTP port.

Communications Timeouts

Example:

```
[TIMEOUT] <seconds>
```

This directive specifies the amount of time to wait for mail server responses. By default, APSMail will wait up to 30 seconds for a response from a mail server. This value *must not* be zero.

Shorter values do not improve performance in any way.

If this value is too low, it may cause APSMail to prematurely decide that a server is not responding and will result in errors. If too high, real errors from the server will not be detected for some time.

```
[TIMEOUT] 60
```

Local Machine Name

Example:

```
[LOCALHOST] <hostname>
[LOCAL HOST] <hostname>
[STATION] <hostname>
[HELLO] <hostname>
[HELO] <hostname>
[EHLO] <hostname>
```

This directive specifies the name to use on the mail server greeting message. Per the SMPT RFC, this should be the name of the local host. If not specified, APSMail will look up the name of the local host. This can also be specified in an environment variable, but any value appearing in this directive will override an environment setting.

Mail Server Security

SMTP supports a single security option. ESMTP requires the MAIL FROM command (like SMTP), but also provides two other alternatives for security. At this time, APSMail does not support these additional security options. If the SMTP service is local to the APSMail service, these options are not needed.

Mail Server Login

Example:

```
[MAILFROM] <email address>
[MAIL FROM] <email address>
```

This directive specifies the email address to send to the mail server in the MAIL FROM command. All servers require MAIL FROM. Some require that the address be known to the server or that the address be in the same internet domain as the server.

The value used for MAIL FROM can come from a variety of places if not specified in an explicit directive. The APSMAIL_LOGIN environment variable will be used if this value is not set.

If it is not explicitly set by the environment variabl and this directive does not appear, then the value from the SENDER or FROM directives will be used.

```
[MAIL FROM] administrator@acme.com
```

Basic Message Control

The FROM Address

Example:

[FROM] *<email specifier>*

This directive specifies the FROM address of the message. If not set, the SENDER will be used. This value is used in the FROM header of the message. If different from the SENDER, some email clients will show one or the other or a combination of both the FROM and SENDER headers.

This is *supposed* to be (according to the RFC) only used to display who the message was from. However, its misuse has caused some email clients to perform special processing, if different.

The *<email specifier>* can be just an internet email address or can be something like:

```
"John Doe" jdoe@acme.com
```

The Sender

Example:

[SENDER] *<email specifier>*

This directive specifies the address of the sender of the message. If not set, the FROM address will be used. This value is used in the SENDER header of the message. If different from the FROM, some email clients will show one or the other or a combination of both the FROM and SENDER headers.

This is *supposed* to be (according to the RFC) used to record who the actual sender was, for purposes of error handling and replies. However, its misuse has caused some email clients to perform special processing, if different.

The *<email specifier>* can be just an internet email address or can be something like:

```
"John Doe" jdoe@acme.com
```

Message Subject

Example:

[SUBJECT] *<text>*
[RE] *<text>*

This directive tells APSMail the subject line to use for this message.

[SUBJECT] Your password will expire in {%*DaysLeft*} day(s)

Addressing the Message

Example:

```
[TO] <email specifier(s)>
```

This directive tells APSMail who to send the message to directly (as opposed to copies). More than one address can be specified, separated by semi-colons.

These addresses are used in the TO header of the message as well.

Each *<email specifier>* can be just an internet email address or can be something like:

```
"John Doe" jdoe@acme.com
[TO] {mail};{pagermail}
```

Sending Carbon Copies

Example:

```
[CC] <email specifier(s)>
```

This directive tells APSMail who to send copies of the message to. More than one address can be specified, separated by semi-colons.

These addresses are used in the CC header of the message as well, so they are visible to all message recipients.

Each *<email specifier>* can be just an internet email address or can be something like:

```
"John Doe" jdoe@acme.com
[CC] helpdesk@mycompany.com
```

Sending Blind Carbon Copies

Example:

```
    [BCC] <email specifier(s)>
```

This directive tells APSMail who to send copies of the message to. More than one address can be specified, separated by semi-colons.

These addresses are *not* visible to message recipients.

Each *<email specifier>* can be just an internet email address or can be something like:

```
"John Doe" jdoe@acme.com
[BCC] audit@mycompany.com
```

Intermediate Message Control

Specifying the Reply To Address

Example:

```
[REPLYTO] <email address>
[REPLY TO] <email address >
```

This directive sets the REPLY-TO header in the message. If a reply is sent to the message, the reply is to go to this address instead of to the sender.

The *<email address>* should be an internet email address.

```
[REPLY TO] helpdesk@mycompany.com
```

Specifying Where to send Delivery Errors

Example:

```
[ERRORSTO] <email specifier>
[ERRORS TO] <email specifier>
```

This directive sets the ERRORS-TO header in the message. *Some* mail servers will use the email address specified in this header to send delivery error messages to. If this directive is not specified, errors are sent to the sender or reply-to address.

The *<email address>* should be an internet email address.

```
[ERRORS TO] helpdesk@mycompany.com
```

Requesting Read Receipts (Return Receipts)

Example:

```
[RETURNRECEIPT] <email address>
[RETURN RECEIPT] <email address>
[RETURNRECEIPTTO] <email address>
[RETURN RECEIPT TO] <email address>
```

This directive specifies that *read* receipts are requested from each of the recipients of the message. Read receipts are only expected to be sent when the message is marked as read by the user of that client.

There is no guarantee that the client program will actually send such a receipt. Email clients can usually be configured to suppress, prompt or automatically return such a receipt. This directive therefore can only *request* such a receipt.

The *<email address>* is optional. If not included, then no specific address will be included with the message and any return receipt, if sent, will be sent to the sender of the message. It should be just an internet address, not a full email specifier.

```
[RETURN RECEIPT] audit@mycompany.com
```

Requesting Delivery Receipts (Disposition Notification)

Example:

```
[DISPOSITION] <email address>
[DISPOSITIONNOTIFICATION] <email address>
[DISPOSITION NOTIFICATION] <email address>
[NOTIFICATION] <email address>
[DISPOSITIONTO] <email address>
[DISPOSITIONNOTIFICATIONTO] <email address>
[NOTIFICATIONTO] <email address>
[DISPOSITION TO] <email address>
[DISPOSITION NOTIFICATION TO] <email address>
[NOTIFICATION TO] <email address>
[NOTIFY] <email address>
```

This directive specifies that *delivery* receipts are requested from each of the recipients of the message. Delivery receipts are usually sent by the email SERVER that delivers the message to the email client. While servers can be configured as to whether they return these receipts or not, it is more likely that such a request be honored.

There is still no guarantee that such a receipt will actually be sent. This directive therefore can only *request* such a receipt.

The *<email address>* is optional. If not included, then no specific address will be included with the message and any receipt, if sent, will be sent to the sender of the message. It should be just an internet address, not a full email specifier.

```
[NOTIFY] audit@mycompany.com
```

Specifying the Sender's Organization

Example:

```
[ORGANIZATION] <text>
```

This directive sets the ORGANIZATION header in the message. This is a relatively new header in the SMTP specification. Most clients just display its contents.

```
[ORGANIZATION] Acme, Inc.
```

Specifying the Message Priority

Example:

```
[PRIORITY] LOW
[PRIORITY] LO
[PRIORITY] NORMAL
[PRIORITY] NORM
[PRIORITY] HIGH
[PRIORITY] HI
```

This directive sets the message priority. This has little to do with message handling at any point in the delivery of the message. It is typically used by the email client reading the message to determine how to display it.

```
[PRIORITY] HIGH
```

Advanced Message Control

Message Formatting

Example:

```
[TEXTMODE] PLAIN
[TEXTMODE] ENRICHED
[TEXTMODE] HTML
```

This directive tells APSMail the format of the message body. APSMail will encode the body as required by this directive and set the MIME type of the message accordingly. If this directive is not specified, PLAIN is assumed.

APSMail is not involved with the formatting of the message body, this setting merely tells the system what MIME type to use. The development of mail content using ENRICHED or HTML formatting is completely up to the site.

Alternative Character Sets

Example:

```
[CHARSET] <ISO-character-set-identifier>
```

This directive tells APSMail what character set to associate with the message body and to use for plain text character quotation. By default, ISO-8859-1 is used.

APSMail does no edit checking on this value.

Custom X-MAIL Headers

Example:

```
[X-<name>] <value>
```

The Electronic Mail Format specification allows for extensions to the message headers in the form of what are called X-Headers. These headers are in the form of X- + *<some name text>* + ":" + *<some value>*. If a client does not know how to process such a header, it is merely carried along and ignored.

APSMail automatically inserts a single X-header, called X-Mailer, which identifies the version of APSMail used to generate the message.

Additional custom headers can be included in the message as well using this directive. Header names in this context *are* case-sensitive.

A blank <value> will suppress a particular header, so, for example,

```
[X-Mailer]
```

will suppress the sending of the APSMail header.

Message Body

Anything *not* on a directive line contained in a template file (or included template files) will be added to the message body. *This includes blank lines!*

Strategies & Best Practices

There are as many ways to set up your email directories as there are to set up a computer. This section discusses several ways to do it and the advantages of each.

The bottom line is that APS (actually, APSMail) must be able to locate its mail files. Each event that can be responded to via email has, in the APS.cfg file, the name or path of an email file. A set of directories where these files reside can also be configured in APS.cfg.

The simplest configuration is to just specify the entire path of each file for each event. Put each file wherever it is most convenient, whether they exist on the same directory, a directory specific to APS email files, or anywhere else is immaterial. APS can find them, since the full path is specified in the configuration.

This configuration, while the simplest, is also the least flexible and, possibly, the most difficult to maintain. A given event will be always responded to using the same file, with the same content, regardless of the user, user type or situation. The only controlling factor for the content is the event itself.

In addition, the files could be scattered across your drive, creating confusion, as they may intermingle with other files.

CA recommends that a site create a specific APS email directory, possibly with subdirectories (discussed later). All files should exist within (or under) this directory. The directory should be specified as the mail directory in the APS.cfg file.

Many sites limit their thinking of email to just sending mail to users. There are other uses as well. For example, sites can send mail to administrators when expiring certain users so that a customer service department can proactively contact the user. Another example: using third party tools, administrators can be paged if a user is disabled due to a failure count problem (thus alerting the administrator to a possible denial of service attack).

A useful technique is to embed a user attribute into the file name or the path. This is most used to send different files based on the user's language. Thus, the mail directory might be specified as:

```
/mail/{preferredLanguage}/
```

In this case, APS will replace *{preferredLanguage}* with the value stored in the target user's profile (such as EN or FR).

This technique can also be used to send different files based on other information about the user, such as the user's type or the Line of Business of the user. This is usually done within the file name rather than the directory, such as:

```
PasswordWarning_{userType}.email
```

When the user's attribute userType contains Administrator, the file PasswordWarning_Administrator.email will be sent.

Some words of warning when using embedded attributes:

- If the user record does not contain the attribute or the value is blank, the reference will be replaced with a null value. Thus, if no value for userType is stored, the file name sent will be PasswordWarning_.email.
- On Unix, file names are case-sensitive. Thus, a site must be very careful when storing values in the user record. Since a file should match for each possible value (because of the prior bullet), this is not a problem, since values are usually entered from a drop-down list.

SMTP (Mail) Servers

The purpose of APSMail is to dynamically format messages and deliver them into a mail system; it is *not* APSMail's purpose to actually deliver the mail or to track it once routed to an SMTP server. SMTP, by its very nature, is an asynchronous delivery method and APSMail is *not* part of the SMTP "network".

SMTP servers can be configured to retry sending, log errors, archive mail, etc., these are not the responsibilities of APSMail. APSMail will format the message and the network architecture should be designed so that APSMail can hand off that message to an SMTP server as fast and as reliably as possible.

SMTP servers are typically free of charge. UNIX and Windows Servers include them. SMTP servers do not usually consume large amounts of resources (full email servers can, but not just SMTP servers).

CA recommends that an SMTP server be installed on the same machine with APSMail, if at all possible. This will completely eliminate the network between APSMail and the SMTP server, reducing the network latency to almost nil and removing a point of failure from the entire data flow.

From there, the SMTP server can be configured to retry, etc, to provide maximum fault tolerance in the event that the internet is inaccessible, etc.

Once APSMail delivers the message to the SMTP server, the connection is dropped and the APSMail code moves on. The result is logged (if logging is turned on) and the message is completely forgotten. If the mail is not delivered to the ultimate recipient, or there is a delay, the SMTP network is responsible for notification, not APSMail. You can configure, using the email templates, the information that you want APSMail to pass to the SMTP server how you want errors reported, but all such reports are through the SMTP system itself and do not involve APSMail.

Using [REM]

The [REM] (or [COMMENT], if you prefer) directives consume a small amount of disk space and a minimal amount of processing time, yet they are easily the most useful and used (yet not used enough!) directives available for email templates.

Any line beginning with [REM] will be completely *ignored* by APSMail. This allows a site to put information about *why* certain directives have certain settings and allow a site to keep a change log for that file.

Comments are *not* sent to the SMTP server and are not placed into the LOG file.

```
[REM] THIS IS A MASTER EMAIL TEMPLATE THAT
[REM] CONTAINS MACHINE-SPECIFIC CONNECTIVITY
[REM] DIRECTIVES ONLY
[REM]
[REM] History:
[REM]    04/14/03 ET          Initial Coding
[MAIL SERVER] 127.0.0.1
[LOCALHOST] PRODUCTION
[MAILFROM] SiteMinder System
```

IUsing [INCLUDE]

The [INCLUDE] directive, available for email template files, can be used to "hide" machine-specific details about email delivery. Thus, such things as the SMTP server, the login information for that email server, and return/delivery receipt information, can be localized and stored separately on each machine, then *all* of the email templates, in all environments (development, test, staging and production) are identical, only a master INCLUDE file differs.

```
[INCLUDE] {$MASTER_EMAIL_TEMPLATE}
[TO] {mail}
[FROM] SiteMinder Application
[SUBJECT] Just Wanted You to Know
Dear {cn},
Just wanted to welcome you to our system.
```

This master include file is sometimes referenced by an environment variable within the template actually used. This shields the email templates from differences in the file system structures:

Internationalization/Localization

There are actually two different ways to perform localization of mail. Both methods *require* multiple template files, one for each locale to be supported by the site.

The first mechanism, which was the only mechanism available prior to the release of APS 5, was to use different email templates, each entirely self-contained and each in a different supported language. Sites then place subdirectories under the mail directory for each language, named with the ISO code for that language.

In each application, the email template to use are then referenced with the ISO code to use as part of the file name, *specified as an attribute lookup*, as in:

```
Password Expired Email={preferrendLanguage}/PExpire.etf
```

During the file resolution (assuming that the application support attribute lookup), the reference to *{preferredLanguage}* would be resolved to whatever (assumed to be ISO code) is stored in the (LDAP) user record and the correct email template would be used (assuming it is available).

The second method, newly available to APSMail, is very similar, but allows sites to centralize the directives used for the mail with liberal use of the *[INCLUDE]* directive.

```
Password Expired Email=PExpire.etf
```

```
[INCLUDE] {$MASTER_EMAIL_TEMPLATE}
[TO] {mail}
[FROM] SiteMinder Application
[INCLUDE] {preferrendLanguage}/PExpired.etf
```

Then each include file contains the locale-specific information.

```
[SUBJECT] Just Wanted You to Know
Dear {cn},
Just wanted to welcome you to our system.
```

```
[SUBJECT] Pour votre Cognaissance
Bonjour {cn},
Bienvenue à notre système.
```

Addressing Mail

Neither APS nor APSMail "know" users' email addresses. When you specify where email is to be sent (using the [TO] notation described later in this chapter), you can specify an explicit email address, such as:

```
[TO] APS-Support@netegrity.com
```

Or you can reference a user attribute, using the "{" and }"} notation, such as:

```
[TO] {mail}
```

Just keep in mind that an attribute called mail will be used. If there is no such attribute {mail} will be used as the mailing address (which will probably not work).

If your site uses multiple User Directories and the name of the attribute is not consistent across directories, use the **[MAPPING]** section of the APS Configuration File to create a common name for all directories and use the common name here.

Return Addresses

Return and From addresses can be specified in the body of the mail file. These addresses may or may not actually exist (though if a reply to address does not exist, end users will receive an undeliverable mail notification if they reply to mail).

It is very useful to specify generic *alias* addresses for these values. Thus, the Reply To address could be *password-support@MySite.com*. Within the site's mail server, this address would be an alias pointing to a "real" email address, representing the person actually performing the support. It is better to use an alias than a real address for this purpose, so that the support person can be changed without changing the email files and any mail that has already been sent will still point to a valid address.

Handling Returned Mail

Sometimes the mail address for a user will be invalid. It may never have been entered correctly or the user's address may have changed and the change has not been reflected in your directory. In these cases, email will be undeliverable (it will "bounce"). A site should be able to gracefully handle this condition.

The easiest way to handle these cases is to set up a dedicated mail server to handle APS mail (if the site has enough volume to warrant it). This server should have a legitimate postmaster defined. Bounced mail notifications will be sent to the postmaster (which is usually an alias rather than a specific email address). A support representative can review these notifications and address each one individually. It is impossible for APS (or APSMail) to handle this condition itself, since mail is submitted to the mail server for delivery at some point in the future. By the time that it is discovered that the mail is undeliverable, APS has long advanced to other things. Sometimes, such detection may not take place for *days*!

The Errors-To feature *can* be used, but be forewarned, many mail servers do not support this feature.

Mail Macros

Each mail file should contain all of the information required to send the file, with the exception of the information about the mail server (which is stored in the APS Configuration File or environment variables).

APSMail has a very simple, yet strict, format for email template files.

The template file may contain references to user attributes and/or macros. APS fully supports this feature of APSMail as follows:

As each line is read from the file, the line will be examined. Any text surrounded by curly braces ("{" and "}") will be treated as an attribute query (lookup). That is, the text *within* the braces will be used as an attribute lookup (using code similar to the SiteMinder userattr response) and then replaced (along with the braces) with the returned values. Note that if the user does not have a value for the attribute, the braces and attribute name will be removed. They may appear anywhere in the line and are case-insensitive. They may **not** span multiple lines, though the replacement text may contain embedded newline characters.

APSMail, called by APS, supports macro expansion. These are values supplied by APS and are not stored in the User Directory. If they appear in the line, they will be replaced by the appropriate value. They may appear anywhere in the line and are case-insensitive. Macros are surrounded by percent signs ("%").

The macros supported by APS include:

%DAYSLEFT%	The number of days left before the user's password expires (calculated by APS). This value is only available when a password change warning is being issued.
%PASSWORD%	For password change confirmations only, this macro is replaced with the user's new password. In general, this macro should not be used.

The macros supported by FPS include:

%MESSAGE%	For error mail, this contains the text of the error message.
%PASSWORD%	During confirmation, this is the clear text password for the user. You cannot use *{userPassword}* (which would return the LDAP value), since the returned value would be encrypted.
%HALFPASSWORD1%	The first half of the new password, in clear text. One secure way to transmit the new password to the user is to send half of the password via email and to display the other half on the browser. This macro allows you to include the first half of the password via email.
%HALFPASSWORD2%	The second half of the new password, in clear text. One secure way to transmit the new password to the user is to send half of the password via email and to display the other half on the browser. This macro allows you to include the second half of the password via email.

Replacement macros and attributes are also supported in file names, though a Unix site must be very careful with case sensitivity in file names. This can be useful for supporting multiple languages or special mails by user type.

This example might be used as the English language warning mail under APS.

```
[FROM] SiteMinder Administration
[TO] {mail}
[RE] Your password to our site will expire
[REPLYTO] HelpDesk@oursite.com
Your password on our site will expire in %DAYSLEFT% days. In order to maintain
your access to our site, you must change your password before it expires.

To change your password, you can go to http://www.yoursite.com/CPW/SmCPW.exe
```

Best Practice Recommendation

Many APS sites have found a *considerable* advantage to putting a dedicated SMTP (mail) server on *each* SiteMinder Policy Server. Some of the benefits include:

- SMTP servers are generally free.
- APS does not spend any time resolving the network address (or spend any time "on the wire") because the server is at 127.0.0.1. This can increase performance *significantly*.
- There is no single point of failure for mail.
- Returned mail and delivery errors are located in the dedicated server(s) logs, eliminating the need to examine a corporate mail server's logs.
- The mail server need not have any incoming mailboxes, eliminating maintenance concerns.
- The mail server must support mail relaying. Some corporate servers are configured so that they will not perform relaying.
- Firewalls can be used to protect the mail server. Except for returned mail (which can be redirected elsewhere, if desired), there is no need for a "hole" in the firewall to access the mail server. Mail is outgoing only.
- Most SMTP servers can be configured to only allow connections *from* specific machines. A dedicated, local, SMTP server can be configured to only accept connections from the local machine. This can improve performance (slightly) and security (significantly). Note that this bullet eliminates the need (entirely) for authentication to the SMTP server, which is not supported by APSMail.
- SMTP servers typically limit the number of connections allowed *from* a specific client (usually 10 connections). If more than 10 threads are sending mail, this would cause APS mail sends to fail (this has never been reported in a production environment, but is possible). Such a dedicated server can have this restriction raised or removed without impacting security.

Troubleshooting

The SiteMinder Console Log

APSMail writes to the SiteMinder Policy Server log when running under the Policy Server.

APSMail will log errors and confirmations to this log. The first troubleshooting action should be to examine this log, if available. Error messages are self-explanatory and are usually followed immediately by a recommended action.

This is usually good for finding mail options and server handshake problems, such as invalid email addresses, server connectivity issues and server login problems.

The Send Log

If the LogPath is specified to APSMail, each transmission is logged as a block entry describing where the message was sent, the content of the message and the ultimate result of the send.

This log is good for reviewing the results of the transmission and the formatting of the body of the message. Message headers are also shown.

```
***********************************************************************
Sent:        Tue, 1-Apr-2003 at 17:03:52
SMTP Server: belmail.netegrity.com (accepted by SMTP server)
SMTP Login:  "Eric Theis"<erict@netegrity.com>
=======================================================================
Date: Tue, 01 Apr 03 22:03:51 GMT
From: "Eric Theis" <erict@netegrity.com>
Sender: "Eric Theis" <erict@netegrity.com>
Subject: Test Email from NPSMail
To: Eric The Kid <erict@netegrity.com>
X-Mailer: <Netegrity NPS Mailer V2.0cv1>
MIME-Version: 1.0
Content-Type: Multipart/Mixed; boundary="__NextPart__-p1KaLmIg2kB7fZp5Tm1q"

--__NextPart__-p1KaLmIg2kB7fZp5Tm1q
Content-Type: text/html; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable
Content-description: Mail Message Body
-----------------------------------------------------------------------
This is a test of the new Email system.<br>
There should be a single file attached to this email, a copy of this
template<br>
.......................................................................
Attached text/plain file: .\Test.email
```

For each message, a block like the following is recorded.

The first section of each log block is information about the actual transmission. This block identifies the server and connection information and the result of the send.

The second block identifies the message header and includes such items as who the message was from and who it was sent to.

The third block is the formatted message body.

Further blocks identify the attached files, if any, and the identified MIME types of each file.

The Communications Log

APSMail will write a log containing detail about its actual communications with the mail server. This is turned on by setting an environment variable called APSMAIL_COMMLOG to identify the file to which the log should be written.

This communications log can get very large, since it will show the transmission of attachments (which can be arbitrarily large) and will show encoded data. Generally, this log should *not* be enabled, unless troubleshooting problems with server communications.

Formatting Problems

The best way to troubleshoot formatting problems is to start with the Send Log, since it will show the formatted message as sent to the mail server. The message might be base 64 encoded on transmission, in which case the communications log will not be human readable.

Most formatting problems are caused by either using the wrong TEXTMODE directive or by using macros/lookups that can change the visual formatting.

Delivery Problems

If APSMail is showing that the message was accepted by the server, yet the recipient does not get the message, then troubleshooting the mail system is required.

APSMail passes the message to the SMTP server and then forgets about it. All delivery is done by the mail server, not APSMail, at that point. If the SMTP server accepts it, any problems with delivery will either be recorded by the mail server(s) or through delivery notification.

Limitations

- APSMail does not support SMTP server login (authentication) at this time.
- APSMail does not currently support mail attachments of any kind.
- All sending of mail to the SMTP server is *synchronous*. That is, the calling code waits for the transfer to complete before continuing.
- APSMail puts no restrictions on the *number* of connections to a given SMTP server. However, some SMTP servers do restrict the number of connections from a given host. This may mean that a site has to configure their SMTP service to remove this restriction. For example, if an SMTP server restricts a given client to 10 connections, but APSMail, running under a SiteMinder Policy Server, needs to send eleven pieces of mail at once, the eleventh send may fail because the SMTP server rejects the eleventh connection (there are all kinds of timing considerations that come into play). This is yet another reason to have a "private" SMTP server dedicated to each instance of APSMail.

Chapter 17: Internationalization

This section contains the following topics:

Introduction (see page 391)
SmCPW, APSAdmin & Forgot (User Interfaces) (see page 393)
Redirection (see page 394)
Email (see page 395)

Introduction

Note: As every developer that has ever attempted it knows, internationalization is a very complex subject. This section is not intended to be a treatise on internationalization; it is intended only as instructions and guidelines for using the internationalization features of APS to localize its functionality at your site.

Internationalization is the capability of a program to be localized, that is, to vary the display of information based on where the user is and what the user (and the user's browser) is capable of. Typically, this is primarily equated to language translation, but may include more than that.

For example, most Canadians speak English, but Canada is not divided into "states"; instead, it is divided into "provinces". To vary the output for Canadian users to display a prompt as "Province" instead of "State" is part of localization. Whereas internationalization is the capability, localization is the actual implementation for a specific locality; a locality being defined (at least for our purposes) as a combination of language and country code.

Every piece of text or graphic presented to a user by APS can be modified by a site. For our purposes, the word translation can be used, though it is not entirely accurate in the traditional sense, just as localization is not strictly language translation.

Every error message, prompt, and result is stored within the APS programs as a key. With each key is a default translation. During processing, APS asks the APSXlate library for text associated with the key. If there is no text associated with a given key, APS will use the default text. All default text is in US English.

APSXlate uses translation files to look up text for each key. When APS asks for a translation, it tells APSXlate what the user's language and country settings are. APSXlate then will select the appropriate files to use to look up the correct translation.

These translation files are stored on both the Policy Server and on the Web Server. Under Windows, they are stored in a directory called Language that must exist as a subdirectory under where APSXlate.DLL is located. Under Unix, APSXlate uses an environment variable to locate the Language directory.

Under the language directory, there are subdirectories for each language **and** locality. For example, there should be directories called EN and EN-US. Files located in the EN subdirectory are for general English and files in EN-US contain overrides that are specific for English in the United States. APSXlate loads files from the EN directory first, then loads files from the EN-US directory, the EN-US translations overlaying any duplicate entries loaded from the EN directory.

There is actually a total of up to four files loaded by APSXlate for any translation; "Common" files and files that are specific to an application. The details of how files are loaded and reloaded are detailed in the APSXlate documentation. What you need to know here is the names of the files that APS uses.

SmCPW (the web-server-side CGI program users access to change their passwords) uses a file called SmCPW.lang, stored in the language directories on the Web Server. APS uses a file called APS.lang, stored in the language directories on the Policy Server. There may be multiple copies of each of these files, one for each locale supported by the site.

Within each file, there are definitions for each key and the translation for that key. To change the text associated with a given key, locate the key within the file (for the target language) and change the associated text.

The keys should never be changed, since the key is what is used to locate the text. Only the text should ever be changed.

Since every text string is "translated", even from English to English, you can change the actual text for every message in the system by modifying the translation files.

The files, as supplied, contain all default strings. If you were to delete the translation files (this is not recommended!), the default text would still be displayed to the user, because it is actually hard-coded into APS. The advantage is that if the user's locale has no translation files (for example, he wants Greek), the default English translation will still be used.

There are three points of contact between APS and the user:

- SmCPW, where the user can change their password.
- Redirections
- Email

SmCPW, APSAdmin & Forgot (User Interfaces)

SmCPW (change password interface), APSAdmin (help desk interface) and Forgot (part of the FPS interface) can infer the user's locality from information passed from the user's browser. The rules for determining the browsers language are:

1. Does a header (SiteMinder response) called HTTP_LANGUAGECOOKIE exist?

   If so, its value is the *name* of a cookie to use. The cookie should contain the desired language information using the same format as the HTTP accept-language header. It will be processed instead of the value of the accept-language header. No other responses or cookies are examined.

2. Does a header (SiteMinder response) exist called SM_LANGUAGE (and SM_COUNTRY)?

   If so, their values indicate the actual ISO language and country codes to use. No additional processing will occur.

3. The HTTP accept-language header will be used.

4. If a header (SiteMinder response) exists called SM_LANGUAGE1 (and SM_COUNTRY1), it will be appended to the HTTP accept-language value retrieved in step 3 above.

The resulting string (either from the cookie or the HTTP header) is evaluated from left to right to find a supported language/country combination. If any combination has full support (both language and country), it will be used, otherwise, the first value with a language only match will be used.

If no match can be found, US-EN will be used.

SmCPW

The default form that SmCPW builds uses translations for all prompts, window titles and messages. These all come from the SmCPW.lang file. To support multiple locales, copy the default SmCPW.lang file to each required locale (directory) and translate the messages for the appropriate language.

To modify a default in English, just modify the English language file.

Of course, your site can always use a custom form and handle the form localization yourself.

Error and confirmation messages used by SmCPW originate from either the SmCPW.lang file or the APS.lang file, depending on where the message actually originated. Just locate the desired message to translate and change it, either in the English language file or in the language appropriate for the change.

For all such error messages, regardless of where it originates, if it "looks" like a URL, SmCPW will redirect the user rather than displaying the message. Thus, you can do your own localization by redirecting messages to known Web Pages that do their own translation.

If you want to entirely do your own translation, you should set the SM_LANGUAGE and SM_COUNTRY responses for SmCPW to EN and US. Supply your own form (that adapts to the user's locality) and modify all of the EN-US message translations to URLs instead of text. In this case, SmCPW will display no messages itself, it will just redirect the user to pages that can adapt themselves. The full capabilities of SmCPW are detailed elsewhere in this document.

APSAdmin

The forms that APSAdmin builds use translations for all standard elements, window titles and messages. These all come from the APSAdmin.lang file. To support multiple locales, copy the default APSAdmin.lang file to each required locale (directory) and translate the messages for the appropriate language.

Section labels and field prompts are specified in the [APSAdmin] section of APS.cfg and are not normally translated. If translation for these elements is desired, put a pound sign ("#") in front of the prompt definition in APS.cfg. This tells APSAdmin that the prompt should be translated. The key for translation is the full prompt value (including the "#") and the default translation has the "#" removed.

To modify a default in English, just modify the English language file.

Redirection

APS does not actually display its own messages when an authentication-time event occurs. Instead, it redirects the user to particular pages. There are two ways to localize this communication.

One way is to create a "generic" page for each type of redirection that performs its own adaptation for the user's locality, much the same as SmCPW.

The other way is to write specific pages for every event for every locality, then perform *redirection overrides* based on some value in the user's record (described elsewhere in this document). This mechanism is **not** recommended because of the following case: Suppose the user typically accesses your site from her office in Russia. The preferred language in her Directory entry indicates Russian (which uses a Cyrillic font). Now suppose she's visiting New York and accesses some sort of kiosk browser that does not have a Cyrillic font installed. Since the selection is based on the directory entry instead of the browser entry, the page will try to display Cyrillic text on a browser that does not support it.

Email

Email should always be sent using the language stored in the user's directory entry, if it is available, since 1) the email may not be going to the same user that is trying to authenticate (it could be a hacker) and 2) some mail originate offline, when a browser is not involved.

The easiest way to localize email is to store a copy of each email file for each language in a separate directory, much like the language files. Then, when you specify the Mail Directory in the APS Configuration File, include, as part of the path, the text {preferredLanguage}. APS will replace that text with the user's stored language, thus changing the actual directory that APS will look for the file on. The correct file will then be loaded.

Keep in mind that the Administrator's language will sometimes be different from the user's, so keep in mind who the intended recipient of each email is to be.

See Using Email (see page 363) for more information about handling email.

Chapter 18: About Password Security

Passwords are the earliest and most common security measure. Password cracking is the earliest and most common security hazard. The careless use and maintenance of passwords represents the greatest threat to the security of a network. Thus, it is very important that your users choose a password that is difficult for another user to determine.

This section contains the following topics:

Cracking Passwords (see page 397)
Keeping Passwords Safe (see page 398)

Cracking Passwords

A password is stored on the system in encrypted form. It has been run through an encryption algorithm. There should be no algorithm that will take a password in encrypted form and give back the original password, so that *crackers* can't find out a password just by asking the system. Instead, they use a program like "Crack" to breach password security. The Crack program works by taking strings of characters and encrypting them, then comparing the encrypted text against the password in encrypted form. If the two encrypted versions are the same, then the string of characters is the password.

It would take too long to simply try every possible combination of letters you could have as your password -- over 100,000 years on a reasonably fast machine. So Crack tries the most likely combinations. It starts with everything it can find out about you on the system, like your login name, your full name, your address, your social security number, etc. Trying all of these takes a few seconds.

Then it moves on to a huge dictionary containing words from all languages, place names, people names, names of characters in books, jargon, slang, and acronyms. It tries all of them as your password. This takes several minutes. After Crack is done with that, it tries variations on those words, such as:

- any word, written backwards
- any word, with a punctuation character at the end
- any word, with a punctuation character at the beginning
- any word, with a punctuation character in the 3rd character place
- any word, replacing all *e's* with *3's*

- any word, capitalized
- any two words, put together with a number between them

It tries nearly every combination, and often successfully completes the task.

Keeping Passwords Safe

There are tricks to creating a good password that can't be easily determined yet can be remembered. System Administrators often set up strict password guidelines for their users. Here are some common DOs and DON'Ts:

DOs

- Use a password that contains non-alphabetic characters, e.g., digits or punctuation.
- Use a password that is easy to remember so that you do not have to write it down.
- Use a password that you can type quickly, without having to look at the keyboard. This makes it harder for someone to steal your password by looking over your shoulder.
- Change your password periodically.
- Change your password if you suspect that your account has been compromised.

DON'Ts

- Use your userid in any form (reversed, capitalized, doubled, and so on)
- Use your first, middle or last name in any form. Do not use your initials or any nicknames you may have.
- Use your spouse, significant other's, or child's name.
- Use a word contained in English, or foreign language dictionary.
- Use other information easily obtained about you. Examples include your telephone number, identification number, brand of your automobile, etc.
- Use a password of all numbers, or a password composed of the same character.
- Use a password shorter than seven characters.

- Write your password on desk blotters, calendars, or store it on-line.
- Reveal your password to anyone.

As a security precaution, many companies analyze their employees' passwords using the very same tools that attackers use. This is a good practice, but in most cases, the only way to ensure that password guidelines are followed is to have users change passwords through software that enforces the rules.

Chapter 19: SmPortal/SmTransact Installation and Configuration

This section contains the following topics:

Introduction (see page 401)
Installation (see page 402)
Operation (see page 404)
Configuration (see page 405)
Verification (see page 412)

Introduction

A typical installation of SiteMinder places the Web Servers in the DMZ and the SiteMinder Policy Server inside the firewalls. They communicate with each other using secure, encrypted communications through any intervening firewalls.

Several services supplied by CA Professional Services should be run behind the firewall, such as Advanced Password Services (APS) and Distributed Directory Administration (DDA). These services require access to the underlying user directory, usually LDAP, which should not be exposed within the DMZ for security reasons.

The architecture to accomplish this securely, easily, and with minimal configuration, is implemented by the SmPortal/SmTransact tunnel.

The SiteMinder Agent API toolkit includes a service-processing tunnel called SmTransact. SmTransact allows an arbitrary block of data (limited to 32k in size) to be passed through the firewall to the Policy Server. The Policy Server then passes the data to a library called SmTransact. SmTransact can then pass back up to 32k of response data.

The problem with this system is that it can only process a single service. It is up to the caller(s) to identify the processing that is to occur on the back-end and to write an SmTransact library that can dispatch the service request to the correct application code.

The SmPortal/SmTransact libraries solve this problem. This is a layer, consisting of the SmPortal library on the client and SmTransact on the Policy Server, which allows multiple services to run through the Agent API tunnel. It also provides the following additional capabilities:

- Buffer blocking, which allows more than 32k blocks to be passed (either direction) through the tunnel.
- Simplified configuration, so that each service need not supply its own communications configuration utility.

- Session tracking, with application startup and cleanup.
- Data streaming. Applications using this tunnel can stream data across the tunnel, rather than moving an entire block at a time.
- Service provider preloading. Service libraries are preloaded on the Policy Server, which improves run-time performance.
- Any number of agent/server combinations for fault tolerance.

These features are primarily of concern to developers.

About Firewalls

This kit uses the same libraries that SiteMinder uses to communicate between its agents and the Policy Servers. The same ports are used for this communication and the same encryption methods are used.

If a SiteMinder Web Agent can communicate through the firewall, since SmPortal should be using the same port and such, no additional configuration should be required on the firewall.

Installation

This section describes how to install SmPortal and SmTransact.

SmPortal - The Client Side

The SmPortal module consist of these files:

- The SmAgentAPI library (SmAgentAPI.dll or SmAgentAPI40.dll on Windows NT, libsmagentapi.so or libsmagentapi40.so on Solaris) is installed on the Policy Server and must be copied to the Web Server.
- The SmPortal library (SmPortal.dll on Windows NT or libsmportal.so on Solaris) is supplied with the SmPortal/SmTransact tunnel kit.
- SmPortal.cfg is supplied with the SmPortal/SmTransact tunnel kit.

Additional files may be installed to support the SmPortal module. These files include:

- APSPing (APSPing.exe on Windows NT) is an executable that can be used to test connectivity.
- SmPortalVfy (SmPortalVfy.exe on Windows NT) is a program that verifies the contents of the SmPortal.cfg file.

- APSVersion (APSVersion.exe on Windows NT) is an executable that can report the version numbers embedded in code provided by CA Professional Services. Not all CA code supports APSVersion.
- APSXLate and its supporting utilities. SmPortal is capable of localizing (translating) its error messages. The APSXLate module supplies this service to SmPortal. If APSXLate is not supplied, SmPortal will still function, but will always return its default error messages in English.
- APSEncrypt (APSEncrypt.exe on Windows NT) is an executable that can encrypt a shared secret. The encrypted value can be copied into the SmPortal.cfg file so that the shared secret need not be stored in clear text.

All files will be installed using the supplied installation program.

SmTransact - The Policy Server Side

On the SiteMinder Policy Server side, these files must be installed:

- SmTransact.dll (libsmtransact.so on Solaris),
- PortalTest.dll (libPortalTest.so on Solaris)
- APSTransponder.dll (libAPSTransponder.so on Solaris)

  Install this file on the SiteMinder BIN directory so that SmPortalTest and SmPortalVfy can be used from the client side to test the configuration.

Note: If APSPing, SmPortalTest and SmPortalVfy are not be used, the APSTransponder and PortalTest libraries are not needed. However, we highly recommend that they be installed anyway.

If the SiteMinder Policy Server is running, you may need to stop it before you can perform the installation.

There is a version of the SmTransact library that is supplied with SiteMinder itself. That version works well with most known SmPortal/SmTransact services, *but* definitely will not support SmPortalTest and SmPortalVfy as supplied. You should install the later version of SmTransact, if possible. There are no known problems with doing this.

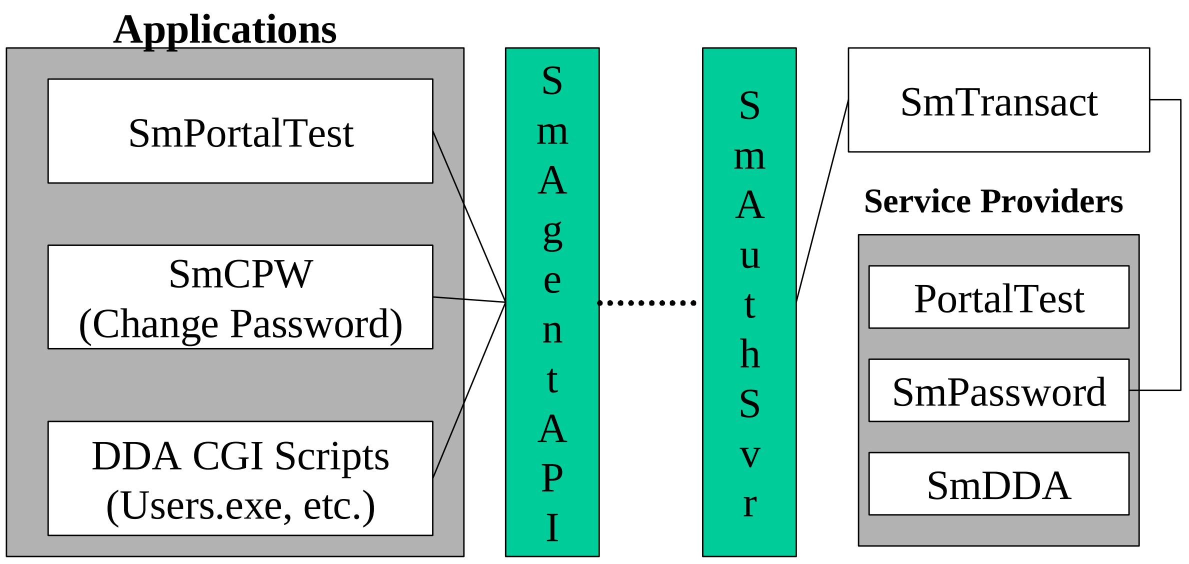
You can install SmTransact on your Policy Server using the supplied installation scripts.

Operation

Client applications, often running on a Web Server, make calls to the SmPortal library. SmPortal determines the location of the service provider and how to get there (using SmPortal.cfg). The call is made to SiteMinder communications code and data is passed to the SiteMinder Policy Server.

The SiteMinder Policy Server gets the request from the client and calls code within SmTransact. SmTransact determines the service being requested and loads the proper library (making initialization calls at that time).

SmTransact then calls the proper service code within the service library and returns the data to the caller. SmPortal and SmTransact perform some behind-the-scenes communication to handle large buffers (>32k).



Configuration

All configuration information for SmPortal is stored in the SmPortal.cfg file. SmPortal looks for this file using several techniques, in this order:

1. (Windows ONLY) SmPortal looks on the same directory that the SmPortal.dll file was loaded from. This file will be used, if it exists.
2. The current working directory will be checked for a file called SmPortal.cfg. If it exists (and is readable), it will be used.
3. (Unix ONLY) An environment variable called SMPORTAL will be checked. If it exists and contains the name of a readable file, the file pointed to will be used.

   If the file is still not found, an error will occur (it will not use default values).

   The SmPortal.cfg file supplied with the kit is not configured to work immediately after installation. It must be modified before communications will work correctly. It must also be modified as each new service is installed.

The following shows the file as it is installed (more or less):

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Servers
;;
;; Each server should have an ip
;; address, specified as:
;;   <server>.ip=aaa.bbb.ccc.ddd
;;
;; It *may* also have a port as:
;;   <server>.port=44442
;;
;; It *may* also have a timeout as:
;;   <server>.timeout=30000
;;
;; It *may* also have a minimum number
;; of connections as:
;;   <server>.ConnMin=1
;;
;; It *may* also have a maximum number
;; of connections as:
;;   <server>.ConnMax=5
;;
;; It *may* also have a connection
;; step value as:
;;   <server>.ConnStep=1
;;
;; The entry "Servers" should contain
;; a comma-delimited list of all
;; server names.
;;
;; Entry names are not case-sensitive.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
Servers=MyServer

MyServer.ip=127.0.0.1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Agents
;;
;; Each agent should have  list of
;; servers, specified as:
;;   <agent>.servers=<s1>,<s2>...
;;
;; It must also have a shared secret as:
;;   <agent>.secret=xyzzy
;; shared secrets may be encrypted using
;; the APSEncrypt utility (see documentation)
;;
;; It *may* also specify round robining
;; (instead of fail over) as:
;;   <server>.RoundRobin
;;
;; The entry "Agents" should contain
;; a comma-delimited list of all
;; Agent names.
;;
;; Entry names ARE case-sensitive.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
Agents=ghost

ghost.Servers=MyServer
ghost.Secret=[NDSEnc-A]6UUY0Uf4E6042CEuYo8H(E4UdyZ

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Services
;;
;; Each service should point to a
;; single agent, specified as:
;;   <service>.agent=
;;
;; It *may* also specif<agent>y tracing as:
;;   <service>.Trace
;;
;; It *may* also specify debugging as:
;;   <service>.Debug
;;
;; The entry "Services" should contain
;; a comma-delimited list of all
;; Service names.
;;
;; Entry names are not case-sensitive.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Services=PortalTest,SmPassword

PortalTest.Agent=ghost
SmAPS.Agent=ghost
```

The file is well documented, so that changes can be made with minimal (if any) referral to external documentation.

Any text following a semicolon (;) is considered a comment. Blank lines are ignored.

The file consists of three sections.

The Server Section

The Server Section defines the SiteMinder Policy Servers that are available from this computer.

The Servers= entry contains a comma-delimited list of all of the servers defined for this machine.

Server names are arbitrary. CA Professional Services recommends that the server names match the network name of the Policy Server machine. The name is only used to cross reference values within the configuration file. The name may contain embedded periods and is not case-sensitive.

Any number of servers may be defined in this section, but each server must have at least one entry following the Servers entry:

```
<servername>.ip=<IP-address>
```

This entry defines the IP address used to get to the specified Policy Server. The IP address specified here must be the one used to access the Policy Server from the server running the SmPortal code. Thus, if address translation is in effect at the firewall, the address may not match the actual address of the Policy Server.

The easiest way to determine this address at most sites is to open the *SiteMinder Web Agent Management Console* on the system and examine the address of the Primary SiteMinder Policy Server. If the Web Agent communicates properly with the SiteMinder Policy Server, then this is the correct address.

The *<servername>* portion of the setting name must match the name listed in the Servers setting.

Optional Server Settings

```
<server>.port=CA Portal
```

This setting may be used to change the port used to communicate with the Policy Server. The default is port 44442. It *must* match the port used by the *Authentication Service* on the SiteMinder Policy Server.

```
<server>.timeout=<timeout in milliseconds>
```

This allows the administrator to override the timeout for the specified server. The default is 30000 (30 seconds). This is a reasonable timeout and should not, in general, be changed.

```
<server>.ConnMin=<minimum connections>
```

This setting allows the specification of the minimum number of connections to the server that SmPortal should initialize. The default value of one should be used in almost all cases. This setting is intended for use by applications that require multiple connections to a Policy Server.

```
<server>.ConnMax=<maximum connections>
```

This setting allows the specification of the maximum number of connections to the server that SmPortal should initialize. The default value of one should be used in almost all cases. This setting is intended for use by applications that require multiple connections to a Policy Server.

```
<server>.ConnStep=<connection step value>
```

If the minimum and maximum connections are set, this value specifies the stepping value that SmPortal should use between the minimum and maximum values. For example, if the minimum value is one and the maximum value is 25, the stepping value might be five. This would cause SmPortal to increment the number of connections by five each time that it runs out of connections. The default value of one should be used in almost all cases. This setting is intended for use by applications that require multiple connections to a Policy Server.

The Agent Section

The Agent section tells SmPortal which agents exist and the servers that know about them.

The Agents= entry contains a comma-delimited list of all of the agents which may be used from this machine.

Agent names are *not* arbitrary. Each must match an Agent configured on a Siteminder Policy Server. The name *is* case-sensitive. Typically, the same agent names are used for both services and the resident Web Agent, but this is not required.

Any number of Agents may be listed immediately after the *Agents* line. Each Agent must be defined using the following two lines:

```
<agent>.Servers=<serverlist>
<agent>.Secret=<shared secret>
```

The first line defines the servers on which this agent is defined. The server list contains a comma-delimited list of servers that know about this Agent. Each server so referenced must be defined in the server section.

The order in which servers are defined is significant. By default, SmPortal will attempt to contact servers in the order listed here. Multiple servers are used for fault-tolerance; if a server cannot be contacted, the next server in the list is used. Any number of servers can be listed.

The Secret line defines the shared secret used between the Agent and the Policy Server. This must be the same secret configured on the Policy Server(s) for the Agent.

Secrets can be stored in either plain text or in encrypted format. If the secret is stored in plain text and this is a problem, define specific agents for services, thus hiding the shared secret used by the Web Agent. Since no policies need to be created in order to run the services through the tunnel (though they may be required to access the application through the Web Agent), security is maintained.

To store the secret in encrypted format, use the APSEncrypt utility to encrypt the shared secret, then copy the encrypted value (output from APSEncrypt) into the <*agent*>.Secret line in SmPortal.cfg.

Optional Agent Settings

By default, if multiple servers are specified for an agent, the servers will be used in failover mode (the first server used unless it is unavailable, in which case, SmPortal will use the second server, and so on).

To specify that the server list is to be used in round robin format, use the following setting:

```
<agent>.RoundRobin
```

SmPortal cannot do *true* round-robining because each application that uses SmPortal is a completely independent instance. Instead, SmPortal will *simulate* round-robining by randomly rotating the list of servers, then using the resulting list as though failover mode were specified. This technique provides the same results as true round-robining.

The Service Section

The Service section defines the services that can use SmPortal and what Agent configuration each should use.

The Services= entry contains a comma-delimited list of all services that may be used from this system.

Each application or API that uses a service has its service name encoded within it. See the documentation for the service itself to determine the name of the service. Note that the name may not correspond to the actual product name.

After the Services= entry, each service is identified by a single line that define the Agent to be used by that service.

```
<service>.Agent=<agent>
```

The value for the agent must correspond to an Agent name defined in the previous section.

The service name for SmPortalTest is PortalTest and should already be listed in SmPortal.cfg.

Some applications may support *qualified* service names. For example, SmPortalTest allows multiple definitions of the PortalTest service within the configuration file. Each entry must be *qualified* as in:

Qualifier1.PortalTest.Agent=ghost

AnotherInstance.PortalTest.Agent=ranger

Qualifiers allow the same service to run on multiple servers, yet still be addressable independently. In the PortalTest example, it allows the SmPortalTest application to select from several possible configurations (targets) for the PortalTest service.

If a service name is to be qualified, all optional settings should be qualified as well.

If a qualified service name is requested by an application, but no qualifiers match for that service, the unqualified settings for the same service will be used (if configured).

Not all applications support qualified services.

Optional Service Settings

`<service>.Trace`

This setting turns on trace logging for the service. The back-end service library will be notified that tracing is requested. The actual tracing performed is up to the service itself and will vary. All tracing is written to the SiteMinder console log.

Trace and debug logging may also be turned on within the code. If so, these settings will have no effect.

`<service>.Debug`

This setting turns on debug logging for the service. The back-end service library will be notified that debug tracing is requested. The actual logging performed is up to the service itself and will vary, though SmTransact performs some debug logging as well, if this setting is turned on.

Trace and debug logging may also be turned on within the code. If so, these settings will have no effect.

Verification

Two utilities may be supplied with the kit that can be used to verify a configuration.

SmPortalTest

The SmPortalTest (SmPortalTest.exe on Windows) can be used to verify that SmPortal can communicate properly with a SiteMinder Policy Server. SmPortalTest communicates through the tunnel to the PortalTest service, running on a Policy Server.

SmPortalTest takes each of its command line arguments and transmits them to PortalTest. PortalTest merely reverses the passed string and returns it.

A typical execution, on a properly configured site, might look like:

```
>SmPortalTest abc
Returned (4 bytes): "cba"
```

The string abc is passed and cba is returned. This indicates proper configuration.

A failure might be indicated thus:

```
>SmPortalTest abc
Returned Error Code = -18
Message: "Unable to establish server's IP address"
```

For suggested solutions to possible error codes, see Connectivity Problems.

The PortalTest library must be installed on each server that you wish to test a tunnel to. Note that the SmTransact library as supplied with SiteMinder does not include a version of PortalTest.

We recommend that *each agent* configuration be tested using SmPortalTest. This is done by configuring the PortalTest service for each agent (in SmPortal.cfg), in turn, then executing SmPortalTest to make sure that that agent's configuration is valid.

Additional Arguments

Most arguments passed to SmPortalTest will be sent to the PortalTest library as stated above. However, SmPortalTest recognizes some special command line arguments.

The available arguments will be displayed when -? appears on the command line.

Language Code

To specify a language other than English to use for error messages, use the -L option to specify the ISO code for the language, as in:

```
SmPortalTest -LFR
```

Country Code

To specify a country other than the US to use for error messages, use the -C option to specify the ISO code for the country, as in:

```
SmPortalTest -CCA
```

Isolating Transactions

Normally, SmPortalTest creates a single connection to the Policy Server and issues each service request to PortalTest on the same connection. This option tells SmPortalTest whether or not each service request (transaction) should be sent on the same connection or on a separate connection.

Isolation can be turned on or off and will affect all transactions appearing after the setting. For example,

```
SmPortalTest -I+ abc def -I- ghi jkl
```

will cause SmPortalTest to isolate the abc and def transactions. The ghi and jkl will be processed on the same connection.

Automated Testing

There are three arguments that are used for automated testing. The -X option specifies the number of iterations to be used (the default is infinite). The -S option specifies the number of seconds to wait between each transaction and the -T option specifies that automated testing is to be used and can specify the size of each transaction. When using Automated Testing, the isolation setting is honored.

For example,

SmPortalTest -T

will perform automated testing using the default transaction size of 512 bytes. The test will be continued until Control-C is pressed. SmPortalTest will not wait between transactions. Transactions will not be isolated (since this is the default).

SmPortalTest -I+ -X15 -S3 -T256

This command line will perform 15 (the -X option) isolated transactions (the -I+ option), 15 seconds apart (the -S option), using a 256 byte buffer.

There is a 1K limit to the automated testing buffer at this time.

Testing Specific Configurations

The -Q option can be used to *qualify* the service name used when referencing the SmPortal.cfg file. If -Q is used, it should be immediately followed by the desired qualifier.

```
SmPortalTest -Qghost abc -Qranger def
```

If -Q is used without a qualifier, no qualifier will be used.

```
SmPortalTest -Qghost abc -Qranger def -Q ghi
```

SmPortalVfy

SmPortalVfy is a new utility supplied with SmPortal version 5.0 that verifies the entire SmPortal.cfg file. It will only work properly if SmTransact version 5.0 (or later) is installed on the Policy Server.

SmPortalVfy is typically invoked with no command line arguments. It will test each setting in the default SmPortal.cfg file and report the results.

The verification process consists of two phases. First, the utility cross checks all of the configuration to ensure that the settings make sense within the file. If any problems are detected, they are reported and the verification process stops.

The second phase actually *tests* each setting within the file by attempting to communicate with the component. SmPortalVfy will attempt to communicate with each server, each agent's server (as the agent), and each service (as the agent for each of the agent's defined servers) defined within the file.

SmPortalVfy may not detect that a server is unavailable during its initial checking. This is normal and depends on a number of factors. If a server is unavailable, it will definitely be detected when the agents are checked against that server.

Any error during the communications phase will terminate processing.

SmPortalVfy supports one optional argument that specifies a path to SmPortal.cfg. This can be used to verify a new SmPortal.cfg file before putting it into production.

APSPing

APSPing is a new utility supplied with SmPortal version 5.0 that can test the connectivity to a single policy server, a service on that server, or an SmTunnel function housed by a service. It will only work properly if the (new) APSTransponder library is installed on the Policy Server.

Chapter 20: Internationalization & Localization (APSXLate)

This section contains the following topics:

Introduction (see page 417)
Installation & Configuration (see page 417)
Translation Files (see page 419)

Introduction

Several SiteMinder subsystems use the APSXLate library to perform *internationalization* and *localization*. The APSXLate library uses *translation files* to perform its function.

Advanced Password Services is capable of displaying user panels or error messages in multiple languages. It programmatically determine the user's *locality* (a combination of language and country) and then use the APSXLate library to translate text for that locality.

Installation & Configuration

The library consists of a single library called APSXLate (a DLL on Windows NT, SO on Solaris) and a single executable called APSXlateTest (an EXE on Windows NT).

A directory called Language is also required. This directory will contain the language translation files.

The APSXLate library locates the Language directory using the following rules.

1. (Windows ONLY) From directory where DLL was loaded, use subdirectory called Language
2. Use environment variable APS_LANG_PATH
3. (Unix ONLY) Use $SMHOME/Language
4. Use ./Language/
5. Use ./

Log Files

The library may need to write log information. If so, it will create a log filenamed using the current date in the format YYYYMMDD.log.

These log files will be placed in a subdirectory under the language directory (see above) called "logs", *if the subdirectory already exists*. If it does not already exist, the files will be placed in the language directory itself.

You can also specify the directory into which the logs are to be placed using an environment variable called APS_LANG_LOGS.

Note that wherever the files are to be placed, the current user must be able to create and write to this directory. If the directory does not exist or is not writable, no logs will be written.

As little information as possible is actually written to the log file, in order to boost performance and reduce the disk space requirements.

Translation file parsing errors, missing translation keys, and reload notifications (this occurs when a translation table is reloaded because the underlying files have changed) are the only items logged at the time of this writing. The DEBUG keyword (usable in translation files, see the next section) may, in the future, cause additional logging to occur.

Log files are only opened when required. Once opened, it will stay open until the process terminates or a log is written on a different date. Therefore, an older file might remain open after its date. The next log will cause the old file to be closed a new one opened.

The log files are primarily used to track missing translations. If an application requests the translation of a key that is not defined in the translation files, the key and default values are logged (the default is recorded so that no logging will occur until the translation files are reloaded).

By periodically watching for log files, a site can tell if unusual languages or countries are being requested or if a translation key is missing from a translation file.

APSXlateTest

This file is an executable that can be used for testing. Use a command line argument of **-?** to request its possible command line arguments.

Translation Files

When an application requests a language object from APSXLate, it specifies an *application*, a *language* and an optional *country*. Using this information, APSXLate uses up to *four* files to build a translation table to use for the locale. Given a language code of EN, a country code of US, and an application of APP, the four files would be (all paths relative to the translation files directory):

```
EN/(Common).lang
EN/APP.lang
EN-US/(Common).lang
EN-US/APP.lang
```

Each file's translation information is loaded, in turn, into the in-memory translation table. If a value already exists when a file is loaded, the existing entry is overwritten by the more specific translation file.

For example, the EN/APP.lang file might define a translation key for "Jurisdiction" that translates to "State or Province". In the EN-US/APP.lang file, the same translation key might translate to "State". In the EN-CA (Canadian English), the translation might be "Province".

Header information (also specified in the translation files) also override previously loaded settings, with the exception of directories (see the description of the DIRECTORY keyword for details).

If a translation value is not found for a given key, the application can specify a default value (usually in US English) that will be returned by the library and recorded in the log file and the in-memory translation table (the translation files are never updated automatically). If no default is specified, the key itself is returned as the value. By doing this, a language which has no translation files will still show up in English. Also, if a key is missing from a translation file, the value will still show in English.

A special exception is the Image Directory and Directory settings. If no value is read from any of the translation files, EN will always be returned, so that graphics can be properly displayed (and external data loaded), even for unsupported languages (albeit only English will be provided).

If a language is requested, but none of the four files can be found, APSXLate will log a message to the log file and will use the settings for EN.

Translation File Format

A translate file consists of a file header, which must exist and may not contain embedded blank lines (blank lines may exist *before* the first header line and at least one blank line must appear *after* the last header line).

Leading and trailing blanks and tabs are ignored.

Comments

Any file *beginning* with two forward slashes ("//"), a semicolon (";") or a pound sign ("#") are considered comments (trailing comments are not

supported). Comment lines may appear anywhere in the file and are ignored.

Continuation Lines

Sometimes translation values exceed a single physical input line. Start a line with an ampersand ("&") to continue from the previous line. The ampersand and any trailing spaces are removed and the continuation line is added to the previous line (with a space between).

Continuation lines are not supported within the header section of the file.

Header Keywords

The header consists of a collection of keywords, followed by values. The keyword may be separated from its value by at least one space. Additional spaces or tabs are not significant. Keywords are not case-sensitive.

Comments may appear within the header, but once a single header keyword has appeared, no blank lines may appear until the end of the file header. The file header ends when a blank line appears. Note that at least one header keyword is *required*.

Many keywords have multiple possible spellings or formats.

```
Long Name <language long name>
LongName <language long name>
Name <language long name>
Description <language long name>
```

This allows a name to be associated with the language for documentation purposes.

```
Images <image directory>
Image Directory <image directory>
Image Path <image directory>
```

The value for *<image directory>* is returned to the application when requested. Usually, this is a subdirectory of a general image directory that contains language-specific images. If requested by an application, but not specified in the translation file(s), APSXLate will always return EN, so that English graphics can always be available.

```
Directory <type> <directory>
```

The specified *<directory>* will be returned when the *<type>* is requested. Directory types used by an application are defined by the application. For DDA, for example, this allows the site to specify directories where email files, list files, and include files will be stored for each language.

```
Charset <charset value>
Character Set <charset value>
```

The character set code associated with the language. "ISO-8859-1" is used if this value is not specified.

```
Prompt Suffix <suffix>
```

Allows the language to specify the suffix to use when an application requests a prompt translation. By default, a colon (":") will be used.

```
Direction LTR
Direction RTL
```

Specifies the direction used by the language. Currently, RTL is properly parsed, but only LTR is actually used (RTL is ignored). This keyword is for future expansion.

```
Debug
```

Turns on debugging for all objects using this translation file. At this time, no debugging code exists, but it may be added in the future.

Button Definitions

Buttons are defined using two keywords, each appearing on a separate line.= Each keyword must be followed by at least one space. Additional spaces or tabs are ignored. The keywords are not case sensitive.

Button settings are explicitly requested by applications.

```
Button <key>
```

Specifies the key that identifies the button. Applications request values by key, so that all translations for a specific button must use the same key, which is defined by the application. Values for <*key*> are case-sensitive.

```
Label <key-label>
```

Specifies the label to use for the button key defined just previously (using the Button keyword). If not specified for a button, the <*key*> will be used (useful only for English).

Translations

Translations are defined using two keywords, each appearing on a separate line. Each keyword must be followed by at least one space. Additional spaces or tabs are ignored. The keywords are not case sensitive.

Key translations are explicitly requested by applications, either as a "normal" translation or as a "prompt" (in which case the prompt suffix is affixed to the translated value).

```
Key <key>
```

Specifies the key that identifies the translation. Applications request values by key, so that all specific translations must use the same key, which is defined within the application. Values for *<key>* **are** case-sensitive.

Many keys are actually their English value. This easily ensures that if a key is not defined, at least an English value will be displayed (since the library returns the key if there is no translation defined or a default value supplied by the application).

```
Value <translation>
```

If the translation is long, the value for <*key*> may be encoded (usually with a t_ prefix to prevent naming conflicts). This improves performance when looking up values in the translation table.

```
Val <translation>
```

Specifies the translated value for the key defined just previously (using the *Key* keyword). Translation values may be continued onto the next line as described previously.

Chapter 21: Best Practices for Storing Legacy/Back-End Credentials

This section contains the following topics:

Executive Summary (see page 423)
Details (see page 425)
Summary (see page 427)

Executive Summary

SiteMinder is often used to protect legacy or database applications that require their own user credentials. These applications are usually front-ended by a web application that must log in (authenticate) to the legacy application, behind the scenes, using these user credentials.

As SiteMinder and other Single Sign-on (SSO) solutions have been implemented within large organizations for large commercial application, tight integration has been undertaken by CA and those applications' vendors. When such integration is available, SSO can be a boon to productivity and reliability.

When such integration is not yet available or is not possible, application developers and integrators sometimes request that SiteMinder supply a user's LDAP credentials (user id and password) to their application so that they can perform this login. This is unnecessary and, from the security and data maintenance standpoints, this is a *very* bad idea for a number of reasons:

- If one application becomes compromised, all applications at the site become compromised.
- Due to architectural limitations of SiteMinder, all protected pages in the original authentication realm may receive the password (*all* pages receive the user id anyway). Note the word "may", SiteMinder only guarantees that the first page receives the value; others might as well, depending on a large number of factors. If a user logs into one realm, it is not available to other realms unless stored elsewhere, such as within a custom cookie or session storage.
- Password policies must conform to a "least common denominator" format. That is, password construction must conform to the limitations of the simplest application. For example, if an application can only handle 6 character, case-insensitive passwords, then the user's site password can only be subject to that limitation.

- Password synchronization is very non-robust and difficult to implement. When the user's site password changes, the change must be replicated to all back-end systems. This operation is very error prone, due to outages. As additional applications that require this service are installed, performance (during password change) suffers as well.
- SiteMinder authentication will require a password, since this password will be used elsewhere on the site. In other words, all of the other authentication options, such as tokens and certificates, will be unavailable to the site.

A more secure, robust and reliable solution meeting application integration requirements has been developed and deployed.

Essentially, the legacy application requires *a set of credentials*, not *the* set of credentials. The site should store the alternative credentials in the user's directory entry, encrypted, and supply them to the single application only. This has several significant advantages:

- Each application has its own unique password (and, possibly, user id). If one application becomes compromised, no other application is at risk.
- Each application can receive its own credentials when it needs them, regardless of how authentication took place: which realm or what authentication scheme was used.
- Each application's credentials can be subject to its own format restrictions.
- No password synchronization is required.
- Since SiteMinder authentication does not use the same credentials as the applications, the site may use any of the many authentication options that SiteMinder supports, such as certificates, without impacting any applications whatsoever. In addition, the site may "mix and match", meaning that different users may use different authentication methods. These methods may be implemented later, as the site matures and requirements change, without impact.
- Application passwords can be changed by a background process as needed. Users need not know that their back-end credentials have even changed.

This solution does not work in all cases. We have encountered at least one site where it does not work. That site had the additional requirement that the legacy application be accessible internally using a thick client. When the employees authenticated using the thick client, the user was required to entire the credentials. Our customer wanted those credentials to be the same as those used for the web site. That was a special case, however, and should not be considered typical.

Details

There are several possible implementations. This document describes the idea and only one possible implementation. The implementation described here was selected because it scales well and because it can be used on an incomplete site. That is, as new applications are SiteMinder enabled at a site, little additional programming and *no* directory schema changes are required.

The basic solution requires that the credentials for a specific application be stored in the User's Directory Entry. For the purposes of this discussion, let us assume that the User Directory is an LDAP directory. It does not have to be; it could be any SiteMinder-supported User Directory, with the exception of Windows NT Domains, since there is no place in an NT Domain entry to store additional credentials and the schema cannot be extended.

The credentials for a given application should be encrypted. It is up to the application to decide how and to do so. Encryption keys should be different for each application so that if the keys are compromised, they do not compromise another application at the site.

All credentials for all applications should be stored in a single, multi-valued attribute in the user's entry, prefixed by the application name. For example, a record might contain the following entries:

```
HRAPP=UhdNoiHhs4hjkj
CRMAPP=(JH)h9kdYhUKK-
```

Using a tool such as CA Professional Service's SmWalker (and its KeyValue function), you can retrieve the value associated with a given key as the result of an Active Expression. The Human Resources application can then be passed the (encrypted) credentials as a header variable. Note that the credentials remain encrypted; the application itself is responsible for decrypting them, if required.

Alternatively, each application's credentials can be stored in individual user attributes, each specific to the application. While this simplifies updating application credentials (see next section) and retrieving the value for the application (no active expression is needed), this does not scale. As new applications are added to the system, additional attributes are needed, resulting in schema changes and possible data conversions, depending on the underlying user directory.

Updating Application Credentials

The application credentials can and should be changed on a regular basis. However, users need not know that this is being done, nor do they need to know what the new credentials are. For that matter, no human needs to know them and, depending on the back-end system, they may not even be human-readable.

Each application can change their credentials independently, on their own cycle (frequency).

To change the credentials requires a batch job (e.g. cron on Solaris, At on Windows) that generates the new credentials, updates the back-end system and updates the directory entry for the user. Without cross-database transactions this can be error-prone; however, the process can be rerun and restarted without user notification or affect. It is important that these scheduled tasks *not* run concurrently, as they could end up interfering with each other. For example, if two separate jobs attempted to update the same user's record simultaneously, some data loss may occur.

The generator will need to encrypt the value in a way compatible with the data consumer.

If credentials are stored as multiple values in a user's entry, then care must be taken to preserve the credentials of other applications. Fortunately, if code is written well, these other values cannot be lost, since, while LDAP does not support transactions, it does guarantee "all or nothing" updates, meaning that all values for the attribute will be replaced or none of them will.

Updating User Credentials

Users can change their directory passwords (used by SiteMinder) at any time without any impact. As additional applications are added to the site, even if they require legacy credentials, they do not add overhead to the change process (since there is no synchronization). Applications do not care when the user's base password changes.

Additional Advantages

Password Format & Lifetime

Because the user's base password is separate from application credentials, it can be forced to conform to any Password Policies that the site sees fit to implement. Password Services or Advanced Password Services can require very complex formats and/or lifetimes with no impact on legacy applications.

Alternative Authentication Schemes

The credentials used by SiteMinder and the applications need not even be the same type. Whereas the application(s) need passwords, a site could use any of the alternate authentication mechanisms that SiteMinder supports, including token cards and certificates.

Note that while an initial implementation at a site may not require this level of security, it may become an issue later, as more applications, perhaps requiring a greater level of security, are integrated into the site.

Summary

Application designers often ask for SiteMinder to supply a user's credentials (user id and password) to their application so that the application can perform a login against a back-end system.

This document has presented an alternative design to store a separate set of credentials for each application, encrypted, and provide those credentials only to that application as needed.

Appendix A: How Do I?

This section discusses how to use APS features to implement common site requirements. All of the information presented in this section exists elsewhere in this document in reference format. This section presents the information in a guideline format.

This chapter will often refer the reader to other parts of this document for further details.

This section contains the following topics:

How Do I: Set up passwords to expire? (see page 430)
How Do I: Customize the change password form? (see page 432)
How Do I: Configure "3 Strikes and You're Out"? (see page 432)
How Do I: Force users to change their password at next login? (see page 434)
How Do I: Determine why a user's authentication is rejected? (see page 434)
How Do I: Enable an account that APS has disabled? (see page 435)
How Do I: Retrieve the date & time that a user last logged in? (see page 435)
How Do I: Limit password changes to once per day? (see page 435)
How Do I: Eliminate the current password from the change password form? (see page 436)
How Do I: Use a backup LDAP server (replicant)? (see page 436)
How Do I: Use a consumer LDAP server (satellite)? (see page 437)
How Do I: Detect & Prevent FPS Misuse? (see page 437)
How Do I: Lock FPS (mis) users out of our site? (see page 437)
How Do I: Re-enable FPS for an account? (see page 437)
How Do I: Automatically login the user using FPS? (see page 438)
How Do I: Allow a user to select their own password during FPS? (see page 438)
How Do I: Retrieve statistics about FPS usage? (see page 438)
How Do I: Securely support FPS challenge questions? (see page 439)
How Do I: Get the FPS challenge questions into the User Directory? (see page 441)
How Do I: Integrate the APSAdmin interface into my own Identity Management application? (see page 441)
How Do I: Make the APS API return error codes instead of error text? (see page 442)

How Do I: Set up passwords to expire?

Assuming SiteMinder and APS are configured correctly (See Event Handling), setting up password expiration is easy. The event configuration is required to support the redirection of the user when the password actually expires.

To control password expiration, a site needs to specify the following settings in the APS Configuration File:

Password Expiration	This setting controls how often users are expected to change their password.
Expiration Warning	This setting controls how long *before* the password expires that APS will start warning the user that the password will expire, thus giving the user a chance to change the password before it expires.
Expiration Grace	Once the password expiration period has passed, APS will attempt to force the user to change their password. This setting controls how long APS will make these attempts. Once the grace period has expired, the user will be disabled.
Grace Logins	This limits the number of logins that a user can attempt after the password expires before the user is disabled.
Expired Redirect	This setting tells APS where to send the user when the user's password actually expires (at the end of the grace period).
Warning Redirect	APS uses this setting to determine where to send the user when warning that the password will expire. This is usually a password change form.
Expire Change Redirect	APS uses this setting to determine where to send the user when the user must change their password. This is usually a password change form.
Expired Password Mail	This setting tells APS the name of a file to send to the user when the password actually expires. It is rarely used, since it is generally more use friendly to present a page to the user when this event occurs.
Password Warning Mail	This setting tells APS the name of a file to send to the user when APS must warn the user that the password will expire. It is not often used, but it is actually *more* user friendly to use this than a redirection (assuming that email addresses are accurate), since it does not interrupt the user's workflow.

Force Change Mail	This setting tells APS the name of a file to send to the user when the user must change their password. It is never used, since the user must change their password before accessing the site.

Password expiration works as follows:

The password expiration date is calculated (by APS) from the following:

1. If the Password Last Changed date (smapsLastPasswordChange) is available, use it, otherwise use the Base Date (smapsBaseDate) if available, otherwise use today's date.
2. Add the value of the **Expiration Delay** setting. This is the *Soft Password Expiration Date*.
3. Add the value of the **Expiration Grace** setting. This is the *Hard Password Expiration Date*.

Starting **Expiration Warning** days before the *Soft Password Expiration Date*, but before the *Hard Password Expiration Date*, every time that the user attempts authentication, the user will be sent to the **Warning Redirect** page and the **Password Warning Mail** will be sent.

If the current date (*and time*) is *after* the *Hard Password Expiration* date, the user is disabled. When this happens, the user is redirected to the **Expiration Redirect** page, if specified, and the **Expired Password Mail** is sent. The user is not allowed to access the site (the authentication is *rejected*).

If the current date (and *time*) is *after* the *Soft Password Expiration* date, but *before* the *Hard Password Expiration* date (and no **Grace Logins** are allowed), the user's authentication is approved, but the user will be redirected to the **Expire Change Redirect** page (if this setting is missing, the **Force Change Redirect** setting will be used instead) and the **Force Change Mail** will be sent to the user.

If the current date (and *time*) is after the *Soft Password Expiration Date* and **Grace Logins** are allowed:

1. If the current user has used all of the available grace logins, the user is disabled. When this happens, the user is redirected to the **Expiration Redirect** page, if specified, and the **Expired Password Mail** is sent. The user is not allowed to access the site (the authentication is *rejected*).
2. If this is the *last* available grace login, the user will be authenticated, but will be put into a force change password situation (by AZRedirect).
3. If this is not the last available grace login, the AZRedirect will not enforce a *forced* password change.

How Do I: Customize the change password form?

This topic is discussed in detail in Chapter 6, Change Password Interface (SmCPW) (see page 271).

How Do I: Configure "3 Strikes and You're Out"?

Assuming that SiteMinder and APS are configured correctly (See Event Handling.), setting up this feature is easy. The event configuration is required to support the redirection of the user when the event actually occurs. You may also wish to review Special Case: Three Strikes, You're Out.

To control this feature, a site needs to specify the following settings in the APS Configuration File:

Max Failures	This setting controls how many sequential failures it takes to trigger this event.
Failure Count Timeout	APS uses this setting to determine how long it should "remember" failures and, if *Auto Reset Failure Count* is specified, how long user account should be disabled before being automatically re-enabled.
Auto Reset Failure Count	If this setting is specified, then users will be automatically reset after *Failure Count Timeout* minutes of inactivity.
Failure Redirect	This setting tells APS where to send the user when the event occurs. It should not be used (the reasons are discussed in the text).

Max Failures Mail

This setting tells APS the name of a file to send to the user when the event occurs. This is the preferred method for notifying the user because it is more secure.

There is another setting, not implemented within APS, that *seems* to perform this function, but actually performs a completely different function. The two settings can be used together to make the system much user friendlier.

Form Login has a feature called *Retries*. This setting *does not* control when the user is disabled. It instead controls the number of consecutive attempts that can be made before the browser must be restarted (note that *Basic Login* has the same feature, but the number of attempts is hard-coded into browsers as the value "3").

The Retry setting *only* controls the browser, not user disabling. If the user attempts three different user id's, for example, the Retry setting will still take effect, though an APS setting will not. On the other hand, three attempts on three different browser instances will not invoke Retry, but may disable the user via APS.

The Retry setting is intended as a "speed bump" for programmatic password crackers. That is, the cracking program must handle the shut down/restart of the browser ever *Retry* attempts. The APS setting is a complete "stopper", absolutely closing off the user account to the cracking program.

The two settings should **not** have the same value. It is far user friendlier to set the Retry setting to a value of, say, 3 and the APS Failure Count setting to a larger value, say, 5 or 9.

In this way, a user that has legitimately forgotten his password will hit the Retry setting first. Presumably, the result of hitting this count presents the user with some message asking the user to call Customer Support. A legitimate user will do so. Since the user was never disabled, the reset is not needed.

However, a programmatic password cracker will ignore the "speed bump" message and will continue to attempt to crack the site. Eventually, the APS setting will be invoked and the user will actually be disabled (and an administrator notified?). The APS setting might be 9, but, with proper password policies, it is impossible to crack a password in 9 (or even 99) tries!

How Do I: Force users to change their password at next login?

To perform this function, SiteMinder and APS must be configured correctly (See Event Handling.). The event configuration is required to support the redirection of the user when the event actually occurs.

To do this, the user's entry must be "flagged" to require the change. Under Windows NT, there is a checkbox within the user profile that can be changed using the *User Manager for Domains* utility supplied with Windows NT.

For LDAP directories, use the APSForcePWChg - Set Force Password Change Flag provided with APS, the SmBlob utility, or modify the **smapsImmediateChange** attribute in the user record to set this flag.

You can use the Auto Force Chang setting to have APS force the user to change their password when the password is changed by an external utility.

To control this feature, a site needs to specify the following setting in the APS Configuration File:

Force Change Redirect:

APS uses this setting to determine where to send the user when the user must change their password. This is usually a password change form.

How Do I: Determine why a user's authentication is rejected?

Sometimes, user authentications seem to be rejected arbitrarily. Note that no action performed by a computer is completely arbitrary: there is always a reason. Sometimes the reasons are just not very obvious.

The first thing to do is to check the SiteMinder Authentication Server's console log. SiteMinder will log whether the initial authentication succeeded or failed. APS will always honor this determination first.

If SiteMinder successfully authenticated the user, but authentication still failed, APS probably rejected the authentication. Check the console log for APS' logging of the reason for rejection.

If further information is required, try turning on APS tracing (using the Trace setting). APS will then display considerably more information about why the user was rejected.

How Do I: Enable an account that APS has disabled?

This is discussed in detail in User Directories: Schema, Storage and Capabilities (see page 189).

How Do I: Retrieve the date & time that a user last logged in?

If this information is to be displayed on administrator's screen, get the value of **smapsLastLogin** for the desired user. If the information is to be displayed on the user's own screen, use the value of the **smapsPreviousLogin** attribute (since the Last Login attribute contains the date/time of the current login).

Both of these dates are in Greenwich (ZULU) Time.

How Do I: Limit password changes to once per day?

This feature seems to be a desirable feature of Advanced Password Services. However, upon further evaluation, it is not, for the following reasons:

This limit cannot be honored when the user is *forced* to change their password. For example, the user voluntarily changes their password in the morning, then forgets what it was changed to. The administrator then resets the password (through whatever means). The user would then typically be forced to change the password yet again when he next logs in. The limit cannot be honored in this case.

In order to be user friendly, the user should be prevented from *requesting* the password change, not just the actual change process. That is, the change password form should not be displayed; a warning message should be displayed instead.

- This limit is not really necessary, since its *intent* is to prevent a user from changing the password (when required), then immediately changing it back to the original value. This can be prevented using the Reuse Delay setting. By setting **Reuse Delay** to 1 (day), the user can change the password as frequently as desired, but cannot reuse a particular value for 24 hours.
- This is not to say that this limit is a bad idea. It can be relatively easy to implement at a site by using some simple server-side code within a custom change password form. This code must appear within the optional change password form only (for the reasons described above).

The SiteMinder Policy for the custom form should pass the value of the **smapsLastPasswordChange** attribute to the form. The server-side code should check to ensure that the proper amount of time has elapsed for the user to (voluntarily) change their password. If not, the server-side code should display a user-friendly message instead of the change password form.

It is very important that the document's content expire immediately. This will prevent the user from pressing the "Back" button on the browser to access the form again. Browsers will recognize that the content has expired and will request new content from the server. This allows the server side code to execute again to check for the validity of the password change.

Note that the result of the Response should not be cached (this is defined within the Response definition).

How Do I: Eliminate the current password from the change password form?

Traditional operating systems will force a password change immediately after login. When this occurs, it makes no sense to ask the user to enter their current password, since the password was just entered during login.

In the Web environment, this does not work. Browsers provide a "Back" button that would allow the user to return to a previous panel to enter a new password again. A user could thus be forced to change their password, walk away from their keyboard, and another user press the "Back" button to change the password again (since the current password is not needed).

Because of this, APS *requires* that the current password be entered as part of *all* password changes. Even if a user presses the "Back" button, the password cannot be changed without re-entering the current password (browsers do not preserve the contents of password fields).

A site might wish to bypass this by supplying a hidden field with the current password value. This field would be used on the password post, but the user would not have to enter it. While the "Back" button problem is solved (since the older password is still there and won't match the just changed value), it creates several security holes:

- The original password is visible by viewing the source for the page.
- The password must be retrieved during the authentication process and passed as a Response. Without resorting to complex code, this value is visible to **all pages** at the site (since it must be an OnAuthAccept response).
- A rogue web programmer might be able to change any password on the site without knowing the existing password.

Bottom line: This functionality should never be implemented.

How Do I: Use a backup LDAP server (replicant)?

See Fault Tolerance & Performance..

How Do I: Use a consumer LDAP server (satellite)?

See Fault Tolerance & Performance..

APS writes to the LDAP server on all authentications. The write includes not only login information, but may include setting the user as disabled.

APS implements the Writeback settings (See LDAP Write Back Information.) to handle this type of configuration quite efficiently. These settings will cause APS to do all reads from the satellite LDAP server and perform all writes to the primary server.

How Do I: Detect & Prevent FPS Misuse?

"FPS Misuse" is defined as:

- Users who utilize FPS frequently for whatever reason.

  To keep this from happening, use the **Too Recently Used** settings to prevent users from freely using FPS.

  To detect this kind of misuse without restricting user access, use the **Logging Attribute**. You can create searches and reports to determine how often each account has used FPS and how frequently.

- Hackers who try to use FPS to break into your site.

  To keep this from happening, use the **Too Recently Attempted** settings to prevent hacker attacks. Also, many of the mail events can be used to send mail to internal security administrators when certain types of attacks are detected.

  Use the **Audit Log** setting and monitor this log.

How Do I: Lock FPS (mis) users out of our site?

To lock users out of FPS, use the **Lockout Count** and related settings. If the user is to be locked completely out of your site and you have APS installed, use the **Lockout Group DN** setting to put the user in an APS disabled group.

How Do I: Re-enable FPS for an account?

If FPS is disabled due to Lockout, an administrator should delete the value of the smfpsLockoutCounter **for that user accoun**t.

If the user cannot use FPS because of the **Too Recently Used** or **Too Recently Attempted** settings, the best bet is to let the user utilize the Help Desk to reset their password until the required time has elapsed.

If it is absolutely necessary to reset this information, delete the values stored in **smfpsLog** attribute in the user record. The side-effect is that all audit information about the use of FPS by this user is lost.

If the user cannot access FPS because the account is disabled, then the account must be re-enabled. This is discussed in detail in User Directories: Schema, Storage and Capabilities (see page 189).

How Do I: Automatically login the user using FPS?

You should use the OneShotPassword functionality, so that you need not expose the "real" password. The setup required for using OneShotPassword is described in Confirm Pages (see page 315).

How Do I: Allow a user to select their own password during FPS?

Once again, this comes down to the BACK button on the browser. In order to change the password, the user *must* know his old password. Since, by definition, FPS is used when the user does *not* know their own password, standard APS change password services cannot be used. FPS provides a random password reset, then requires that the user utilize the standard APS service upon login. This prevents the BACK button security hole described in the previous section.

You can use the **[FPS-Change]** section to allow a user to do this securely, if it absolutely must be done. Keep in mind that this is not very secure.

How Do I: Retrieve statistics about FPS usage?

FPS keeps a lot of information about its use. The two settings to maximize the log data are **Audit Log** and the **smfpsLog** attribute.

FPS records all activity in the file specified **Audit Log** in comma-delimited format. The fields logged are date, time, instance, state, user DN (if known) and message. The instance field is merely a one-up number that can relate two messages running for the same request. The instance number is not maintained between requests from the same user.

Since the file is recorded in comma-delimited format, it is easy to import this file into a number of applications, databases and report writers.

There is no provision within FPS for rotating or truncating this file, so your site should prepare for it.

smfpsLog provides a way of determining the FPS activity of a particular user.

How Do I: Securely support FPS challenge questions?

The best way that we have determined to do this is something we call "the thousand questions". This mechanism requires that users select ten or more questions from a large list of questions (usually presented to the user as a series of drop-down boxes). Using the *Special Instructions (see Special Instructions)*, FPS can select a subset of the entered questions to ask. Really secure sites will *consume* questions as they are asked, preventing them from being asked again, and will require that the user select additional questions in order to use FPS in the future.

"The Thousand Questions" refers to the long list of questions from which the user can choose. There need not actually be a thousand questions. Some questions can, indeed, be politically incorrect for some users; users need not select that question, so they need not answer it. Example questions appear below:

1. From which high school did you graduate?
2. In what city were you born?
3. Up until January 2001, how many cars have you owned?
4. What brand was your first car?
5. What color was your first car?
6. What is the first high school you attended?
7. What is the first phone number you remember, without the area code?
8. What is the first type of plane you flew on?
9. What is the last name of your favorite athlete?
10. What is your best friend's dog's name?
11. What is your favorite airport?
12. What is your favorite beverage?
13. What is your favorite board game?
14. What is your favorite bread?
15. What is your favorite breakfast cereal?
16. What is your favorite cartoon character?
17. What is your favorite cartoon?

18. What is your favorite cheese?
19. What is your favorite color?
20. What is your favorite flower?
21. What is your favorite fruit?
22. What is your favorite gemstone?
23. What is your favorite holiday?
24. What is your favorite jellybean color?
25. What is your favorite movie?
26. What is your favorite Olympic sport?
27. What is your favorite property in the board game Monopoly?
28. What is your favorite restaurant?
29. What is your favorite soft drink?
30. What is your favorite sport?
31. What is your favorite stage play?
32. What is your favorite TV show?
33. What is your favorite type of fish?
34. What is your favorite type of pasta?
35. What is your favorite type of steak?
36. What is your favorite vacation spot?
37. What is your first boss' name?
38. What model year was your first car?
39. What was the name of your first elementary school?
40. What was the name of your first grade teacher?
41. Where did you go for your first vacation without your parents?
42. Where did you meet your significant other?
43. Who did you go to the prom with?
44. Who is your favorite author?
45. Who is your favorite college professor?
46. Who is your favorite composer?
47. Who is your favorite musical artist?

How Do I: Get the FPS challenge questions into the User Directory?

This is not a function of APS. While APSAdmin could be used to enter simple questions and answers as separate user attributes, it cannot handle the formatting described in Special Instructions.

A site should use its user maintenance application to perform this function. For LDAP User Directories, Delegated Management Services (DMS) is ideally suited for this purpose.

The CA Professional Services Deployment Group has experience creating and modifying such applications, for both DMS and non-DMS environments. Contact your CA Sales Representatives for details.

How Do I: Integrate the APSAdmin interface into my own Identity Management application?

APSAdmin (See Help Desk Interface (APSAdmin) (see page 277).) is a help desk tool that can be used to allow administrators to enable/disable user accounts and, possibly, to reset their password. This "How Do I?" is assuming that a site wants to use the actual forms generated by APSAdmin.

An APSAdmin session consists of two parts:

1. Finding the user;
2. Showing the user on the form.

Most of the configuration of APSAdmin is involved with the second part, which is described in Help Desk Interface (APSAdmin).

The user navigation portion of APSAdmin is very rudimentary and is really designed to allow developers to access user data for testing purposes. A full function user navigation tool would require considerable configuration and design, since only the site knows the logiv of how their users are organized in the directory.

However, the *second* part (the user form) can be invoked directly from any application. Very often during Proof of Concept or initial rollouts, sites can avoid writing a considerable amount of code by invoking the APSAdmin user form directly from within their Identify Management application (DMS2, CA Identity Manager, etc.).

The fastest way to do this is to modify your *existing* user management application *on the user profile edit form*:

1. Determine if the current user is an administrator and has the rights to perform the APSAdmin operations for this user.
2. Put an HTML link (using the "Action:" tag) on the page labeled "Administrator Functions" or some such. This tag should invoke the APSAdmin interface with the proper query string to invoke the page for the current user (See Suppressing the User Selection Panel Altogether). Some sites prefer to bring this page up in a separate browser session, but that is a site choice.

After this is done, valuable time can be spent customizing the forms, rather than accessing data and performing edits.

How Do I: Make the APS API return error codes instead of error text?

Many sites use the APS Application Programming Interface (See Application Programming Interface (see page 319)). The API functions take, as arguments, the desired ISO language and country codes and returns errors as *text*, properly localized, rather than error codes. This can be a bit tricky if the application wishes to perform specific handling for specific errors.

The use of text, rather than codes, goes beyond simple translation issues. Many error messages, especially the password content messages, have values (parameters) inserted into them at run time. For example, one supplied (English) error message might come out "Your password must contain at least 2 numeric digit(s)". The "2" is inserted at run-time, based on what is actually configured in the APS Configuration File.

However, it is still not difficult for a site to set up the API to return error codes.

All server-side messages are stored in files called APS.lang under the Language subdirectory. There may be more than one APS.lang file, one for each language and/or language/country code.

The format of a language file is described in Internationalization (see page 391). Essentially, each message is identified by a "key" and has an associated "value". The application (APS) requests the value of a "key" for a given language/country code. The translation library (APSXLate) loads the proper application language file (APS.lang) for the requested language/country code, then looks up the key.

For a given application, the keys are all constant, for all versions of the file, but the *values* associated with each key reflect the language for which that file is intended.

To make the APS API return error codes instead of text, copy the APS.lang file from the EN directory to a new directory called API (the "language code" will now be "API" rather than "EN"). Edit the file using the text editor of your choice and replace the value associated with each key with the numeric code that you wish APS to return for that particular message.

When your application wishes to make an APS API call, supply "API" as the language. APS will load the APS.lang file associated with the "API" language code. It will then return the message (value) associated with each key. In the "API" language file, these are numeric codes rather than text string.

The API will always return an error *string*, not a number. By following the instructions above, you can have APS return an error code, but it will still be returned in a *string*. Your application may have to convert it to a numeric value.

There *is* a way to take advantage of this for the purposes of testing/debugging. APS will log the returned error message to the SiteMinder Authentication Console Log. Logging numeric values will be rather cryptic.

When converting a string to a number, most languages will convert all characters in the string *up to the first non-numeric character*. One recommendation is to *not* replace the entire value in the language file with the desired error code: instead, insert the desired numeric code *at the beginning of the value, followed by a colon*. For example, replace:

```
key ERR_DICTIONARY
val New password includes an embedded word that is disallowed.
```

With:

```
key ERR_DICTIONARY
val 20:New password includes an embedded word that is disallowed.
```

Now, when this error is returned to your application and you convert the returned string, it will see "20". Yet the entire message is logged to the console log. Your application could then even peel off everything before the colon (including the colon) and still display the message to the user.

You could even treat "API" as a *country code* instead of a language code, allowing the *text* portion to change by actual language.

Your application may wish to preserve the inserted parameters (as decribed in the example above). This is also easy to do.

The replacement parameters are *position independent*, meaning that the order that they appear within the string is not important. A given parameter can even be inserted more than once.

Using the technique described above, you can still pass the important insertion parameters (note that insertion parameters differ by error message):

```
key ERR_MINIMUMLEN
val New password must be at least %0 character(s) long.
```

Could become:

```
key ERR_MINIMUMLEN
val 35 (%0)
```

When converting the error string to a number, your application will still see code 35. But the important insertion parameter (the minimum length) will get passed as well.

The two techniques can be combined, since a given parameter can be inserted more than once:

```
key ERR_MINIMUMLEN
val 35 (%0):New password must be at least %0 character(s) long.
```

Appendix B: Troubleshooting

Many things can happen when APS is not properly configured. A few of the more common issues are described below.

In all cases, the first thing that should be examined is SiteMinder's Authentication Service Console Log on the Policy Server. APS records all significant activity into this log. If something is happening that you suspect should not be happening, always check this log first.

Always make sure that you have applied all of the latest patches/service packs to your APS installation. Often, a given problem has already been found at another site and fixed in a patch. Patches are available for download from the Netegrity support site.

If you still cannot resolve the issue, open a case using Netegrity's support Web Site. In order to optimize response time from Netegrity, be sure to do the following:

1. Make sure that the case clearly identifies the issue as an APS issue (both in the text and as the product/component).
2. Make sure to identify the correct versions of APS, SiteMinder and your Operating System platform(s).
3. Attach your entire APS.cfg and SmPortal.cfg files to the case.
4. Attach a complete copy of the SiteMinder Authentication Service Console Log to the case. Please do not just cut/paste important lines, sometimes the issue requires that we see the entire authentication process.

   A site can request that additional information (traces) be written to this log. To do so, set the "Trace" keyword in the APS Configuration File. To turn off this tracing, delete the "Trace" keyword.

   In APS Version 4.0, all of the entries placed into the Authentication Log were reviewed and their format standardized. Each specifically identifies its source as APS. Please note that it would be impossible to document the information logged, since it changes from release to release.
5. Be sure to properly prioritize your case. The rules of thumb are:
   a. Priority ONE is reserved for PRODUCTION SERVER OUTAGES ONLY and require immediate attention.
   b. Priority TWO is for serious outages that will impact rollouts, are holding up development or testing efforts, or are considerable inconveniences in production (such as requiring frequent server restarts, etc).
   c. Priority THREE is for all other issues.
   d. Capabilities questions (such as "Can APS do this...?") should be directed to your sales engineer or sales representative, rather than logged to our support site.

Unfortunately, a few customers have been known to overstate the priority of their cases intentionally, in an effort to get more immediate response.

Rather than being productive, this has two adverse effects:

- It will cause the resolution of other customers' issues to be delayed as we deal with the higher priority case. Please be consciencious to other customers.
- Support becomes aware, over time, of customers that overstate priorities. After awhile, such priorities may not be taken as seriously as they should be, since Support may not be able to differentiate

This section contains the following topics:

Connectivity Problems (see page 446)

Connectivity Problems

Various APS components communicate with the Policy Server via SmPortal/SmTransact. This is the troubleshooting section for those connectivity issues.

This kit provides a comprehensive set of error codes that can be used to determine and resolve problems with services.

These errors are distinct from application errors and should not be confused with them. The errors displayed by SmPortalTest are all tunnel errors and that makes SmPortalTest a good tool for validating configuration.

A zero error return indicates success (though the service itself may have failed and thus returned a non-zero application error).

Negative numbers indicate client side (SmPortal) errors. These errors typically indicate an invalid configuration.

Positive errors indicate an error that occurred on the Policy Server when SmTransact attempted to execute the Service Provider (back-end) code. These are typically errors in service code.

A typical problem is that either multiple copies of SmPortal.cfg exist or that it does not exist on the same directory with SmPortal.dll. Only a copy on the same directory with SmPortal.dll will ever be used. Be careful of multiple copies of SmPortal.dll, because then it is difficult to determine which copy Windows elected to use.

Client Side (SmPortal) Errors

AGENT_FAILURE(-1)

This is, by far, the most frequent error encountered. It indicates that the SiteMinder Policy Server could either not be contacted (rare) or that the Policy Server refused the connection (very common). This is caused by either the agent not being defined on the Policy Server or a mismatch on the shared secret.

The best way to troubleshoot this error is to examine the SiteMinder console log (usually of the Authentication service), looking for entries denying a connection by an agent. The messages there are often cryptic, so check carefully the definition of the agent, especially the name of the agent (it is case-sensitive) and its shared secret.

This error should be considered equivalent to error -17, described below. Different versions of the SiteMinder Agent API return different errors, so you might see either error.

NOT_IMPLEMENTED (-11)

The application requested a function that is not currently implemented in SmPortal. This generally means that the service either requires a later version of SmPortal or that there is a significant problem with the application. Contact your application (or API) developer for resolution.

ERROR (-12)

This error indicates that a fault was detected during processing. This generally indicates a programming error.

ACCESS_DENIED (-13)

Indicates that the application program is not authorized to call into the Portal. This indicates a programming error in the application or API. Contact your application (or API) developer for resolution.

NO_AGENT (-14)

This error indicates that the Agent entry specified for the service could not be located in SmPortal.cfg. Check to make sure that the agent configured for this service is defined in the file.

NO_SECRET (-15)

SmPortal was unable to determine the shared secret for the specified agent. Check to ensure that a secret is defined for this service's agent.

NO_SERVERS (-16)

This error indicates that there are no servers defined for this agent in the SmPortal.cfg file.

AGENT_INIT (-17)

This error indicates that the SiteMinder Agent API could not be initialized. Typically, this error is returned if:

1. The server could not be found at the specified IP address;
2. The wrong port is configured for the server;
3. The agent name is undefined on that SiteMinder Policy Server;
4. The agent defined on the SiteMinder Policy Server should not be coming from that IP address;
5. The shared secret is wrong.

The Agent API intentionally does not distinguish between these errors. To do so would make cracking security significantly easier. To determine which of these cases is the actual cause, examine the Authentication Server's console log (on the Policy Server).

NO_SERVER_IP (-18)

This error indicates that one of the servers defined for the agent was not defined elsewhere in the SmPortal.cfg file. Make sure that all of the servers defined for the Agent are defined with, minimally, an IP address (and that the IP address is not blank).

INVALID_CALL_HANDLE (-19)

This is an internal application error indicating that the application passed an invalid handle to the SmPortal API. This indicates a programming error in the application or API. Contact your application (or API) developer for resolution.

NO_DYNAMIC_LIBRARY (-20)

The dynamic library (SmPortal.dll on Windows NT, libSmPortal.so on Solaris) could not be located. Update your PATH (Windows) or LD_LIBRARY_PATH (Solaris) environment variable *in the context of the running user* so that the dynamic library can be found. This error may also occur if the library can be located, but cannot be loaded because of another dependency.

NO_CONFIGURATION (-21)

The SmPortal configuration file (usually SmPortal.cfg) could not be located or opened. This usually means that the file either doesn't exist, is in the wrong place, or there is a missing or improperly set environment variable.

Server Side (SmTransact) Errors

All of these errors indicate a problem on the Policy Server.

INPUT_TOO_SMALL (1)

This error occurs when attempting to communicate with SmTransact without using SmPortal. It should never occur under normal circumstances.

NO_SERVICE (2)

This error occurs when attempting to communicate with SmTransact without using SmPortal. It should never occur under normal circumstances.

NOT_INSTALLED (3)

The service is not installed on the Policy Server.

BUFFER_VIOLATION (4)

This indicates a programming error in the service provider. Contact your service provider developer for resolution.

APPLICATION (5)

This indicates a programming error in the service provider. Contact your service provider developer for resolution.

BAD_SERVICE (6)

This indicates a programming error in the service provider. Contact your service provider developer for resolution.

LIB_PROBLEM (7)

This indicates a programming error in the service provider. Contact your service provider developer for resolution.

OUTPUT (8)

This indicates a programming error in the service provider. Contact your service provider developer for resolution.

NOBUFFER (9)

This indicates a programming error in the service provider. Contact your service provider developer for resolution.

CHUNKING (10)

This indicates a programming error in the service provider. Contact your service provider developer for resolution.

LIB_INIT (11)

This indicates a programming error in the service provider. Contact your service provider developer for resolution.

LENGTH_MISMATCH (12)

This indicates a programming error in the service provider. Contact your service provider developer for resolution.

VERSION (13)

Indicates that the version of SmPortal.dll and the version of SmTransact.dll do not match. You will need to upgrade one or the other.

SiteMinder Authentication Log

SmTransact keeps logs of virtually all of its activities. Most operational activity and errors are logged in the SiteMinder Policy Server console log (by default, in the Authentication Service log).

If trace and debug logging are enabled for a service, the messages are also placed into the SiteMinder console log.

New passwords chosen by users are always rejected

The password policy may be too strict or improperly configured. Review your password policy defined in APS.CFG. Check the Content Minimums options for consistency: if the sum of *Letters* plus *Digits*, for example, is greater than *Maximum Password Length*, no password could meet the criteria and all passwords will be rejected. Use the APSTestSettings utility to determine the settings in effect for the user having the problem.

Also check the SiteMinder Authentication Service Console Log. APS almost always logs even more information about a password content rejection to the log that it displays to the user. For example, if the password is rejected because it matches all or part of the user's profile, the actual attribute that matched is logged to the console log.

User Accounts are mistakenly disabled

No users in any user directory may have exactly the same username. If accounts with identical usernames exist in these separate directories, some or all of these accounts may be disabled at random. See the description associated with the "Failure Count Timeout" setting in the APS Configuration File for more information, restrictions and related issues.

All (or many) accounts are disabled when a single user fails authentication

This occurred at one site when the User prefix/suffix combination was incorrectly defined in the User Directory entry in SiteMinder's Policy database. The prefix had a wildcard that caused all (or many) user accounts to be checked during authentication.

APS does not disable LDAP user accounts

LDAP directories must not be read-only. Advanced Password Services saves information to each user's record in the LDAP directory, so the directory must not prevent active writing.

If you are using multiple LDAP Directory servers in a Master/Slave replication arrangement, you will have to use the write back settings. See the associated settings in the chapter entitled The APS Configuration File.

Automated Email Notification does not send email

Verify that your SMTP settings are configured correctly. Check the SiteMinder Authentication Server Console log (with maximum tracing enabled) to see if any messages are logged from Advanced Password Services or from APSMail. Use APSMailTest to determine if your settings are correct. Try logging mail to a log file.

Verify that the automated email settings in the APS Configuration File are set properly.

Ensure that an email file exists for the proper event for the particular language and that its format is correct and complete. You can use APSMailTest to check this.

Check the error logs on your SMTP server.

Windows NT Users cannot change passwords

When using Windows NT domains, ensure that the global password policy does not require users to log in to the domain in order to change their passwords. CA is working with Microsoft to determine why 0x8007054B (and other) errors occur when this flag is set and the user attempts to change their own password. This problem *only* occurs for Windows NT domains; it does not occur for other directory types.

"Change Password Service is not configured correctly"

This error message is displayed by SmCPW if the SMCPW response is not connected (or returning data) to the SmCPW page. SmCPW requires that a specific SiteMinder response be sent to it by SiteMinder. This response is an Active Attribute obtained via a call into the "smaps" library to a function called "SMCPW". If this response does not exist, is improperly formed, or is not connected to the SmCPW rule within the SiteMinder Policy, this error will occur. Note that the text of the error may vary if the Language file translates the text.

Error reading user entry ("Not Found") during password change

This is usually caused by bad Access Control Information (ACI) that prevents a user from reading their own record. During password change operations, APS uses the user's own credentials to bind to the LDAP directory. If these credentials have insufficient access, the Directory Provider will report that the record could not be found. There is no way for APS to identify the difference.

User is asked to authenticate during the FPS process

This means that the presentation pages or one of their components are protected. Check the SiteMinder *authorization* (AZ) log to see what page is being accessed.

Usually, the culprit is a cascading style sheet, a frame, or a graphic.

Index

@

@<classname> • 72

[

[Complexity] • 130
[Custom] • 129
[Dictionary] • 129
[FPS-Change] • 175
[FPS-Confirm] • 176
[FPS-Errors] • 186
[FPS-Identify] • 155
[FPS-No Mail] • 187
[FPS-Verify] • 165
[Logging] • 128
[Mail] • 126
[ODBC] • 136

A

A note about • 69
About Firewalls • 402
About Password Security • 397
Access Control • 285
ACCESS_DENIED (-13) • 447
Account Expiration • 266
Account Expiration & Purge • 111
Account Inactivity Warning • 266
Active Expression Parameters • 113
Add FPS Log • 145, 245
Add To Group • 148, 248
Additional Advantages • 426
Additional Arguments • 413
Addressing Mail • 383
Addressing the Message • 374
Admin Translation • 149, 248
Administration and Operations • 255
Advanced Message Control • 377
Advanced Password Services • 18
AGENT_FAILURE(-1) • 447
AGENT_INIT (-17) • 448
All (or many) accounts are disabled when a single user fails authentication • 451
Allowed • 154
Allowed Characters • 92, 180
Alternative Authentication Schemes • 427
Alternative Character Sets • 377
Always Redirect • 120
Always Redirect AZ • 120
APPLICATION (5) • 449
Application Macros • 368
Application Programming Interface • 319
Application Programming Interface (APSAPI) • 22
APS Attributes/Enable/Disable (all platforms) • 45
APS Configuration File (APS.CFG) • 61
APS does not disable LDAP user accounts • 451
APS Processing during User Authentication • 256
APS.cfg configuration • 361
APSAdmin • 219, 231, 237, 251, 394
APSAdmin (Help Desk Interface Stub) • 55
APSAdminGetDirectory • 331
APSAdminRead • 333
APSAdminWrite • 337
APSComplexity - Calculate Password Complexity • 347
APSEncrypt - Encrypt an Administrator Password • 349
APSExpire • 150, 219, 231, 236, 251, 297
APSExpire Event Log • 299
APSForcePWChg - Set Force Password Change Flag • 352
APSMAIL_COMMLOG • 366
APSMAIL_LOCALHOST • 366
APSMAIL_LOGIN • 366
APSMAIL_LOGPATH • 366
APSMAIL_PATH • 366
APSMAIL_SERVER • 366
APSMailTest - Test Email Settings • 349
APSPing • 415
APSPing - Test connectivity • 350
APSTestSettings - Check APS Configuration • 348
APSVersion - Display Module Version Numbers • 350
APSXlateTest • 418
APSXlateTest - Test Translation Settings • 350
Attribute Length • 238
Attribute Mapping • 134
Attribute Match Maximum • 96
Attribute Suppression • 135
Attribute/Column Names • 190
Audit Log • 152
Authenticate • 330

Authenticate User • 138, 240
Authentication Scheme • 60, 341
Authorization Mapping and AZRedirect • 267
Auto Force Change • 98
Auto Reset Failure Count • 102
Automated Email Notification does not send email • 451

B

Backup Policy Servers • 263
BAD_SERVICE (6) • 449
Basic Message Control • 373
Best Practice Recommendation • 386
Best Practices for Storing Legacy/Back-End Credentials • 423
BUFFER_VIOLATION (4) • 449
Button Definitions • 421

C

CA Identity Manager • 362
Case Switch • 131
Change Confirmation Mail • 125
Change My Password (SmCPW) • 19
Change Pages • 314
Change Password Forms • 271
Change Password Interface (SmCPW) • 50, 271
Change Password Page Configuration • 56
Change Password Service is not configured correctly • 452
Change Required • 94
ChangeMyPassword • 326
ChangeMyPassword2 • 328
Changing the Graphics • 284
Changing the Prompts • 280, 284
Character Values • 132
Check LDAP Password Existence • 77
Checking at Authentication (SmAPS) • 20
CHUNKING (10) • 450
Clear FPS Log • 145, 245
Clear Login History • 144, 244
Client Side (SmPortal) Errors • 447
Collecting Identification Information • 155
Combination Alphanumeric • 87
Combination Digits • 86
Combination Letters • 85
Combination Lowercase • 86
Combination Other • 88
Combination Punctuation • 87
Combination Symbols • 87
Combination Uppercase • 86
Combinations of Character Types • 85
Command Line Options • 348
Command Line Parameters • 301
Comments • 370, 420
Communications Timeouts • 371
Compare FPS Answer • 147, 247
Compiling & Linking • 319
Complexity • 89
Configuration • 49, 365, 405
Configure Required Parameters in the APS.cfg File • 106
Configure the Max Failures Parameter • 106
Configure the Override Max Failure for Realm OIDs Parameter • 107
Configure the Realm Based Max Failures Parameter • 107
Configuring as a Wedge • 341
Configuring for FPS • 342
Confirm Pages • 315
Connectivity Problems • 446
Contact CA Technologies • 3
Context Macros • 69
Continuation Lines • 420
Controlling Frequency of Use • 160
Cracking Passwords • 397
Custom Extensions • 128
Custom Extensions (SmAPSEx) • 345
Custom Forms • 53
Custom Logging • 128
Custom Logging Extension (SmAPSLog) • 343
Custom X-MAIL Headers • 378
Customization Options • 278, 282
Customization Required for Supporting Internationalization • 361
Customization without Custom Forms • 272
Customizing the Form • 54
Customizing the Look and Feel • 280, 284

D

Daily Processing (APSExpire) • 297
Date Format • 190
Define • 71
Define Macros • 63, 368
Defining Macro Substitutions • 370
Delegated Management Services (DMS2) • 358

Delegated Management Services (DMS2) Interface • 45
Delete Login History • 143, 244
Delivery Problems • 388
Details • 425
Directives • 369
Directory • 127, 152
Disable • 80
Disable Redirect • 116
Disabled Mail • 157
Disabled URL • 157, 309
Disabled User Mail • 121
Disabling a User • 216, 228
Disabling a User Account • 235, 250
Disabling, Enabling & Redisabling • 293
Disabling/Enabling User Account • 249
Disabling/Enabling User Accounts • 215, 226, 235
Disallowed Characters • 92, 180
DMS Workflow Library • 358
Documentation Changes • 4
DON'Ts • 398
DOs • 398

E

Email • 395
Email Templates • 367
Enabling a Disabled User Account • 360
Enabling a User Account • 217, 229, 235, 250
Enabling and Disabling Accounts • 355
Enabling and Disabling User Accounts • 358
Ensuring Password Validity • 356
Enumerate • 137, 240
Environment Variables • 368
ERROR (-12) • 447
Error Cases • 156
Error Conditions • 167
Error Pages • 318
Error reading user entry (• 452
Event Handling • 58
Event Processing • 300
Event Redirection • 113
Exclude Attributes • 97
Executive Summary • 423
Expiration Grace • 110
Expiration Warning • 109
Expire Change Redirect • 118
Expired Password Mail • 123
Expired Redirect • 117
Expiring Password • 108
External Macro File • 64

F

Failure Count Retention • 103
Failure Count Timeout • 102
Failure Redirect • 116
Failure Tracking • 99
Fault Tolerance • 365
Fault Tolerance & Performance • 219, 231, 237, 251
Field Mappings • 133
File Sections • 70
Force Case • 90, 181
Force Change • 179
Force Change Mail • 123
Force Change Redirect • 117
Forcing Password Changes • 292
Forgot (Forgotten Password User Interface Stub) • 54
Forgotten Password (FPS) Interface (Forgot) • 305
Forgotten Password Configuration • 57
Forgotten Password Questions/Answers (all platforms) • 46
Forgotten Password Services (FPS) • 21, 219, 230, 236, 251
Forgotten Password Services (FPS) Data • 208
Form Handling • 169
Form Processing • 158
Format • 62
Format for the Property File • 361
Formatting Problems • 388
Forms • 165
FPS Change Password Process • 175
FPS Confirm Process • 176
FPS Errors • 186
FPS Identify Process • 154
FPS LDAP Settings • 154
FPS Process Flow • 267
FPS Verify Process • 165
Function Descriptions • 320

G

General FPS Settings • 151
General Settings • 72
Generational & Other Automatic Redirection • 118
Generational Redirect • 119
Get FPS Log • 144, 245
Get Login History • 142, 243

Get Object Info • 137, 240
Get User Groups • 140, 242
Get User Properties • 140, 241
Get User Property • 138, 240
Grace Logins • 110
Grouping into Sections • 288
Groups • 286

H

Handling Errors • 273
Handling FPS Mail Errors • 187
Handling Returned Mail • 384
Header Keywords • 420
Help Desk Interface (APSAdmin) • 277
Help Desk Interface Configuration • 57
Help Desk Support (APSAdmin) • 21
How APSMail Uses Templates • 367
How Do I
- Allow a user to select their own password during FPS? • 438
- Automatically login the user using FPS? • 438
- Configure • 432
- Customize the change password form? • 432
- Detect & Prevent FPS Misuse? • 437
- Determine why a user's authentication is rejected? • 434
- Eliminate the current password from the change password form? • 436
- Enable an account that APS has disabled? • 435
- Force users to change their password at next login? • 434
- Get the FPS challenge questions into the User Directory? • 441
- Integrate the APSAdmin interface into my own Identity Management application? • 441
- Limit password changes to once per day? • 435
- Lock FPS (mis) users out of our site? • 437
- Make the APS API return error codes instead of error text? • 442
- Re-enable FPS for an account? • 437
- Retrieve statistics about FPS usage? • 438
- Retrieve the date & time that a user last logged in? • 435
- Securely support FPS challenge questions? • 439
- Set up passwords to expire? • 430
- Use a backup LDAP server (replicant)? • 436
- Use a consumer LDAP server (satellite)? • 437

How Do I? • 429
How to Configure a Separate Maximum Failure Counter and Threshold for OTP Authentication • 103

I

Identify Pages • 307
Ignore • 79
Inactive Redirect • 117
Inactive User Mail • 123
Inactivity Disabled Mail • 125
Inactivity Warning • 112
Inactivity Warning Mail • 124
Including Another Template • 370
Init User • 138, 240
Initial • 171, 178
Initial Invocation • 306
INPUT_TOO_SMALL (1) • 449
Installation • 32, 402
Installation & Configuration • 31, 417
Integrating APS with User Management Tools • 355
Integrating Forgotten Question and Answers • 361
Integration • 355
Interfaces for Delegated Management Services (DMS2) • 23
Intermediate Message Control • 375
Internationalization • 360, 391
Internationalization & Localization (APSXLate) • 417
Internationalization/Localization • 381
Introduction • 17, 255, 363, 391, 401, 417
Invalid Mail • 167
Invalid Password Dictionary • 129
Invalid URL • 167, 313
INVALID_CALL_HANDLE (-19) • 448
Is Group Member • 141, 242
IsMyPasswordValid • 322
IsMyPasswordValid2 • 324
IsPasswordValid • 320
IUsing [INCLUDE] • 381

K

Keeping Passwords Safe • 398
Known Directory Idiosyncrasies • 221, 233, 237, 252

L

LDAP Blob • 77
LDAP Directories • 150, 211, 302
LDAP Disabled Groups • 76
LDAP Password Attribute • 78

LDAP Reverse Groups • 135, 218, 230
LDAP Updates • 49
LDAP Write Back Information • 74
LDAP Writeback DN • 75
LDAP Writeback Password • 76
LDAP Writeback Server • 75
LDAP Writeback Use SSL • 76
Length • 131
LENGTH_MISMATCH (12) • 450
LIB_INIT (11) • 450
LIB_PROBLEM (7) • 449
Limitations • 389
Local Machine Name • 372
Lockout Count • 163
Lockout Mail • 164
Lockout Timer • 164
Lockout URL • 163, 310
Log Files • 418
Log Path • 127
Logging Path • 371
Lookup • 137, 159, 170, 240
Lookup User • 140, 241
Lookups • 368

M

Macros • 367
Mail • 177, 187, 188
Mail Error Pages • 318
Mail Macros • 384
Mail Server Login • 372
Mail Server Security • 372
Mail Settings • 125
Maintaining Forgotten Password Services (FPS) Questions/Answers • 357
Making the User Display Read Only • 279, 282
Match • 90
Max Attempts Frequency • 161
Max Failures • 99
Max Failures Mail • 122
Max Failures On Change • 101
Max Inactivity • 111
Max Success Frequency • 160
Maximum Length • 81, 182
Maximum Repeat • 89, 185
Message Body • 378
Message Formatting • 377
Message of the Day • 120
Message Subject • 373
Microsoft Active Directories • 223
Microsoft Windows Domains • 233
Minimum Alphanumeric • 83, 183
Minimum Combinations • 88
Minimum Digits • 83, 183
Minimum Length • 81, 181, 182
Minimum Letters • 82, 183
Minimum Lower Case • 182
Minimum Lowercase • 82
Minimum Other • 84, 185
Minimum Punctuation • 83, 184
Minimum Symbols • 84, 184
Minimum Uppercase • 82
Missing URL • 155, 166, 308, 312
Multiple URL • 156, 308

N

Native Data Formats • 239
Native Password Policy Support • 214, 225, 234
Network Latency • 364
New Password Length • 179
New passwords chosen by users are always rejected • 450
No Data Mail • 168
No Data URL • 167, 314
NO_AGENT (-14) • 447
NO_CONFIGURATION (-21) • 448
NO_DYNAMIC_LIBRARY (-20) • 448
NO_SECRET (-15) • 447
NO_SERVER_IP (-18) • 448
NO_SERVERS (-16) • 448
NO_SERVICE (2) • 449
NOBUFFER (9) • 449
NoMatch • 91
Not Found URL • 156, 309
NOT_IMPLEMENTED (-11) • 447
NOT_INSTALLED (3) • 449
Null Sections • 289
Number of Rows • 239

O

ObjectClasses • 48
Obtain the OID of Realms Protected Configured for OTP Authentication • 105
ODBC (RDBMS) Directories • 237
ODBC Directories • 151, 303
ODBC Encrypt • 174
ODBC Queries • 49, 136

Offline Expiration Warning • 109
Ongoing Failures Mail • 122
Operation • 404
Optional • 158, 169
Optional Agent Settings • 411
Optional Server Settings • 409
Optional Service Settings • 412
Other Content Requirements • 89
Other Field & Group Handling • 295
Other LDAP Settings • 76
Other User Directory Changes • 49
OUTPUT (8) • 449
Overview • 61

P

Parse Attributes • 96
Password Complexity • 130
Password Content Settings • 80
Password Expiration • 108
Password Expiration Warnings • 266
Password Format & Lifetime • 426
Password Length • 81
Password Lifetime • 260
Password Replication/Synchronization • 252
Password Reuse • 93
Password Warning Mail • 124
Percentage • 95
Percentage Sequencing • 95
Performance • 69, 303, 364
Performance Adjustments/Job Definitions • 302
Policies • 59
Policy Database Configuration • 55
Poll • 73
Preparation & Prerequisites • 31
Preventing Attacks • 163
Programming Notes • 305
Prompts • 289
Purge After • 112

Q

Query String Options • 273

R

Realms • 58
Recognizing a Disabled Account • 235, 249
Recognizing a Disabled User • 215, 226
Redirection • 394
Remove From Group • 148, 248
Renaming LDAP Groups • 135
Replace Quote With • 136
Requesting Delivery Receipts (Disposition Notification) • 376
Requesting Read Receipts (Return Receipts) • 375
Required • 158, 169
Required Content • 82
Reset FPS Lockout • 99
Resource Utilization • 365
Responses • 59
Restricting the Directory or Directory Type • 278
Restricting Words from the User's Profile • 95
Retry URL • 166, 313
Return Addresses • 383
Reuse Count • 94
Reuse Delay • 94
Rule Groups • 59
Rules • 58
Running APSAdmin from the Command Line • 295
Run-time Password Checking • 98
Run-Time Requirements • 320

S

Schema & Storage • 189
Schema Changes • 47
Schema Updates • 211, 223, 234, 237
Security Note • 296
Sending Blind Carbon Copies • 374
Sending Carbon Copies • 374
Sending Mail • 121
Server • 126
Server Communications Control • 371
Server Side (SmTransact) Errors • 449
Set Login History • 143, 243
Set Password • 141, 242
Set Password Checksum • 146, 246
Set User Property • 139, 241
Setting Overrides • 65
Settings That Can Appear Anywhere • 71
Setup • 53, 275
Showing a Cancel Button • 280, 283
Showing a Close Button • 283
Showing a Help Button • 279, 283
Simple Mail Transfer Protocol (SMTP) • 363
SiteMinder (Basic) Password Services & Advanced Password Services • 23
SiteMinder Authentication Log • 450
SiteMinder Policy Server • 32

SiteMinder's [Basic] Password Services • 265
smapsAccountInactivityDays • 190
smapsBaseDate • 191
smapsDisableAfter • 192
smapsDisableUntil • 193
smapsExpirePasswordDays • 193
smapsFailureCount • 194
smapsFailuresSinceLastLogin • 195
smapsFailuresSincePreviousLogin • 196
smapsGenerationalRedirects • 197
smapsGraceLoginsUsed • 197
smapsHistory • 198
smapsImmediateChange • 198
smapsInactivityWarning • 199
smapsLastLogin • 199
smapsLastPasswordChange • 200
smapsLoginHistory • 200
smapsMaxFailures • 201
smapsMustLoginBy • 201
smapsNextAction • 202
smapsOldBlob • 203
smapsPassword • 203
smapsPreviousLogin • 204
smapsTotalFailures • 204
smapsTotalLogins • 205
SmCPW • 116, 393
SmCPW - Load Test Password Changes • 351
SmCPW, APSAdmin & Forgot (User Interfaces) • 393
smfpsLockoutCounter • 206
smfpsLog • 206
smfpsOneShot • 207
SmPortal - The Client Side • 402
SmPortal/SmTransact Installation and Configuration • 401
SmPortalTest • 413
SmPortalVfy • 415
SmPortalVfy - Verify SmPortal Configuration • 353
SMTP (Mail) Servers • 380
SmTransact - The Policy Server Side • 403
Sorted • 130
Special APS Processing/Limitations • 218, 229, 236, 251
Special Case
  Three Strikes, You're Out • 263
Special Field & Group Handling • 294
Special Setup • 214, 225, 234
Special Setup (Queries) • 239
Specifying Attributes/Columns • 287
Specifying Form Content and Access Control • 285
Specifying Groups • 287
Specifying the Mail Server • 371
Specifying the Message Priority • 377
Specifying the Reply To Address • 375
Specifying the Sender's Organization • 376
Specifying Where to send Delivery Errors • 375
Strategies • 355
Strategies & Best Practices • 378
Summary • 427
Suppressing the Reset Button • 280, 283
Suppressing the User Selection Panel Altogether • 281

T

Template Processing Control • 370
Test Handle • 137
Test Password Checksum • 147, 246
Testing Expressions • 70
The Agent Section • 410
The Communications Log • 388
The FROM Address • 373
The Send Log • 387
The Sender • 373
The Server Section • 408
The Service Section • 411
The SiteMinder Console Log • 387
Timeout • 73, 176, 179
Timeout URL • 168, 175, 314, 315
Timer • 174
To use 56NetegrityAPS.ldif on iPlanet LDAP Directories Only: • 214
Too Recently Attempted Mail • 162
Too Recently Attempted URL • 162, 310
Too Recently Used Mail • 161
Too Recently Used URL • 160, 310
Trace • 72
Translating Prompts • 289
Translation File Format • 420
Translation Files • 419
Translations • 422
Triggering APS Events Properly • 357
Troubleshooting • 387, 445
True password expiration • 266

U

UNIX • 33, 36, 40, 43
Updating Application Credentials • 426
Updating User Credentials • 426

URL • 155, 165, 175, 177, 186, 187, 307, 311, 315
Use External Macros • 71
Use Internal Disables • 78
User Accounts are mistakenly disabled • 451
User Classes • 68
User Directories
  Schema, Storage and Capabilities • 189
User Directory • 286
User Directory Credentials • 55
User Display • 282
User Entries • 48
User is asked to authenticate during the FPS process • 452
User Presentation Forms • 54
User Properties • 140, 241
User Selection • 278
User Update • 285
Using [REM] • 380
Using Email • 363
Using Persistent Cookies • 266
Using Self() • 291
Utilities • 347

V

Validating Password Resets • 292
Varying Content by Context • 290
Varying Content by User • 290
Verification • 412
Verify Forms • 311
Verify That Attempt Failure Thresholds for OTP and Other Authentication Do Not Aggregate to Lock Out Users • 107
VERSION (13) • 450

W

Warning Redirect • 118
Web Server – Change Password Interface • 35
Web Server – Forgotten Password Interface • 38
Web Server – Help Desk (APSAdmin) Interface • 42
Well Known SiteMinder Attributes • 369
Windows • 32, 35, 38, 42
Windows NT Users cannot change passwords • 452